Università degli Studi di Verona

Doctoral Thesis

Hyper Static Analysis of Programs

An Abstract Interpretation-Based Framework for Hyperproperties Verification

Author:
Michele Pasqua

Advisor:
Prof. Isabella Mastroeni

Defence commettee members:
Prof. Maurizio Gabrielli[‡]
Prof. Isabella Mastroeni[⋆]
Prof. Antoine Miné[†]

Thesis referees:
Prof. Antoine Miné[†]
Prof. David A. Naumann[§]

Local commettee members:
Prof. Roberto Giacobazzi[⋆]
Prof. Isabella Mastroeni[⋆]
Prof. Roberto Segala[⋆]

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

at the

Department of Computer Science

May 15, 2019

[†] Sorbonne Université - Paris (France)
[§] Stevens Institute of Technology - Hoboken (USA)
[⋆] Università degli Studi di Verona - Verona (Italy)
[‡] Università di Bologna - Bologna (Italy)

UNIVERSITÀ
di **VERONA**
Dipartimento
di **INFORMATICA**

Declaration of Authorship

I, Michele Pasqua, declare that this thesis titled, "Hyper Static Analysis of Programs: An Abstract Interpretation-Based Framework for Hyperproperties Verification" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Pasqua Michele

Date: May 15, 2019

"The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise."

Edsger W. Dijkstra

Università degli Studi di Verona

Abstract

Department of Computer Science

Doctor of Philosophy in Computer Science

Hyper Static Analysis of Programs

An Abstract Interpretation-Based Framework for Hyperproperties Verification

by Michele Pasqua

In the context of systems security, information flows play a central role. Unhandled information flows potentially leave the door open to very dangerous types of security attacks, such as code injection or sensitive information leakage. Information flows verification is based on a notion of dependency between a system's objects, which requires specifications expressing relations between different executions of a system. Specifications of this kind, called hyperproperties, go beyond classic trace properties, defined in terms of predicate over single executions. The problem of trace properties verification is well studied, both from a theoretical as well as a practical point of view. Unfortunately, very few works deal with the verification of hyperproperties. Note that hyperproperties are not limited to information flows. Indeed, a lot of other important problems can be modeled through hyperproperties only: processes synchronization, availability requirements, integrity issues, error resistant codes check, just to name a few.

The sound verification of hyperproperties is not trivial: it is not easy to adapt classic verification methods, used for trace properties, in order to deal with hyperproperties. The added complexity derives from the fact that hyperproperties are defined over sets of sets of executions, rather than sets of executions, as happens for trace properties. In general, passing to powersets involves many problems, from a computability point of view, and this is the case also for systems verification.

In this thesis, it is explored the problem of hyperproperties verification in its theoretical and practical aspects. In particular, the aim is to extend verification methods used for trace properties to the more general case of hyperproperties. The verification is performed exploiting the framework of abstract interpretation, a very general theory for approximating the behavior of discrete dynamic systems.

Apart from the general setting, the thesis focuses on sound verification methods, based on static analysis, for computer programs. As a case study – which is also a leading motivation – the verification of information flows specifications has been taken into account, in the form of Non-Interference and Abstract Non-Interference. The second is a weakening of the first, useful in the context where Non-Interference is a too restrictive specification.

The results of the thesis have been implemented in a prototype analyzer for (Abstract) Non-Interference which is, to the best of the author's knowledge, the first attempt to implement a sound verifier for that specification(s), based on abstract interpretation and taking into account the expressive power of hyperproperties.

Contents

List of Figures

Preface

The present thesis is the result of my research work, developed at the University of Verona during part of my PhD. Indeed, I spent most of my first year (out of three) working in a different research project, leading to publications [Dalla Preda and Pasqua, 2016] and [Dalla Preda and Pasqua, 2018], but not related to the present thesis. Nevertheless, in the remaining two years I was able to obtain some results about hyperproperties verification, namely the topic of the present thesis.

The thesis is organized in nine chapters and one appendix, containing long proofs. Chapter 1 serves as an introduction of the problem I addressed. It contains also a description of the contributions and a conceptual road-map guiding the reader through the central part of the thesis. Chapter 2 contains notions and mathematical notations required to fully understand the concepts showcased in the thesis. From Chapter 3 to Chapter 8 we have the main contributions of the thesis, interleaved with already known results and concepts. I have chosen this mixed presentation in order to make the text more easy and, hopefully, enjoyable to read. Nevertheless, my contributions are well marked within any chapter. Finally, Chapter 9 concludes and discusses future research directions.

Most of the contents of this thesis have been published in international peer-reviewed conferences and they have been developed under the supervision of Isabella Mastroeni, Associate Professor at the University of Verona. In the rest of the work I adopt the academic "we" instead of "I".

The first version of the thesis has been submitted for the reviewing process in December 15, 2018. The present text is the revised and corrected version.

Verona – May 15, 2019

1 Introduction

Concerning systems security, a fundamental problem is how to protect the confidentiality or, dually, integrity of data manipulated by programs. Namely, it is important to guarantee that no confidential information can be caught by unauthorized *attackers* observing the executions of a system. The standard way used to protect private data is *access control*. As the name indicates, access control verifies the system rights at entry-point and it does not take into account the propagation of information. This is inadequate in many situations, in fact the system may leak/compromise information after the access check. Instead, *information flow control* tracks how information propagates through the system during execution, to make sure that it handles the information securely. Unhanded information flows potentially leave the door open to very dangerous types of attacks, such as code injection or sensitive data leakage. So, it is important to build verification mechanisms, aiming to check if systems comply with a given confidentiality/integrity specification, stating which flows of information are allowed and which are not.

The problem of information flows control is far from a solution. There are several difficulties concerning information flows verification. First of all, the problem is undecidable, hence a form of approximation is mandatory. Unfortunately, also approximated verification is not easy for information flows, since they are formalized as *hyperproperties*. This means that they cannot be verified on single executions, since they require the comparison of multiple computations: this makes hard to exploit standard verification techniques. Indeed, whilst for trace properties standard techniques can be used with an acceptable degree of approximation, the verification of hyperproperties is a hard problem to solve in a sufficiently precise way. Being a hyperproperty means to be modeled as a set of sets of executions and, indeed, the extra level of sets introduces a lot of technical problems for approximation-based verification methods. Hence, in order to obtain significant results w.r.t. analysis precision, we need new verification methods which approximate sets of sets (instead of just sets) of executions, namely we need verification methods for hyperproperties.

Note that, information flows control is the leading motivation, but hyperproperties can express a lot of other useful specifications, like availability requirements, process synchronization problems, cryptographic protocols checks, etc. The latter, as well as information flows, cannot be expressed with classic trace properties.

The goal of the thesis is to face the problem of verifying hyperproperties, namely to develop a methodology which goes beyond the standard trace properties. In order to cope with this problem, we need to reason about systems semantics, namely the representations of systems executions, and hence we rely on *abstract interpretation*, which is a general framework for the sound approximation of the semantics of discrete dynamic systems.

Since its origin in 1977, abstract interpretation has been widely used, implicitly or explic-

itly, to describe and formalize approximate computations in many different areas of computer science, from its very beginning use in formalizing (compile-time) program analysis frameworks to more recent applications in model checking, program verification, comparative semantics, malware detection, code obfuscation, etc. When reasoning about systems executions a key point is the degree of approximation given by the choice of the semantics used to represent computations. In this direction, comparative semantics consists in comparing semantics at different levels of abstraction, always by abstract interpretation. The choice of the semantics is a key point, not only for finding the desirable trade-off between precision and decidability of program analysis in terms, for instance, of specifications verification, but also because not all the semantics are suitable for proving every possible specification of interest. This means that the specification to verify necessarily affects the semantics we have to choose for modeling the system to analyze.

Hyperproperties verification is a new research topic and so there are very few works about it. For this reason, we started reasoning about hyperproperties from the roots, namely from a structural point of view. In particular we characterized hyperproperties with topological and algebraic approaches. With a better theoretical understanding of hyperproperties, we were able to tackle the problem of verification of hyperproperties, which is more complex than the standard case of trace properties verification. In fact, at the "hyper level", standard systems semantics are not useful, because they are defined at the sets of traces level. This led us to define new semantics, termed *hypersemantics*, computed at the same level of hyperproperties, namely at the sets of sets of executions level. In order to make the verification feasible, we still need approximation, namely simpler (i.e. approximate) versions of the systems (hyper)semantics. As already said, we perform approximation exploiting the framework of abstract interpretation. Once we set the concrete semantics, in our case hypersermantics able to verify hyperproperties, we need abstract domains, in order to obtain approximate, but decidable, verification methods. The verification is performed by means of static analysis, using a decidable abstract interpretation of the hypersemantics. Clearly, classic abstract domains cannot be used for hyperproperties, hence we define some design patterns useful to generate *hyperdomains*, namely abstract domains for hyperproperties verification.

Apart from these general results, we deepen the verification problem of a restriction of subset-closed hyperproperties, i.e. *bounded subset-closed* hyperproperties. These hyperproperties are expressive enough to capture lots of interesting specifications (such as information flows) but their verification is made easier. In particular, the verification of these hyperproperties is bounded to a fixed input cardinality, restricting the search space for confutation. Nevertheless, also for this kind of hyperproperties, the verification has to move to the hyperlevel. We propose a general technique for lifting the semantic operator computing the concrete systems semantics to the hyperlevel. The added value of tackling the problem from a general and formal point of view is that it allows us to discuss and prove soundness and completeness properties.

We instantiate our theoretical results to the verification of (Abstract) Non-Interference, allowing us to design a prototype verifier of Non-Interference for a toy programming language. In particular, we first design the hyperdomain denoting the hyperproperty that has to be satisfied by the semantics of non-interfering programs, i.e. the values of public variables must be always the same single value. Once we fix the domain of the abstract observable hyperproperty we define the abstract hypersemantics computing the executions

of a program on the hyperdomain, exactly as it happens in classic static analysis, but with the only difference that we are abstracting an hypersemantics into an hyperdomain. To the best of our knowledge, this is the first attempt to provide a sound analyzer/verifier of Non-Interference exploiting the expressiveness of hyperproperties. Finally, in order to test the feasibility of the proposed approach we have implemented a prototype (called nonInterfer) of the designed verifier. This prototype exhibits a good trade-off between verification speed and precision.

1.1 Structure of the Thesis and Contributions

Apart from this chapter, Chapter 2 introduces the basic mathematical concepts needed to understand the work and the notations adopted in the rest of the thesis. At the end of the chapter we can find a general introduction to abstract interpretation, which is the main framework we will use. The central part of the thesis comprises chapters from 3 to 8.

In Chapter 3 we introduce the problem of systems correctness which, basically, requires the definition of a (mathematical) model of the system and a (mathematical) description of a specification. Furthermore, concepts as verification, enforcement, systems analysis and approximations are described. Finally, we list some examples of common systems specifications and, in particular, we focus on information flows.

In Chapter 4 we introduce in detail the system model we have chosen, i.e. transition systems, and how to retrieve the behavior, i.e. the semantics, associated to a system. We do the same for specifications, namely we introduce hyperproperties. In Section 4.2.2 we investigate these latter from a theoretical point of view, giving a topological and algebraic characterization of hyperproperties. The topological characterization has been published in [Pasqua and Mastroeni, 2017]. At the end of Section 4.1.1 we describe a very simple programming language Imp, which will be used in the rest of the thesis. Indeed, starting from Chapter 5, the systems we take into account are programs written in Imp.

Chapter 5 aims to make the reader familiar with classic program verification of trace properties. We perform verification by means of analysis, formalized as an abstract interpretation of the concrete program semantics. In particular we build step by step a very simple numerical analysis for programs in Imp. This latter will be taken as a comparison for the case of hyperproperties verification.

Then we go at the hyperlevel. In Chapter 6.1 we deal with the problem of program verification of hyperproperties. Unfortunately, it is not easy to adapt classic static analysis for trace properties, to the more general hyperproperties. In this chapter we highlight the problems concerning hyperproperties verification and some possible solutions. Furthermore, we show how to define, in a constructive way, hypersemantics, i.e. program semantics computing at the level of sets of sets. Every abstract interpretation needs an abstract domain, in order to perform approximate verification. Hence, in Section 6.2, we take into account the problem of defining hyperdomains, namely abstract domains fro hyperproperties. Most of the results presented in this chapter have been published in [Mastroeni and Pasqua, 2017] and, partially, in [Mastroeni and Pasqua, 2018].

In Chapter 7 we focus on particular hyperproperties, the bounded subset-closed one, which exhibit a good trade-off between expressiveness and verification feasibility. Indeed, they could express, among other specifications, information flows and their verification can

be simplified significantly. In particular, we show how to adapt the existing classic static analyses, by lifting the semantic operator used to define the concrete semantics. Some parts of this chapter have been published in [Mastroeni and Pasqua, 2018].

Chapter 8 concludes the contributions of the thesis, giving an applicative example of our theoretical findings. In particular, exploiting the result of Chapter 7 we develop a (parametric) abstract semantics for Abstract Non-Interference and an abstract semantics for Non-Interference. These are formalizations of information flows specifications, hence we build a feasible and sound verification mechanism for information flows. The abstract semantics for Non-Interference has been implemented in a prototype verifier for Imp programs, called nonInterfer. The part of this chapter concerning Non-Interference has been published in [Mastroeni and Pasqua, 2019].

Finally, in Chapter 9 we discuss related works, the possible future research directions and the concluding remarks.

Publications. Most of the results presented in the thesis have been already published in:

[Mastroeni and Pasqua, 2017] Isabella Mastroeni and Michele Pasqua (2017). "Hyperhierarchy of Semantics - A Formal Framework for Hyperproperties Verification". In: *Static Analysis - 24th International Symposium, (SAS 2017), New York City, USA, August 30 – September 1, 2017, Proceedings*, pp. 232–252. DOI: 10.1007/978-3-319-66706-5_12. URL: https://doi.org/10.1007/978-3-319-66706-5_12

[Pasqua and Mastroeni, 2017] Michele Pasqua and Isabella Mastroeni (2017). "On topologies for (hyper)properties". In: *18th Italian Conference on Theoretical Computer Science (ICTCS 2017), Naples, Italy, September 26 – 28, 2017, Proceedings*, pp. 1–12. URL: http://ceur-ws.org/Vol-1949/ICTCSpaper13.pdf

[Mastroeni and Pasqua, 2018] Isabella Mastroeni and Michele Pasqua (2018). "Verifying Bounded Subset-Closed Hyperproperties". In: *Static Analysis*. Ed. by Andreas Podelski. Cham: Springer International Publishing, pp. 263–283. ISBN: 978-3-319-99725-4

[Mastroeni and Pasqua, 2019] Isabella Mastroeni and Michele Pasqua (2019). "Statically Analyzing Information Flows – An Abstract Interpretation-based Hyperanalysis for Non-Interference". In: *Proceedings of the 34rd Annual ACM Symposium on Applied Computing*. SAC '19. Limassol, Cyprus: ACM, pp. 2215–2223. ISBN: 978-1-4503-5933-7. DOI: 10.1145/3297280.3297498

2 Mathematical Background

This chapter introduces mathematical notations, basic notions and existing results that are at the foundation of our work and will serve in subsequent chapters. Then, in the last section, we provide an overview of abstract interpretation, which is the principal theory we use in our work.

2.1 Basic Notions and Notations

We recall now some basic notations of set theory. A set is a (not proper) class of distinct and unordered objects. When an object x is a member, or an element, of the set X we write $x \in X$, we write $x \notin X$ otherwise. We represent extensionally a (finite) set, the elements of which are $x_1, x_2, \ldots x_n$, with the notation $\{x_1, x_2, \ldots x_n\}$. Dually, we represent intensionally a (sub)set (of a set U), the elements of which fulfill a predicate ϕ, with the notation $\{x \in U \mid \phi(x)\}$. When U is clear from the context, we often omit it for brevity. The predicate ϕ is specified by a first-order formula possibly containing the classic logical operators: $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), $\Rightarrow$ (implication), $\Leftrightarrow$ (logical equivalence), $\forall$ (universal quantification), $\exists$ (existential quantification). The empty set is denoted by $\varnothing$ and the cardinality of a set X is denoted by $|X|$.

A set Y is a subset of a set X, written $Y \subseteq X$, if and only if every element of Y is in X. The empty set is a subset of every set. The set of all subsets of a set X, denoted $\wp(X)$, is defined as $\wp(X) \triangleq \{Y \mid Y \subseteq X\}$. Here the symbol $\triangleq$ stands for "is defined as", in contrast to the symbol $=$ which stands for "is the same as". The union of two sets X and Y, written $X \cup Y$, is the set of elements belonging to X or Y, namely: $X \cup Y \triangleq \{x \mid z \in X \vee z \in Y\}$. More generally, the union of a family of sets $\mathcal{X}$, denoted by $\bigcup \mathcal{X}$, is $\bigcup \mathcal{X} \triangleq \bigcup_{X \in \mathcal{X}} X = \{x \mid \exists X \in \mathcal{X} \,.\, x \in X\}$. Similarly, the intersection of two sets X and Y, written $X \cap Y$, is the set of elements belonging to X and Y, namely: $X \cap Y \triangleq \{x \mid z \in X \wedge z \in Y\}$. More generally, the intersection of a family of sets $\mathcal{X}$, denoted by $\bigcap \mathcal{X}$, is $\bigcap \mathcal{X} \triangleq \bigcap_{X \in \mathcal{X}} X = \{x \mid \forall X \in \mathcal{X} \,.\, x \in X\}$. The set-difference between set X and one of its subsets Y, denoted by $X \setminus Y$, is the set of all elements of X which are not elements of Y, namely $X \setminus Y \triangleq \{z \mid z \in X \wedge z \notin Y\}$. Given a set X, a set $\mathcal{P} \subseteq \wp(X)$ is a partition (of X) when $\mathcal{P} \neq \varnothing$, $\bigcup_{P \in \mathcal{P}} P = X$ and for every $P_1, P_2 \in \mathcal{P}$ either $P_1 = P_2$ or $P_1 \cap P_2 = \varnothing$.

The cartesian product of two sets X and Y, written $X \times Y$, is the set of all pairs where the first component is in X and the second component is in Y, namely $X \times Y \triangleq \{\langle x, y\rangle \mid x \in X \wedge y \in Y\}$. The cartesian product can be extended to n-tuples, with $n > 2$: $X_1 \times X_2 \times \ldots X_n \triangleq \{\langle x_1, x_2, \ldots x_n\rangle \mid \forall i \in \{1, 2, \ldots n\} \,.\, x_i \in X_i\}$. We write X^n in order to denoted the n-times cartesian product of X with itself.

Some sets are very important, hence they deserve a reserved symbol denoting them. The

set of natural numbers is denoted by $\mathbb{N}$ and the set of integer numbers is denoted by $\mathbb{Z}$. A numeric (integer) interval $[n, m]$ is the set of all integer numbers between n and m, called bounds, namely $[n, m] \triangleq \{x \in \mathbb{Z} \mid n \leq x \wedge x \leq m\}$. When the left bound is not in the set we use ']' in place of '[', i.e. $]n, m] \triangleq [n, m] \setminus \{n\}$ (and similarly for the right bound).

Finally, we will often use, in order to shorten the notation, the inline conditional construct $(\!|\, \texttt{cond} \,?\, \texttt{doTrue} : \texttt{doFalse} \,|\!)$, borrowed from programming languages like C and Java. The meaning is that the construct is replaced by the term `doTrue` if the condition `cond` is satisfied and by the term `doFalse` otherwise.

2.1.1 Relations and Functions

A *relation* $\mathcal{R}$ between sets $X_1, X_2, \ldots X_n$, for $n \in \mathbb{N}$, is a subset of the cartesian product $X_1 \times X_2 \times \ldots X_n$. The elements x_i, such that $x_i \in X_i$ for every $i \in [1, n]$, are in relation $\mathcal{R}$ when $\langle x_1, x_2, \ldots x_n \rangle \in \mathcal{R} \subseteq X_1 \times X_2 \times \ldots X_n$. A relation $\mathcal{R}$ is *binary* when $n = 2$, namely $\mathcal{R} \subseteq X_1 \times X_2$. We often write $x_1 \ \mathcal{R} \ x_2$ for $\langle x_1, x_2 \rangle \in \mathcal{R}$ and $x_1 \ \not\mathcal{R} \ x_2$ for $\langle x_1, x_2 \rangle \notin \mathcal{R}$. The composition of a relation $\mathcal{R}_1 \subseteq X \times Y$ with a relation $\mathcal{R}_2 \subseteq Y \times Z$ is the relation $\mathcal{R}_2 \circ \mathcal{R}_1 \subseteq X \times Z$ defined as $\mathcal{R}_2 \circ \mathcal{R}_1 \triangleq \{\langle x, z \rangle \mid \exists y \in Y \,.\, x \ \mathcal{R}_1 \ y \wedge y \ \mathcal{R}_2 \ z\}$.

A binary relation $\mathcal{R}$ on a set X is a subset of $X \times X$ and could have some important properties:

reflexivity $\forall x \in X \,.\, x \ \mathcal{R} \ x$

irreflexivity $\forall x \in X \,.\, x \ \not\mathcal{R} \ x$

symmetry $\forall x, y \in X \,.\, x \ \mathcal{R} \ y \Rightarrow y \ \mathcal{R} \ x$

antisymmetry $\forall x, y \in X \,.\, x \ \mathcal{R} \ y \wedge y \ \mathcal{R} \ x \Rightarrow x = y$

transitivity $\forall x, y, z \in X \,.\, x \ \mathcal{R} \ y \wedge y \ \mathcal{R} \ z \Rightarrow x \ \mathcal{R} \ z$

totality $\forall x, y \in X \,.\, x \ \mathcal{R} \ y \vee y \ \mathcal{R} \ x$

A binary relation is termed *equivalence* if it is reflexive, symmetric, and transitive. In a set with an equivalence relation, we consider, for each $x \in X$, the subset of X containing all the elements $y \in X$ such that $x\mathcal{R}y$. This set is called *equivalence class* of x (in X, through $\mathcal{R}$) and usually it is denoted by $[x]_{\mathcal{R}}$. An equivalence relation induces a partition of the set on which it is defined, and the elements of the partition are its equivalence classes.

A binary relation is a *preorder* if it is reflexive and transitive. A *partial order* is an antisymmetric preorder. A partial order which is total is called a *total order* (or linear order). Partial or a total orders are *strict* if they are irreflexive instead of reflexive. Note that, for strict orders, the antisymmetry requirement is useless: $x\mathcal{R}y$ and $y\mathcal{R}y$ cannot both hold since, by transitivity, this would imply $x\mathcal{R}x$, which cannot hold by irreflexivity.

A *function* f from the set X, called domain, to the set Y, called co-domain, is a relation $f \subseteq X \times Y$ such that:

- $\{x \in X \mid \langle x, y \rangle \in f\} = X$
- if $\langle x, y \rangle \in f$ and $\langle x, y' \rangle \in f$ then $y = y'$

This means that a function maps every element of X to one, and only one, element of Y. The element of y in relation with $x \in X$ is said its image and it is usually written $y = f(x)$. The set of all functions from X to Y is denoted by $X \to Y$. When we need to describe functions extensionally, we will use the following notation: $[x_1 \mapsto y_1\ x_2 \mapsto y_2\ \ldots\ x_n \mapsto y_n]$ denotes the function $\{\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \ldots \langle x_n, y_n\rangle\}$. We will often use the lambda notation to denote a function: $\lambda x\,.\,f(x)$. The composition of a function $f \in X \to Y$ with a function $g \in Y \to Z$ is a function $g \circ f \in X \to Z$ such that: $\forall x \in X\,.\,g \circ f(x) = g(f(x))$.

A function $f \in X \to Y$ could have some important properties:

injectivity $\forall x, x' \in X\,.\,f(x) = f(x') \Rightarrow x = x'$

surjectivity $\forall y \in Y \exists x \in X\,.\,f(x) = y$

An injective function is also called *one-to-one*, whilst a surjective function is also called *onto*. A *bijection* (or isomorphism) is an injective and surjective function. The inverse of a bijective function $f \in X \to Y$ is the bijective function $f^{-1} \triangleq \{\langle y, x\rangle \mid \langle x, y\rangle \in f\}$. The *direct image* (sometimes called additive lift) of $f \in X \to Y$ is the the function $\hat{f} \in \wp(X) \to \wp(Y)$ mapping a set $Z \subseteq X$ to the set of images of elements in Z, namely: $\hat{f}(Z) \triangleq \{f(z) \mid z \in Z\}$. Sometimes, we will abuse notation denoting with $f(Z)$ the direct image of f on Z.

A *partial function* f from the set X the set Y is a relation $f \subseteq X \times Y$ such that:

- if $\langle x, y\rangle \in f$ and $\langle x, y'\rangle \in f$ then $y = y'$

Technically, the term partial function is improper, since partial functions are not functions. Partial functions model the fact that for some elements of the domain the function is not defined. The set of all partial functions from X to Y is denoted by $X \hookrightarrow Y$. As for relations, we can define functions with more than one argument: $f \in X_1 \times X_2 \times \ldots X_n \to Y$. Usually, the notation $f(\langle x_1, x_2, \ldots x_n\rangle) = y$ is replaced with $f(x_1, x_2, \ldots x_n) = y$.

In the rest of the thesis, we will use very often the pointwise extension of relations and functions, as described in the following definition.

Definition 1 (Pointwise Extension)**.** Let X, Y, Z be arbitrary sets.

- The pointwise extension of a relation $\leq \subseteq Y \times Y$ is a relation $\dot{\leq}_X \subseteq (X \to Y) \times (X \to Y)$ defined as: $\dot{\leq}_X \triangleq \{\langle f, g\rangle \mid \forall x \in X\,.\,f(x) \leq g(x)\}$.
- The pointwise extension of a function $\alpha \in Y \to Z$ is a function $\dot{\alpha}_X \in (X \to Y) \to (X \to Z)$ defined as: $\dot{\alpha}_X \triangleq \lambda f\,.\,\lambda x\,.\,\alpha(f(x))$.
- The pointwise extension of an operator $\bigvee \in \wp(Y) \to Y$ is an operator $\dot{\bigvee}_X \in \wp(X \to Y) \to (X \to Y)$ defined as: $\dot{\bigvee}_X W \triangleq \lambda x\,.\,\bigvee\{f(x) \mid f \in W\}$.

Usually, the subscripts are omitted when X is clear from the context.

2.1.2 Ordered Structures

A set X with a partial order $\leq$, denoted $\langle X, \leq\rangle$, is called partially ordered set or, more concisely, POSET. If the relation $\leq$ is just a preorder, then $\langle X, \leq\rangle$ is called preordered set or, PROSET. Finite partially ordered sets can be drawn as line diagrams, called Hasse diagrams (Figure 2.1, on the left). We can draw also an infinite POSET by showing a finite part which

illustrates the building principle (Figure 2.1, on the right). In a Hasse diagram, each element $x \in X$ is uniquely represented by a node of the diagram, and there is an edge from a node x to a node y if y covers x, namely, $x \leq y$ and there exists no $z \in X$ such that $x \leq z \wedge z \leq y$. Hasse diagrams are usually drawn placing the elements higher than the elements they cover.

Let $\langle X, \leq \rangle$ be a POSET and let $Y \subseteq X$. We say that $u \in X$ is an *upper bound* of Y if $\forall y \in Y \,.\, y \leq u$; u is said *maximal* if it also belongs to Y. If the set of upper bounds (of Y) has the smallest element, then we call this latter *least upper bound* (or lub; join; supremum) of Y, denoted $\bigvee Y$ (or $\sup Y$). If the lub belongs to Y then it is said *maximum* (or top) and it is usually denoted by $\top$ (or $\max Y$). Dually, we say that $l \in X$ is a *lower bound* of Y if $\forall y \in Y \,.\, l \leq y$; l is said *minimal* if it also belongs to Y. If the set of lower bounds (of Y) has the biggest element, then we call this latter *greatest lower bound* (or glb; meet; infimum) of Y, denoted $\bigwedge Y$ (or $\inf Y$). If the glb belongs to Y then it is said *minimum* (or bottom) and it is usually denoted by $\bot$ (or $\min Y$). Note that maximum, minimum, supremum and infimum, when they exist, are unique. In general, we denote by $x \vee y$ and $x \wedge y$, the elements $\bigvee\{x, y\}$ and $\bigwedge\{x, y\}$, respectively.

A PROSET $\langle X, \leq \rangle$ is called *directed* if every pair of elements $x, y \in X$ has an upper bound in X. This is equivalent to say that every non-empty finite subset of X has an upper bound. Since a POSET is, in particular a PROSET, the definition of directed POSET is immediate. A (*directed*) *complete partial order* or, more concisely DCPO, is a partially ordered set in which every non-empty directed subset has a *least* upper bound. If a DCPO has a minimum, it is called *pointed*.

A subset of a partially ordered set is called a *chain*, when the partial order is total if restricted to the elements of the chain only. Note that every chain is a directed POSET. A partially ordered set is *chain-complete* when every chain, including the empty chain, has a least upper bound. The fact that the lub for the empty chain $\varnothing$ exists implies that the POSET has a minimum element $\bigvee \varnothing$.

An important result in order theory is that a partially ordered set is a pointed DCPO if and only if it is chain-complete, but it is provable only if we assume the axiom of choice (a proof can be found in [Markowsky, 1976]). Due to this equivalence, in the following we use the term *complete partial order* or, more concisely, CPO to refer a pointed DCPO or a chain-complete POSET. We denote a CPO with the tuple $\langle X, \leq, \vee, \bot \rangle$, where $\vee$ is the partial least upper bound operator (see Definition 2) and $\bot$ is the minimum.

A POSET $\langle X, \leq \rangle$ satisfies the *ascending chain condition* (ACC in short) if and only if any infinite increasing chain $x_0 \leq x_1 \leq \ldots$ of elements of X is not strictly increasing, namely: $\exists k \in \mathbb{N}$ such that $\forall j > k \,.\, x_k = x_j$. Dually, it satisfies the *descending chain condition* (DCC in short) if and only if any infinite decreasing chain $\ldots \leq x_1 \leq x_0$ of elements of X is not strictly decreasing, namely: $\exists k \in \mathbb{N}$ such that $\forall j > k \,.\, x_k = x_j$. Note that if a POSET has minimum and it is ACC, then it is a CPO.

A POSET $\langle X, \leq \rangle$ is a *join-semilattice* (or $\vee$-semilattice) if every pair of elements $x, y \in X$ has a least upper bound. This is equivalent to say that every non-empty finite subset of X has a least upper bound. A join-semilattice is said *bounded* if it has minimum, namely the least upper bound of $\varnothing$, and it is said *complete* if it has a least upper bound for every subset of X. Dually, it is a *meet-semilattice* (or $\wedge$-semilattice) if every pair of elements $x, y \in X$ has greatest lower bound. This is equivalent to say that every non-empty finite subset of X has greatest lower bound. A meet-semilattice is said to be bounded if it has maximum,

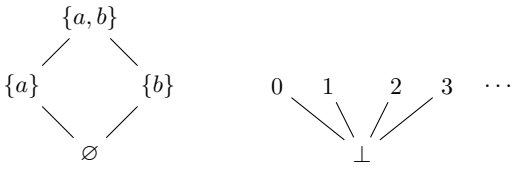


Figure 2.1: Hasse diagram examples.

namely the greatest lower bound of $\varnothing$, and it is said complete if it has a least upper bound for every subset of X. A *lattice* is a join-semilattice which is also a meet-semilattice. In a similar way, we obtain the notions of bounded and complete lattice. We denote a complete lattice with the tuple $\langle X, \leq, \vee, \wedge, \bot, \top \rangle$, where $\vee$ is the (total) least upper bound operator, $\wedge$ is the (total) greatest lower bound operator, $\bot$ is the minimum and $\top$ is the maximum.

So we can distinguish the ordered structures, depending on the subsets which have least upper bound (or, dually, on the subsets which have greatest lower bound). In order to be clear, we define the following operator.

Definition 2. Let $\langle P, \leq \rangle$ be a POSET, then a *partial least upper bound operator* $\vee \in \wp(P) \hookrightarrow P$ is a partial function such that, for every $\mathcal{X} \in \wp(X)$ if $\vee(\mathcal{X})$ is defined then the least upper bound of $\mathcal{X}$, i.e. $\bigvee \mathcal{X}$, exists and $\vee(\mathcal{X}) = \bigvee \mathcal{X}$.

So we can discriminate the various ordered structures introduced so far by looking when this operator is defined, as we can see in the following table.

	$\vee(\mathcal{X})$ is, at least, defined when
POSET	$\mathcal{X}$ is a singleton
POSET with minimum	$\mathcal{X}$ is a singleton or $\mathcal{X} = \varnothing$
DCPO	$\mathcal{X}$ is directed
pointed DCPO	$\mathcal{X}$ is directed or $\mathcal{X} = \varnothing$
chain-complete POSET	$\mathcal{X}$ is a, possibly empty, chain
join-semilattice	$\mathcal{X}$ is a non-empty finite set
bounded join-semilattice	$\mathcal{X}$ is a finite set
complete join-semilattice	$\mathcal{X}$ is a set

It is clear that the partial least upper bound operator is a total function if and only if it is defined over a complete lattice. We abuse notation denoting in the same way the least upper bound $\bigvee \mathcal{X}$ of a set $\mathcal{X}$ and the partial least upper bound operator $\vee(\mathcal{X})$ applied to $\mathcal{X}$.

Let $\langle X, \leq \rangle$ be a POSET with minimum $\bot$ and maximum $\top$. For each $x \in X$ we say that $y \in X$ is the *complement* of x if $x \wedge y = \bot$ and $x \vee y = \top$. A lattice where every element has a complement is called *complemented*. The complement of x is denoted as $\neg x$. A lattice such that for all $x, y, z \in X$ we have $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ and $x \vee (x \wedge z) = (x \vee y) \wedge (x \vee z)$ is called *distributive*. A *boolean algebra* or, more concisely BA, is a complemented and distributive lattice. If the lattice is complete then the boolean algebra is said complete as well. We denote a complete BA with the tuple $\langle X, \leq, \vee, \wedge, \bot, \top, \neg \rangle$, where $\vee$, $\wedge$, $\bot$, $\top$ have the same meaning as in the case of a complete lattice and $\neg$ is the (total) complement operator.

2.1.3 Ordinals

A set equipped with a total order is said *totally ordered* (or linearly ordered; simply ordered). Note that a chain is totally ordered. A binary relation $\mathcal{R}$ over a set X is *well-founded* when every non-empty subset of X has a least element with respect to $\mathcal{R}$. A well-founded total order is called a *well-order* (or well-ordering). A well-ordered set $\langle W, \leq \rangle$ is a set W equipped with a well-ordering $\leq$.

Every well-ordered set is associated with an *order type*. Two well-ordered sets $\langle W_1, \leq_2 \rangle$ and $\langle W_2, \leq_2 \rangle$ have the same order type if and only if they are *order-isomorphic*, that is, if there exists a bijective function $f \in W_1 \to W_2$ such that, for all elements $x, y \in W_1$ we have: $x \leq_1 y \Leftrightarrow f(x) \leq_2 f(y)$. An *ordinal* is the order type of a well-ordered set and it represents all well-ordered sets which are order-isomorphic to it. In fact, a well-ordered set $\langle W, \leq \rangle$, with order type β, is order-isomorphic to the well-ordered set of all ordinals strictly smaller than the ordinal β itself, namely it is order-isomorphic to the well-ordered set $\{w \in W \mid w < \beta\}$. Here $<$ is the strict version of the relation $\leq$. This property permits to define each ordinal as the well-ordered set of all ordinals that precede it, namely:

$$0 \triangleq \varnothing \qquad \text{is the smallest ordinal}$$
$$\beta + 1 \triangleq \beta \cup \{\beta\} \qquad \text{is a successor ordinal}$$

Thus, the first successor ordinal is $1 \triangleq \{0\} = \{\varnothing\}$. The next is $2 \triangleq \{0, 1\} = \{\varnothing, \{\varnothing\}\}$. Continuing in this manner, we obtain all natural numbers, that is, all *finite ordinals*. A limit ordinal is an ordinal number which is neither 0 nor a successor ordinal, namely λ is a limit ordinal when $\forall \beta < \lambda \,.\, \beta + 1 < \lambda$. The set $\mathbb{N}$ of all natural numbers, denoted as an ordinal by ω, is the first limit ordinal and the first *transfinite ordinal* (or infinite ordinal). Using ordinals we can use a technique, generalizing the classic induction on naturals, to prove or define properties of uncountable sets.

Definition 3 (Transfinite Induction)**.** A property ϕ holds for each ordinal β if:

base case $\phi(0)$ holds

successor case for each successor ordinal $\beta + 1$: $\phi(\beta)$ holds implies $\phi(\beta + 1)$ holds

limit case for each limit ordinal λ: $\forall \beta < \lambda$, $\phi(\beta)$ holds implies $\phi(\lambda)$ holds

The classic induction principle works only for denumerable sets and it coincides with the transfinite induction of Definition 3 where we do not have the limit case and where the successor ordinals we take into account are only those strictly smaller that ω.

2.1.4 Functions on Ordered Structures

Let $\langle D, \leq \rangle$ and $\langle D^\sharp, \leq^\sharp \rangle$ be two POSET. A function $f \in D \to D^\sharp$ is said to be *monotonic* (or monotone; order preserving) when, for each $x, y \in D$, $x \leq y \Rightarrow f(x) \leq^\sharp f(y)$. It is *Scott-continuous* when it preserves existing least upper bounds of directed subsets, that is, for directed sets $Z \subseteq D$, if $\bigvee Z$ exists then $\bigvee^\sharp \{f(x) \mid x \in Z\}$ exists and $f(\bigvee Z) = \bigvee^\sharp \{f(x) \mid x \in Z\}$. The definition can be equivalently given using (lub of) non-empty chains in place of directed subsets. Dually, it is Scott-co-continuous when it preserves existing greatest lower bounds of

co-directed[1] subsets, that is, if $\bigwedge Z$ exists, for a co-directed set $Z \subseteq D$, then $\bigwedge^\sharp\{f(x) \mid x \in Z\}$ exists and $f(\bigwedge Z) = \bigwedge^\sharp\{f(x) \mid x \in Z\}$. A function f is said (completely) *additive* (or complete join-morphism) if it preserves existing least upper bounds of arbitrary non-empty sets. Dually, f is said (completely) *multiplicative* (or complete meet-morphism; co-additive) if it preserves existing greatest lower bounds of arbitrary non-empty sets. Finally, f is said *strict* if it preserves bottom elements, namely, whenever bottom elements $\bot \in D$ and $\bot^\sharp \in D^\sharp$ exist, then $f(\bot) = \bot^\sharp$.

Definition 4 (Iteratable Function)**.** Given a POSET $\langle X, \leq\rangle$, with partial least upper bound operator $\vee \in \wp(X) \hookrightarrow X$, we say that a monotone function $f \in X \to X$ is $\vee$*-iteratable* on $x \in X$ when the transfinite iterates of f from x, defined as,

$$f^\lambda \triangleq \begin{cases} x & \text{if } \lambda = 0 \\ f(f^\beta) & \text{if } \lambda = \beta + 1 \\ \bigvee_{\beta<\lambda} f^\beta & \text{if } \lambda \text{ is a limit ordinal} \end{cases}$$

are well-defined.

2.1.5 Fixpoints

Given a partially ordered set $\langle X, \leq\rangle$ and a function $f \in X \to X$, a *fixpoint* of f is an element $x \in X$ such that $f(x) = x$. An element $x \in X$ such that $x \leq f(x)$ is called a *pre-fixpoint* while, dually, a *post-fixpoint* is an element $x \in X$ such that $f(x) \leq x$. The $\leq$-least fixpoint of f, written $\mathsf{lfp}^\leq f$, is a fixpoint of f such that, for every fixpoint $x \in X$ of f, $\mathsf{lfp}^\leq f \leq x$. We write $\mathsf{lfp}^\leq_x f$ for the $\leq$-least fixpoint of f which is greater than, or equal to, an element $x \in P$. Dually, we define the $\leq$-greatest fixpoint of f, denoted by $\mathsf{gfp}^\leq f$, and the $\leq$-greatest fixpoint of f smaller than, or equal to, $x \in X$, denoted by $\mathsf{gfp}^\leq_x f$.

It is interesting, and useful, to know whether a function admits fixpoints, in particular the least and of the greatest one. For this reason, we recall some important fixpoint theorems.

Theorem 1 (Knaster-Tarski Fixpoint Theorem)**.** *Let $\langle L, \leq, \vee, \wedge, \bot, \top\rangle$ be a complete lattice and $f \in L \to L$ monotonic. The set of fixpoints of f is a complete lattice.*

As a corollary of the theorem, we have that f has a $\leq$-least fixpoint $\mathsf{lfp}^\leq f = \bigwedge\{x \in P \mid f(x) \leq x\}$ and a $\leq$-greatest fixpoint $\mathsf{gfp}^\leq f = \bigvee\{x \in P \mid x \leq f(x)\}$. These characterizations guarantee that fixpoints exist but do not gives us any hint on how we can compute such fixpoints. Another fixpoints characterization, which is constructive, is the following.

Theorem 2 (Kleene Fixpoint Theorem)**.** *Let $\langle D, \leq, \vee, \bot\rangle$ be a CPO and $f \in D \to D$ Scott-continuous. Then, f has a $\leq$-least fixpoint which is the least upper bound of the increasing chain*

$$\bot \leq f(\bot) \leq f^2(\bot) \leq \dots$$

namely: $\mathsf{lfp}^\leq_\bot f = \bigvee_{n<\omega} f^n(\bot) = \bigvee\{f^n(\bot) \mid n \in \mathbb{N}\}$.

[1] A POSET $\langle X, \leq\rangle$ is called co-directed if every pair of elements $x, y \in X$ has a lower bound in X.

This ensures that the fixpoint is reached in, at most, a denumerable number of steps. Note that the number of steps is finite when the the CPO satisfies the ascending chain condition. A dual result holds for the greatest fixpoint case.

Having a continuous function is a strong assumption, hence we show now another fixpoints constructive characterization, which relaxes the continuity condition, but requires a possibly transfinite iteration.

Theorem 3 (Cousot Fixpoint Theorem)**.** *Let $\langle D, \leq \rangle$ be a DCPO (not necessarily pointed), $f \in D \to D$ monotonic and $x \in D$ a pre-fixpoint. Then, the iteration sequence:*

$$f^{\lambda} \triangleq \begin{cases} x & \text{if } \lambda = 0 \\ f(f^{\beta}) & \text{if } \lambda = \beta + 1 \\ \bigvee_{\beta < \lambda} f^{\beta} & \text{if } \lambda \text{ is a limit ordinal} \end{cases}$$

converges towards the $\leq$-least fixpoint $\mathsf{lfp}_x^{\leq} f$.

The theorem basically says that f is $\vee$-iteratable on x, with respect to the partial least upper bound operator $\vee \in \wp(D) \hookrightarrow D$, and its least fixpoint greater than x is given by the transfinite iterates of f from x.

2.1.6 Galois Insertions, Closure Operators and Moore Families

Given a POSET $\langle X, \leq \rangle$, a monotone function $\rho \in X \to X$ is an *upper closure operator* (or uco) if it satisfies the following conditions:

extensivity $\forall x \in X \,.\, x \leq \rho(x)$

idempotence $\forall x \in X \,.\, \rho(\rho(x)) = \rho(x)$

Vice versa, if ρ is reductive, namely $\forall x \in X \,.\, \rho(x) \leq x$, and idempotent then it is called lower closure operator (or lco). An upper closure operator is fully identifiable with the sets of its fixpoints, denoted as $\rho(X) = \{x \in X \mid \rho(x) = x\}$. We denote the set of all closure operators of X as $\mathsf{uco}(X)$.

Let $\langle D, \leq \rangle$ and $\langle D^{\sharp}, \leq^{\sharp} \rangle$ be two POSETs. Consider two monotone functions $\alpha \in D \to D^{\sharp}$ and $\gamma \in D^{\sharp} \to D$. We say that $(D, \alpha, \gamma, D^{\sharp})$ is a *Galois connection*, shortly GC, when

$$\forall x \in D \, \forall x^{\sharp} \in D^{\sharp} \,.\, x \leq \gamma(x^{\sharp}) \Leftrightarrow \alpha(x) \leq^{\sharp} x^{\sharp}$$

In this case we write:

$$\langle D, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle D^{\sharp}, \leq^{\sharp} \rangle$$

Here, α and γ are said to be *adjoint functions*[2], where α is the left (or upper) adjoint and γ is the right (or lower) adjoint. An important feature of adjoint functions is that each adjoint can be uniquely determined by using the other one in the following way:

$$\alpha = \gamma^{+} \triangleq \lambda x \,.\, \bigwedge \{x^{\sharp} \in D^{\sharp} \mid x \leq \gamma(x^{\sharp})\}$$
$$\gamma = \alpha^{-} \triangleq \lambda x^{\sharp} \,.\, \bigvee \{x \in D \mid \alpha(x) \leq^{\sharp} x^{\sharp}\}$$

[2]The term adjoint is borrowed from category theory, indeed α and γ are basically adjoint (covariant) functors between the categories induced by the underlining POSETs.

This is tantamount to say that α is additive, i.e. $\alpha(\bigvee Y) = \bigvee^\sharp\{\alpha(x) \mid x \in Y\}$, and γ is co-additive, i.e. $\gamma(\bigwedge Y) = \bigwedge^\sharp\{\gamma(x^\sharp) \mid x^\sharp \in Y\}$. Galois connections have also other nice algebraic properties:

- $\gamma \circ \alpha$ is extensive and idempotent
- $\alpha \circ \gamma$ is reductive and idempotent
- $\gamma \circ \alpha \circ \gamma = \gamma$ and $\alpha \circ \gamma \circ \alpha = \alpha$

When $\alpha \circ \gamma$ is the identity function $\mathsf{id} = \lambda x^\sharp \,.\, x^\sharp$ on $D^\sharp$, we say that $(D, \alpha, \gamma, D^\sharp)$ is a *Galois insertion*, shortly GI. In this case we write:

$$\langle D, \leq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle D^\sharp, \leq^\sharp\rangle$$

In a Galois insertion we have that α is surjective and γ is injective. We can always obtain a Galois insertion starting from a Galois connection. This process is called *reduction* and it consists in collecting together all the elements $x^\sharp \in D^\sharp$ having the same image under γ.

When we also have that $\gamma \circ \alpha$ is the identity function $\mathsf{id} = \lambda x \,.\, x$ on D, we say that $(D, \alpha, \gamma, D^\sharp)$ is a *Galois isomorphism*. In this case we write:

$$\langle D, \leq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle D^\sharp, \leq^\sharp\rangle$$

It is called Galois ismorphism because α and γ are bijective functions.

Let $\langle X, \leq\rangle$ be a POSET with maximum $\top$. Then $Y \subseteq X$ is a *Moore family* of X when for every $Z \subseteq Y$ we have that $\bigwedge Z$ exists and, in particular, $\bigwedge Z \in Y$. This implies that $\bigwedge \varnothing = \top \in Y$. A Moore family is hence closed under meet. The Moore closure of a set $Y \subseteq X$ is defined as $\mathcal{M}(Y) \triangleq \{\bigwedge Z \mid Z \subseteq Y\}$, namely $\mathcal{M}(Y)$ is the smallest (w.r.t. set inclusion) subset of X which contains Y and which is a Moore family of X.

An important fact is that there is a strong link between upper closure operators, Galois insertion and Moore families. Indeed, for every Galois insertion $(D, \alpha, \gamma, D^\sharp)$, we have that $\gamma \circ \alpha$ is an upper closure operator of D. Dually, given a $\rho \in \mathsf{uco}(D)$ and an isomorphism $\iota \in \rho(D) \to D^\sharp$ we have that $(D, \iota \circ \rho, \iota^{-1}, D^\sharp)$ is a Galois insertion. Note that if we chose $D^\sharp = \rho(D)$, then the isomorphism ι is just the identity function $\mathsf{id} = \lambda x \,.\, x$ and the Galois insertion is $(D, \rho, \mathsf{id}, \rho(D))$. The set of fixpoints of an upper closure operator in $\mathsf{uco}(X)$ is a Moore family of X. Hence, we have that $D^\sharp$ in a Galois insertion $(D, \alpha, \gamma, D^\sharp)$ is isomorphic to a Moore family of D.

Finally, note that Moore families, and hence the set of fixpoints of upper closure operators, are closed under meet but not necessarily under join.

2.2 Abstract Interpretation

The prominent application of abstract interpretation is static analysis, where it is used for the definition of abstract semantics of programs. In this setting, the concrete semantic of a program, namely its "meaning", is a representation of all its possible behaviors by means of a mathematical object (usually a set). This latter is, in general, not computable: it is not possible to represent and to compute all possible behaviors of any program in all its possible execution environments. Clearly all non trivial properties of the concrete semantics of

a program are undecidable: it is not possible to answer any question about the behaviors of every program. For this reason *abstract interpretation* is born, as a theory for the sound approximation of the semantics of discrete dynamic systems. The approximation consists in the observation of the semantics at a specified level of abstraction, focusing only on the important aspects of the computation. In this setting abstract interpretation permits to calculate a (computable) abstract semantics of the system, parametrized on the properties of interest. The approximation is sound by design, in the sense that what holds in the abstract holds also in the concrete (no false negatives).

Remark. The choice of the concrete semantics, and consequently the abstract semantics, is relative to the goal of the analysis. Indeed, it is possible that an abstract semantics could be the concrete semantics w.r.t. another one. Abstract interpretation is a theory for relating a concrete and an abstract semantics, hence the focus is in this *relation*, not in the semantics themselves.

Abstract interpretation is not limited to static analysis. Indeed, it is a theoretical framework to reason about computing systems but, at the same time, it offers a constructive methodology to build formal algorithms that manipulate them. So it is not surprising that abstract interpretation is applied in a lot of fields of computer science: static analysis [Cousot and Cousot, 1977], type inference [Cousot, 1997], model checking [Clarke, Grumberg, and Long, 1994], comparative semantics [Comini, Levi, and Meo, 2001], program transformation [Cousot and Cousot, 2002], automatic deduction [Plaisted, 1981], database queries optimization [Helm, Marriott, and Odersky, 1995], cryptography [Adi and Debbabi, 2003], quantum computing [Perdrix, 2008], etc. In the following we introduce the basic concepts of abstract interpretation, in a general setting.

2.2.1 Concrete and Abstract Objects

In the most general sense, abstract interpretation is a theory for the approximation of mathematical objects, whatever these objects refer to. Indeed, abstract interpretation is not focused on the meaning of these objects but in the relation between concrete and abstract, i.e. approximated, elements. The goal of abstract interpretation is to give the formal means for relating mathematical objects at different levels of abstraction, potentially transferring the computations from the concrete level to the abstract one.

The concrete objects domain $\mathcal{O}$ describes the elements of interest. For instance, sets of numbers, functions, collecting semantics of programs, etc. The goal of abstract interpretation is to approximate concrete objects with abstract ones, chosen in an abstract objects domain $\mathcal{O}^\sharp$. In other words, the intention of an abstract interpretation is to find an abstract object A in the abstract domain $\mathcal{O}^\sharp$ which is a correct approximation of the concrete object $C \in \mathcal{O}$. Concrete and abstract objects came with two relations stating the relative precision between elements: more precise elements (concrete or abstract) carry more information. This means that $\langle \mathcal{O}, \preccurlyeq \rangle$ and $\langle \mathcal{O}^\sharp, \preccurlyeq^\sharp \rangle$ are POSET and $C \in \mathcal{O}$ is more precise than $C' \in \mathcal{O}$ if and only if $C \preccurlyeq C'$. Analogously, $A \in \mathcal{O}^\sharp$ is more precise than $A' \in \mathcal{O}^\sharp$ if and only if $A \preccurlyeq^\sharp A'$. Usually, $\preccurlyeq$ is called approximation order and $\preccurlyeq^\sharp$ is called abstract approximation order. Correctness, more often said *soundness*, is given by a correctness relation, stating which are the correct approximations of a concrete object. This relation can be defined in a lot of ways, depending on the algebraic properties that concrete and abstract domains have (see [Cousot and Cousot, 1992] for a detailed explanation). In the following, we assume that the sound-

ness relation is given in terms of either a monotonic abstraction function $\alpha \in \mathcal{O} \to \mathcal{O}^\sharp$ or a monotonic concretization function $\gamma \in \mathcal{O}^\sharp \to \mathcal{O}$. This means that $\alpha(C) \in \mathcal{O}^\sharp$ is a correct approximation of $C \in \mathcal{O}$, or similarly, that $A \in \mathcal{O}^\sharp$ is a correct approximation of $\gamma(A) \in \mathcal{O}$. The domains are called *approximation domains*, since they are useful for comparing objects, w.r.t. their precision.

The monotonicity of the abstraction, or the concretization, function preserves the relative precision of objects, namely $C \preccurlyeq C'$ implies $\alpha(C) \preccurlyeq^\sharp \alpha(C')$, or $A \preccurlyeq^\sharp A'$ implies $\gamma(A) \preccurlyeq \gamma(A')$. Monotonicity is not sufficient to obtain best abstractions. For instance, even if α maps a concrete object to a correct abstract object, it is not guaranteed that this latter is the most precise. We have the best abstraction when the abstraction α is a complete join-morphism or, equivalently, the concretization γ is a complete meet-morphism (assuming that arbitrary join and meet exist). This tantamount to say that α and γ are adjoint functions and hence form a Galois connection. We will deal with Galois connection-based abstract interpretation in a moment.

2.2.1.1 Functions Abstraction

Usually, concrete objects are computed by means of a function on the concrete domain. Hence, once we have abstracted objects, the natural second step is to abstract computations. Indeed, abstract interpretation is motivated by the fact that a concrete function $F \in \mathcal{O} \to \mathcal{O}$ is not computable (or too expensive in terms of complexity). Hence we seek an abstract function $F^\sharp \in \mathcal{O}^\sharp \to \mathcal{O}^\sharp$ which correctly approximates F, and that is computable. As said, the abstract function must be sound, namely:

$$\forall C \in \mathcal{O} \,.\, \alpha(F(C)) \preccurlyeq^\sharp F^\sharp(\alpha(C)) \quad \text{or} \quad \forall A \in \mathcal{O}^\sharp \,.\, F(\gamma(A)) \preccurlyeq \gamma(F^\sharp(A)) \tag{2.1}$$

This means that computing in the abstract always yields less or as much information as computing in the concrete. When the equality is required, namely when we require that the abstract computation does not lose any information w.r.t. the concrete one, we say that $F^\sharp$ is exact, or more commonly *complete*. This happens when concrete and abstract functions commute, i.e. when $\alpha \circ F = F^\sharp \circ \alpha$ or $F \circ \gamma = \gamma \circ F^\sharp$.

2.2.1.2 The Optimal Case of Galois Connections

When the abstraction function preserves existing least upper bounds, i.e. it is additive, there is a unique concretization γ expressing the same soundness relation between concrete and abstract elements: $\gamma \triangleq \alpha^- = \lambda A \,.\, \curlyvee\{C \in \mathcal{O} \mid \alpha(C) \preccurlyeq^\sharp A\}$. Dually, when the concretization function preserves existing greatest lower bounds, i.e. it is co-additive, there is a unique abstraction α expressing the same soundness relation between concrete and abstract elements: $\alpha \triangleq \gamma^+ = \lambda C \,.\, \curlywedge^\sharp\{A \in \mathcal{O}^\sharp \mid C \preccurlyeq \gamma(A)\}$. In this case we have that abstraction and concretization are adjoint functions, hence they form a Galois connection:

$$\langle \mathcal{O}, \preccurlyeq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{O}^\sharp, \preccurlyeq^\sharp \rangle$$

In a Galois connection setting, soundness can be checked, equivalently, in the concrete or in the abstract, in fact it holds:

$$\forall C \in \mathcal{O}, A \in \mathcal{O}^\sharp \,.\, \alpha(C) \preccurlyeq^\sharp A \Leftrightarrow C \preccurlyeq \gamma(A) \tag{2.2}$$

In particular we have that $\alpha(C)$ is the best possible abstract element approximating C, namely it is the most precise, w.r.t. $\preccurlyeq^\sharp$, correct approximation of C in $\mathcal{O}^\sharp$.

One of the fundamental aspects of Galois connection-based abstract interpretations is that the majority of the properties of the approximation process are specified only by the (abstract) domain of mathematical objects chosen for representing the objects of interest. A theory of domains for abstract interpretation was defined in [Cousot and Cousot, 1977; Cousot and Cousot, 1979b] based on the notion of Galois connection.

In a Galois connection setting we can exploit nice algebraic properties. For instance, Equation 2.2 implies that the two definitions of soundness in Equation 2.1 are equivalent. Furthermore, given a concrete function F it is always possible to retrieve a sound approximation, which is also the most precise [Cousot and Cousot, 1979b]. This latter is called *best correct approximation* and it is defined as $F^{\text{bca}} \triangleq \alpha \circ F \circ \gamma$. As a consequence, in order to prove soundness, it is sufficient to prove that $F^\sharp$ approximates F^{bca}, namely prove that $F^{\text{bca}} \mathrel{\dot{\preccurlyeq}^\sharp} F^\sharp$.

Unfortunately, when we need exact approximations things are more complicated. First, even in a Galois connection setting, an exact abstract function is not guaranteed to exist. Second, the two notions of completeness are not equivalent. Indeed we have:

- *backward* completeness, when $F \circ \gamma = \gamma \circ F^\sharp$
- *forward* completeness, when $\alpha \circ F = F^\sharp \circ \alpha$

We have this two different notions of completeness, depending on where we compare the concrete and the abstract computations. If we compare the results in the abstract domain, we obtain backward completeness while, if we compare the results in the concrete domain, we obtain forward completeness. An important result is that a function admits a backward, or forward, complete abstraction if and only if its best correct approximation is backward, or forward, complete [Giacobazzi, Ranzato, and Scozzari, 2000].

Lattice of Abstract Interpretations. A Galois insertion is a Galois connection where $\alpha \circ \gamma = \lambda A \,.\, A$ (i.e. it is the identity function on $\mathcal{O}^\sharp$). It is not restrictive to reason with insertions instead of connections, since any Galois connection can be reduced to a Galois insertion, eliminating the redundant elements. Often it is convenient to consider domains independently from the representation of their objects. In this case they are specified by means of upper closure operators. As we have already seen, an upper closure operator is completely described by the set of its fixpoints $\rho(\mathcal{O}) = \{C \in \mathcal{O} \mid C = \rho(C)\}$. Every Galois insertion $(\mathcal{O}, \alpha, \gamma, \mathcal{O}^\sharp)$, and so every abstract domain of $\mathcal{O}$, is uniquely identifiable with an upper closure operator $\gamma \circ \alpha$ on $\mathcal{O}$. The converse also holds, namely every upper closure operator ρ induces a Galois insertion $(\mathcal{O}, \rho, \mathsf{id}, \rho(\mathcal{O}))$. So there is a one-to-one correspondence between upper closure operators and abstract domains defined by Galois insertions. If $\langle \mathcal{O}, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$ is a complete lattice then $\langle \mathsf{uco}(\mathcal{O}), \dot{\sqsubseteq}, \dot{\sqcup}, \dot{\sqcap}, \lambda C \,.\, C, \lambda C \,.\, \top \rangle$, where $\mathsf{uco}(\mathcal{O})$ denotes the set of all upper closure operators on $\mathcal{O}$, is the *lattice of abstract interpretations* of $\mathcal{O}$. This is the complete lattice of all possible abstract domains of $\mathcal{O}$. The partial order $\dot{\sqsubseteq}$ is used to compare domains: ρ_1 is more precise than ρ_2 if and only if $\rho_1 \mathrel{\dot{\sqsubseteq}} \rho_2$, namely if and only if $\rho_2(\mathcal{O}) \subseteq \rho_1(\mathcal{O})$.

2.2.2 Fixpoint Computations

Very often, the objects of interest are the result of a fixpoint computation, namely they are (usually the least or the greatest) fixpoints of some functions. For instance, this is the case for many formulations of programs semantics. Not all functions admit fixpoints: some assumptions are needed in order to apply one of the fixpoint theorems presented in Subsection 2.1.5. In the rest of the thesis we assume to deal with objects computable by means of functions which admit a fixpoint. Indeed, we only deal with objects which are constructive, as explained in the following definition.

A mathematical object is said to be *constructive*, i.e. expressible in fixpoint form, if there exists a *computational fixpoint definition*[3], computing it.

Definition 5 (Computational Fixpoint Definition)**.** $\langle \mathcal{F}, \mathbb{D}, \pm \rangle$ is a *computational fixpoint definition* when $\mathbb{D} = \langle \mathcal{D}, \leq, \vee \rangle$ is a POSET with partial least upper bound operator $\vee \in \wp(\mathcal{D}) \hookrightarrow \mathcal{D}$, $\mathcal{F} \in \mathcal{D} \to \mathcal{D}$ is $\leq$-monotone, $\pm$ is a pre-fixpoint of $\mathcal{F}$ and $\mathcal{F}$ is $\vee$-iteratable on $\pm$. The object computed is $\mathsf{lfp}^{\leq}_{\pm}\, \mathcal{F} \in \mathcal{D}$.

Remark. $\mathsf{lfp}^{\leq}_{\pm}\, \mathcal{F}$ always exists, applying Theorem 3. We can use this latter even if $\mathbb{D}$ is not a CPO, since $\mathcal{F}$ is supposed to be $\vee$-iteratable and hence the least upper bound of its iterates does exist.

Then an object D is constructive when there exists a computational fixpoint definition $\langle \mathcal{F}, \mathbb{D}, \pm \rangle$, with $\mathbb{D} = \langle \mathcal{D}, \leq, \vee \rangle$, such that $D = \mathsf{lfp}^{\leq}_{\pm}\, \mathcal{F}$. By definition, $\mathsf{lfp}^{\leq}_{\pm}\, \mathcal{F}$ is the limit of the increasing iterates of $\mathcal{F}$ starting from $\pm$, namely $\mathsf{lfp}^{\leq}_{\pm}\, \mathcal{F} = \mathcal{F}^{\lambda}$, for a limit ordinal λ (see Definition 4).

Remark. Usually, $\mathbb{D}$ is at least a CPO (in this case monotonicity implies that $\mathcal{F}$ is iteratable), nevertheless, the least upper bound needs to be defined for the iterates of $\mathcal{F}$, not for every directed subset of $\mathcal{D}$. If $\mathbb{D}$ is a CPO, the pre-fixpoint is usually chosen to be the minimum.

In abstract interpretation, we assume that a concrete object $C \in \mathcal{O}$ is computed by means of a computational fixpoint definition $\langle F, \mathbb{O}, \bot \rangle$, with $\mathbb{O} = \langle \mathcal{O}, \sqsubseteq, \sqcup \rangle$. In particular C is the $\sqsubseteq$-least fixpoint of F greater than $\bot$ and it is computed as $C = \bigsqcup_{n<\lambda} F^n(\bot)$. Similarly, we assume that an abstract object $A \in \mathcal{O}^\sharp$ is computed by means of a computational fixpoint definition $\langle F^\sharp, \mathbb{O}^\sharp, \bot^\sharp \rangle$, with $\mathbb{O}^\sharp = \langle \mathcal{O}^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp \rangle$. In particular A is the $\sqsubseteq^\sharp$-least fixpoint of $F^\sharp$ greater than $\bot^\sharp$ and it is computed as $A = \bigsqcup^\sharp_{n<\delta} F^{\sharp n}(\bot^\sharp)$. These domains are called *computational domains* since they are useful for computing objects. Usually, $\sqsubseteq$ is called computational order and $\sqsubseteq^\sharp$ is called abstract computational order. These latter may, or may not, be equal to the approximation orders. In the following, for simplicity of exposure, we assume that computational and approximation orders coincide, hence we let $\preccurlyeq$ be equal to $\sqsubseteq$ and $\preccurlyeq^\sharp$ be equal to $\sqsubseteq^\sharp$. We wanted to explicit this difference here because, in some cases, as we will see in some chapters, the domain of computation is different from the one in which we compare objects w.r.t. precision.

The abstract interpretation framework allows us to systematically derive the abstract operator $F^\sharp$, computing A, starting from the definition of the concrete operator F, computing C, such that A is correct approximation of C. We can do the same also when we have computations involving fixpoints exploiting approximation and transfer theorems. In the following theorems let $\langle F, \mathbb{O}, \bot \rangle$, with $\mathbb{O} = \langle \mathcal{O}, \sqsubseteq, \sqcup \rangle$, and $\langle F^\sharp, \mathbb{O}^\sharp, \bot^\sharp \rangle$, with $\mathbb{O}^\sharp = \langle \mathcal{O}^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp \rangle$, be concrete and abstract computational fixpoint definitions.

[3] The following definition is taken from the *fixpoint semantics specification* of [Cousot, 2002].

Theorem 4 (Kleenian Fixpoint Approximation)**.** *Assume that the strict Scott-continuous function $\alpha \in \mathcal{O} \to \mathcal{O}^\sharp$ is such that for every pre-fixpoint $C \in \mathcal{O}$ there exists $C' \sqsubseteq C$ such that $\alpha(F(C)) \sqsubseteq^\sharp F^\sharp(\alpha(C'))$. Then $\alpha(\mathsf{lfp}^{\sqsubseteq}_{\bot} F) \sqsubseteq^\sharp \mathsf{lfp}^{\sqsubseteq^\sharp}_{\bot^\sharp} F^\sharp$.*

Theorem 5 (Tarskian Fixpoint Approximation)**.** *Suppose that $\langle \mathcal{O}, \sqsubseteq, \sqcup, \sqcap, \bot, \top\rangle$ and $\langle \mathcal{O}^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp\rangle$ are complete lattices. Assume that the monotone function $\alpha \in \mathcal{O} \to \mathcal{O}^\sharp$ is such that for every post-fixpoint $A^\sharp \in \mathcal{O}^\sharp$ there exists a post-fixpoint $C \in \mathcal{O}$ such that $\alpha(C) \sqsubseteq^\sharp A^\sharp$. Then $\alpha(\mathsf{lfp}^{\sqsubseteq}_{\bot} F) \sqsubseteq^\sharp \mathsf{lfp}^{\sqsubseteq^\sharp}_{\bot^\sharp} F^\sharp$.*

The first theorem relies on the fact that, each abstract iteration step $F^{\sharp n}(\bot^\sharp)$ is a sound approximation of the corresponding concrete iteration step $F^n(\bot)$. Passing to the limit, we obtain that the limit of the abstract iterations chain is a sound approximation of the concrete iteration chain. The second theorem, instead, exploits the fact that $\mathsf{lfp}^{\sqsubseteq}_{\bot} F$ is the smallest post-fixpoint and the fact that a post-fixpoint of a sound abstract function is a sound approximation of a post-fixpoint of the concrete function.

When we require exactness, in this cases backward completeness, we can use the following transfer theorems.

Theorem 6 (Kleenian Fixpoint Tranfser)**.** *Assume that the strict Scott-continuous function $\alpha \in \mathcal{O} \to \mathcal{O}^\sharp$ satisfies the* commutation condition *$F^\sharp \circ \alpha = \alpha \circ F$. Then we have $\alpha(\mathsf{lfp}^{\sqsubseteq}_{\bot} F) = \mathsf{lfp}^{\sqsubseteq^\sharp}_{\bot^\sharp} F^\sharp$.*

Theorem 7 (Tarskian Fixpoint Transfer)**.** *Assume $\langle \mathcal{O}, \sqsubseteq, \sqcup, \sqcap, \bot, \top\rangle$ and $\langle \mathcal{O}^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp\rangle$ complete lattices. Assume that the co-additive function $\alpha \in \mathcal{O} \to \mathcal{O}^\sharp$ satisfies the* commutation inequality *$F^\sharp \circ \alpha \sqsubseteq^\sharp \alpha \circ F$ and that for each post-fixpoint $A \in \mathcal{O}^\sharp$ there exists a post-fixpoint $C \in \mathcal{O}$ such that $\alpha(C) = A$. Then $\alpha(\mathsf{lfp}^{\sqsubseteq}_{\bot} F) = \mathsf{lfp}^{\sqsubseteq^\sharp}_{\bot^\sharp} F^\sharp$.*

In Theorems 4 and 6 Scott-continuity is a too strong hypothesis, since in the proof of [Cousot, 2002] the author only uses the fact that α preserves the lub of the iterates of F starting from $\bot$.

2.2.2.1 Fixpoint Extrapolation

Even if a fixpoint of a function exists, it is not always the case that the iteration sequence computing it converges in finite time. Indeed, Theorem 3 guarantees that $\mathsf{lfp}^{\sqsubseteq}_{\bot} F = F^\lambda$, for a limit ordinal λ, but this latter could be transfinite. Note that in the concrete case we have that the computation does not converge in finite time, this is why we need to move to the abstract domain. Nevertheless, we can have infinite computations also in the abstract domain, for instance when this latter has infinite ascending chains. Indeed, we seek an abstract element such that $\mathsf{lfp}^{\sqsubseteq^\sharp}_{\bot^\sharp} F^\sharp = F^{\sharp\beta}$, with $\beta < \omega$, meaning that $\mathsf{lfp}^{\sqsubseteq^\sharp}_{\bot^\sharp} F^\sharp = \bigsqcup^\sharp_{n<m} F^{\sharp n}(\bot)$, with $m \in \mathbb{N}$. We have that the iteration reaches the fixpoint in finite time when $\mathcal{O}^\sharp$ is finite or when it satisfies the ACC condition. When this is not the case, a classic example is the intervals domain, we need an extrapolation operator forcing convergence (and so termination).

Widening. A widening is a binary operator used to enforce or accelerate the convergence of increasing iteration sequences over abstract domains with infinite or finite but too long ascending chains. Let $\langle X, \leq\rangle$ a POSET, a *widening* $\triangledown \in X \times X \to X$ is an operator satisfying two constraints. First, it must compute upper bounds, namely for every $x, y \in X$ we must have that $x \leq x \triangledown y$ and $y \leq x \triangledown y$. Second, for every increasing chain $x_0 \leq x_1 \leq \ldots \leq$

$x_n \leq \ldots$, the increasing chain $y_0 \triangleq x_0 \leq y_1 \triangleq y_0 \triangledown x_1 \leq \ldots y_n \triangleq y_{n-1} \triangledown x_n \leq \ldots$ stabilizes in a finite number of steps, that is $\exists k \in \mathbb{N} \,.\, y_k = y_{k-1}$. Basically, if x_i are the iterates of the abstract funtion f in the abstract domain, the widening uses two consecutive iterates y_i and $f(y_i)$ in order to obtain the next iteration y_{i+1}. In this way, the widening computes in finite time a post-fixpoint of the abstract function and, in turn, an approximation of the least fixpoint of the concrete function.

2.3 Derived Structures and Abstractions

In this last section we give some constructions, in order to define ordered structures starting from other ordered structures. Furthermore, we do the same for abstractions, namely we show how to derive Galois connections starting from other Galois connections.

Structures.

Definition 6 (Powerset Construction)**.** Let X be a set. Then $\langle \wp(X), \subseteq, \cup, \cap, \varnothing, X, \setminus \rangle$ is a complete Boolean Algebra.

Definition 7 (Dual Structure)**.** Let $\langle X, \leq, \vee, \wedge, \bot, \top \rangle$ be a complete lattice, then we have that $\langle X, \geq, \wedge, \vee, \bot, \top \rangle$ is a complete lattice. The same holds for a lattice, a CPO and a POSET.

Definition 8 (Pointwise Construction)**.** Let $\langle C, \leq, \vee, \wedge, \bot, \top \rangle$ be a complete lattice and X a set. Then $\langle X \to C, \dot{\leq}, \dot{\vee}, \dot{\wedge}, \lambda x \,.\, \bot, \lambda x \,.\, C \rangle$ is a complete lattice, where:

- $f_1 \dot{\leq} f_2 \triangleq (\forall x \in X \,.\, f_1(x) \leq f_2(x))$
- $\dot{\bigvee} Y \triangleq \lambda x \,.\, \bigvee \{ f(x) \mid f \in Y \}$
- $\dot{\bigwedge} Y \triangleq \lambda x \,.\, \bigwedge \{ f(x) \mid f \in Y \}$

The same holds for a lattice, a CPO and a POSET.

Abstractions.

Definition 9 (Compositional Abstraction)**.** Let $\langle X_1, \leq_1 \rangle \underset{\alpha_A}{\overset{\gamma_A}{\leftrightarrows}} \langle X_2, \leq_2 \rangle$ and $\langle X_2, \leq_2 \rangle \underset{\alpha_B}{\overset{\gamma_B}{\leftrightarrows}} \langle X_3, \leq_3 \rangle$ be two Galois connections, then

$$\langle X_1, \leq_1 \rangle \underset{\alpha_B \circ \alpha_A}{\overset{\gamma_A \circ \alpha_B}{\leftrightarrows}} \langle X_3, \leq_3 \rangle$$

is Galois connection. If α_A and α_B are surjective then so is $\alpha_B \circ \alpha_A$, namely we have that $(X_1, \alpha_B \circ \alpha_A, \gamma_A \circ \gamma_B, X_3)$ is a Galois insertion.

Definition 10 (Dual Abstraction)**.** Let $\langle X, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Y, \preccurlyeq \rangle$ be a Galois connection, then we have that $\langle X, \geq \rangle \underset{\gamma}{\overset{\alpha}{\leftrightarrows}} \langle Y, \succcurlyeq \rangle$ is a Galois connection. The same holds for a Galois insertion.

Definition 11 (Subset Abstraction)**.** Let C be a set and $A \subseteq C$, then

$$\langle \wp(C), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(A), \subseteq \rangle$$

with $\alpha(X) \triangleq C \cap A$ and $\gamma(Y) \triangleq Y \cup (C \setminus A)$.

Definition 12 (Elementwise Set Abstraction)**.** Let C and A be two sets, $f \in C \to A$, then

$$\langle \wp(C), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(A), \subseteq \rangle$$

with $\alpha(X) \triangleq \{f(c) \mid c \in X\}$ and $\gamma(Y) \triangleq \{c \mid f(c) \in Y\}$. If f is *surjective* then α is surjective, namely $\leftrightarrows$ becomes $\leftrightarrows\!\!\!\!\rightarrow$.

Definition 13 (Supremum Abstraction)**.** Let $\langle A, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$ be a complete lattice, C be a set and $f \in C \to A$, then

$$\langle \wp(C), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \sqsubseteq \rangle$$

with $\alpha(X) \triangleq \bigsqcup\{f(c) \mid c \in X\}$ and $\gamma(a) \triangleq \{c \mid f(c) \sqsubseteq a\}$.

Definition 14 (FunRel Abstraction)**.** Let D and E be two sets, then

$$\langle \wp(D \to E), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(D \times E), \subseteq \rangle$$

with $\alpha(X) \triangleq \{\langle d, f(d)\rangle \in D \times E \mid f \in X\}$ and $\gamma(Y) \triangleq \{f \in D \to E \mid \forall d \,.\, \langle d, f(d)\rangle \in Y\}$.

Definition 15 (Pointwise Encoding)**.** Let D and E be two sets, then

$$\langle \wp(D \times E), \subseteq, \cup, \cap, D \times E, \varnothing \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows\!\!\!\!\rightarrow}} \langle D \to \wp(E), \dot{\subseteq}, \dot{\cup}, \dot{\cap}, \lambda d \,.\, E, \lambda d \,.\, \varnothing \rangle$$

is a Galois isomorphism, with $\alpha(X) \triangleq \lambda d \,.\, \{e \mid \langle d, e\rangle \in X\}$ and $\gamma(f) \triangleq \{\langle d, e\rangle \mid e \in f(d)\}$.

Definition 16 (Pointwise Abstraction)**.** Let $\langle X, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Y, \preccurlyeq \rangle$ be a Galois connection and Z an arbitrary set. Then

$$\langle Z \to X, \dot{\leq} \rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle Z \to Y, \dot{\preccurlyeq} \rangle$$

is a Galois connection. The same holds for a Galois insertion.

Definition 17 (Non-Relational Abstraction)**.** Let A and B two sets, then

$$\langle \wp(A \to B), \subseteq \rangle \underset{\alpha_{nr}}{\overset{\gamma_{nr}}{\leftrightarrows\!\!\!\!\rightarrow}} \langle A \to \wp(B), \dot{\subseteq} \rangle$$

with $\alpha_{nr}(X) \triangleq \lambda a \,.\, \{f(a) \mid f \in X\}$ and $\gamma_{nr}(\dot{f}) \triangleq \{f \mid \forall a \in A \,.\, f(a) \in \dot{f}(a)\}$.

Definition 18 (Componentwise Abstraction)**.** Let A and B be two sets, then

$$\langle \wp(A \times B), \subseteq, \cup, \cap, A \times B, \varnothing \rangle \underset{\alpha^\times}{\overset{\gamma^\times}{\leftrightarrows}} \langle \wp(A) \times \wp(B), \subseteq^2, \cup^2, \cap^2, \langle A, B\rangle, \langle \varnothing, \varnothing\rangle \rangle$$

with $\alpha^\times(X) \triangleq \langle \{a \mid \exists b \,.\, \langle a, b\rangle \in X\}, \{b \mid \exists a \,.\, \langle a, b\rangle \in X\} \rangle$ and $\gamma^\times(\langle X, Y\rangle) \triangleq X \times Y$. Here, $\subseteq^2 \triangleq \subseteq \times \subseteq$, $\cup \triangleq \cup \times \cup$ and $\cap \triangleq \cap \times \cap$.

3 System Correctness

Systems, from computer programs to biological precesses, should behave as intended. Indeed, every system is designed for a task and the system should complete it, possibly without errors. Unfortunately, errors very often occur and hence some control mechanisms are needed. These latter aim to discover when a system exhibits an unwanted behavior, potentially taking some repairing actions. Basically, these control mechanisms say whether a system is correct or not.

In order to build systems control mechanisms, it is necessary to have to establish what means "to be correct". In the general case, correctness has an informal meaning: given a specification, i.e. an informal requirements description[1] stating what a system could or could not do, a system is *correct* if complies with the specification. Clearly in order to deal with the problem with automatic means it is necessary a more precise definition. For this purpose formal methods are used to prove the correctness of a system.

3.1 Correctness Condition

A control mechanism is itself a system, hence it is legitimate to wonder whether the control mechanism itself is correct. In the following, we refer to the correctness of a control mechanism with the term *soundness*. In this thesis, we take in consideration only control mechanisms which are sound by design. In order to do so, we need to specify all actors needed to build a control mechanism and, in particular, what we call a *correctness condition*.

3.1.1 Informal Introduction

Systems are generic objects so first we need a mathematical representation describing them: the system model. From this latter, we need to retrieve the behavior of a system, namely a representation of its execution. Then, depending on the system model and from the informal requirements, a specification is defined. Finally, a correctness condition is introduced to determine whether the system behavior adheres to the specification or not.

So a control mechanism can be described as follows (see also Figure 3.1). From the behaviors of systems and from a specification the correctness condition is obtained. A security mechanism takes a system and retrieves its behavior. Then it checks whether the latter fulfills the correctness condition. If the answer is positive then the system is correct otherwise the system is incorrect. In this latter case, the mechanism may also perform some actions

[1] The requirements can be even specified in a rigorous mathematical way but are not formal in the sense that there is not a formal link between them and the behavior of the system and/or the execution context.

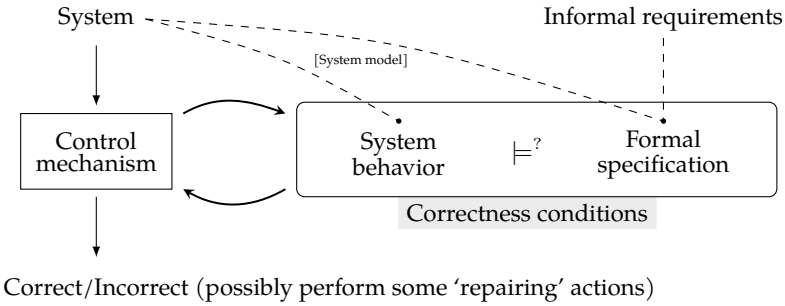


Figure 3.1: Control mechanism process.

in order to revise the system behavior. The control mechanism implements the methods needed for checking the specification.

The correctness condition is also needed to formally prove soundness (only correct system behaviors are accepted by the mechanism) and precision (which correct systems are wrongly ruled out due to mechanism incompleteness) of a given control mechanism. This connection is important as it ensures that the mechanism actually guarantees the specification in the sense of the correctness condition.

To sum up, a specification describes, more or less formally, which requirements a system should fulfill. The system model specifies how to represent the behavior of the system. The correctness condition links the two aspects, describing formally what it means for a system to be correct.

3.1.2 Going (a Little Bit) Formal

As we have seen, with a control mechanism we want to state whether a system is correct or not, namely we want to answer the following question: "is a given system correct?". This implies a formalization of "to be correct", namely the formalization of correctness conditions. In order to define this concept, we take into account a more general setting. Indeed "to be correct" is a particular *property* a system could have.

Suppose we have a set of systems, called Sys, with elements $\mathbb{s} \in \mathsf{Sys}$. In order to reason about systems with automatic means we need to represent these latter as mathematical objects. We need, indeed, a representation function $\mathcal{I} \in \mathsf{Sys} \to \mathbb{REP}^{\mathsf{Sys}}$, mapping a system with its formal representation in $\mathbb{REP}^{\mathsf{Sys}}$. This step is not necessarily automatic indeed, usually, it involves a human interaction.

Remark. Very often, the representation function $\mathcal{I}$ is intended as the *semantics*, formalizing the behavior of systems. Here we have chosen a different name since $\mathcal{I}$ is not restricted to represent semantic information about systems. Indeed, $\mathcal{I}$ could output syntactic information, as the code of a program. This is not a real concern since we can always see the syntax as a particular semantics interpreting symbols with themselves. As a consequence, we use another term in order to avoid confusion.

At this point, a systems property $\mathbb{Q}$ is a function in $\mathbb{REP}^{\mathsf{Sys}} \to \{\text{"yes"}, \text{"no"}\}$ such that, given a system representation $\mathcal{I}(\mathbb{s})$, it returns "yes" if $\mathbb{s}$ has the property $\mathbb{Q}$ and "no" other-

wise. How to define properties from the informal requirements is another not trivial task and, again, it can involve human interaction. From now on, we refer a system property with the term *specification*. This is due to the fact that the term property will be overloaded in the rest of the thesis.

Specifications are not necessarily injective, indeed more than one system could satisfy the same specification. There are also specifications which are not surjective. Indeed, the function $\lambda \mathbb{s}\,.$ "no" (not surjective) represents, informally, "not to be a system", which is not satisfiable by any system. Dually, the specification saying "to be a system" is the function $\lambda \mathbb{s}\,.$ "yes" and it is satisfied by every system. The satisfaction relation $\models \subseteq \mathbb{REP}^{\mathsf{Sys}} \times (\mathbb{REP}^{\mathsf{Sys}} \to \{\text{"yes"}, \text{"no"}\})$ associates a system representation with the specifications it satisfies, i.e. $\mathcal{I}(\mathbb{s}) \models \mathbb{Q}$ if and only if $\mathbb{Q}(\mathcal{I}(\mathbb{s})) = \text{"yes"}$.

Reasoning with functions is sometimes not convenient, hence we redefine specifications in terms of sets. Indeed, a specification induces a trivial partition of $\mathbb{REP}^{\mathsf{Sys}}$. We can hence represent a specification $\mathbb{Q}$ as the set of system representations satisfying it, i.e. $\{\mathcal{I}(\mathbb{s}) \in \mathbb{REP}^{\mathsf{Sys}} \mid \mathcal{I}(\mathbb{s}) \models \mathbb{Q}\} = \{\mathcal{I}(\mathbb{s}) \in \mathbb{REP}^{\mathsf{Sys}} \mid \mathbb{Q}(\mathcal{I}(\mathbb{s})) = \text{"yes"}\}$. In the following, we abuse notation denoting with $\mathbb{Q}$ both a specification and its set-encoding. Hence, we have that system representations are in $\mathbb{REP}^{\mathsf{Sys}}$, specifications are in $\mathsf{Specs} \triangleq \wp(\mathbb{REP}^{\mathsf{Sys}})$ and the satisfaction relation is set-membership.

Definition 19. A system $\mathbb{s} \in \mathsf{Sys}$, whose representation is $\mathcal{I}(\mathbb{s}) \in \mathbb{REP}^{\mathsf{Sys}}$, *satisfies* a specification $\mathbb{Q} \in \wp(\mathbb{REP}^{\mathsf{Sys}})$, written $\mathbb{s} \models \mathbb{Q}$, if and only if $\mathcal{I}(\mathbb{s}) \in \mathbb{Q}$.

In logic, if all models of a formula ϕ are also models of another formula ψ then ψ is said a logical (or semantic) entailment of ϕ. In this case, ϕ logically implies ψ, written $\phi \Rightarrow \psi$. In the setting of specifications, we have that the logical implication is set-inclusion, meaning that if $\mathbb{Q}_1 \subseteq \mathbb{Q}_2$ then all systems satisfying $\mathbb{Q}_1$ also satisfy $\mathbb{Q}_2$. We have that $\langle \mathsf{Specs}, \subseteq \rangle$ is a POSET with a bottom element $\varnothing$ and a top element $\mathbb{REP}^{\mathsf{Sys}}$. These two elements represent, respectively, the strongest and the weakest specification. Indeed, $\mathbb{Q}_1 \subseteq \mathbb{Q}_2$ means that more systems satisfy $\mathbb{Q}_2$ than $\mathbb{Q}_1$, hence $\mathbb{Q}_2$ is, in some sense, weaker than $\mathbb{Q}_1$. In fact, $\varnothing$ is not satisfied by any system and, dually, $\mathbb{REP}^{\mathsf{Sys}}$ is satisfied by every system (they are, respectively, the specifications $\lambda \mathbb{s}\,.$ "no" and $\lambda \mathbb{s}\,.$ "yes" introduced before).

Given a system $\mathbb{s}$ we can hence retrieve its *strongest specification*, which is $\{\mathcal{I}(\mathbb{s})\}$. Indeed, $\{\mathcal{I}(\mathbb{s})\} \subseteq \mathbb{Q}$, for any specification $\mathbb{Q}$ such that $\mathbb{s} \models \mathbb{Q}$. This means that we can use the logical implication, namely set-inclusion, in order to state satisfiability. Indeed we have:

$$\mathbb{s} \models \mathbb{Q} \text{ if and only if } \{\mathcal{I}(\mathbb{s})\} \subseteq \mathbb{Q} \tag{3.1}$$

This is important because $\subseteq$ is much easier to be used than $\in$ in the context of approximation-based control mechanisms. Here, $\subseteq$ is used as the approximation order, since $\langle \mathsf{Specs}, \subseteq \rangle$ is the POSET[2] of systems specifications, where $\subseteq$ compares specifications w.r.t. their degree of approximation.

Remark. In the context of program analysis, $\mathcal{I}(\mathbb{s})$ corresponds to the *standard semantics* (of $\mathbb{s}$) and $\{\mathcal{I}(\mathbb{s})\}$ corresponds to a *collecting semantics* (of $\mathbb{s}$). The collecting semantics is intended as the specification $\mathbb{s}$ *does* have, whence, an arbitrary specification $\mathbb{Q}$ is intended as a specification $\mathbb{s}$ *may* have.

The correctness condition is given by a specification $\mathbb{Q}$ and its satisfiability relation $\models$. A system $\mathbb{s}$ is correct w.r.t. $\mathbb{Q}$ if and only if $\mathbb{s}$ satisfies $\mathbb{Q}$, i.e. if and only if $\mathbb{s} \models \mathbb{Q}$.

[2] Actually $\langle \mathsf{Specs}, \subseteq, \cup, \cap, \varnothing, \mathbb{REP}^{\mathsf{Sys}}, \setminus \rangle$ is complete Boolean Algebra.

3.2 Control Mechanisms: Verification vs Enforcement

Now that we have made clear what is a correctness condition, we can start talking about control mechanisms. As already said, a control mechanism implements the correctness condition and, optionally, it performs some repairing actions if it finds an incorrect system. Hence, we can distinguish two kinds of control mechanisms: verifiers and enforcers.

Verification is intended as the general process of checking if a system complies with a specification. So verification checks if a system is correct or not. This process can be static or dynamic. In the *static* case the analysis of the system is done without executing it (e.g. analyzing its code if the system is a program). If the specification control is performed during the execution of the system, so *dynamically*, then the process is called runtime verification. A verifier (runtime or not) can only figure out if a system performs not allowed actions, nothing else. Giving to the verifier the power to modify the system execution, a mechanism of *enforcement* is obtained. So enforcement assures that the system under control is correct, or better, assures that it behaves like a correct system. Also in this case there is the possibility to perform the process statically or dynamically (runtime enforcer).

The power of the verification/enforcement mechanisms, i.e. the specifications that they are able to verify/enforce, depends on their "capabilities" or, dually, on the constraints they have to respect. For instance, a mechanism could be limited to only observe the behavior of the system, i.e. it does not have extra information other than the actions executed by the system. This type of mechanisms does not have the possibility to look-forward in the system execution, i.e. decisions rely only on the actions already seen. Furthermore, an unrestricted runtime enforcer can stop, insert, delete, delay and edit all the actions performed by the system under control. In addition it could perform some analysis of the system and look-forward for retrieving information about the future actions executed. This modifications to the system behavior can also be performed syntactically before its execution, so statically, and they are termed *rewriting*. A particular class of mechanisms is defined restricting the power of a runtime enforcer to only stop system executions, and it is called *monitoring*.

In any case, a verifier is formally a function $\texttt{Verifier} \in \mathsf{Specs} \times \mathsf{Sys} \to \{\text{“yes”}, \text{“no”}\}$ implementing the correctness condition, namely

$$\texttt{Verifier}(\mathbb{Q}, \mathbb{s}) = \begin{cases} \text{“yes”} & \text{if } \mathbb{s} \models \mathbb{Q} \\ \text{“no”} & \text{otherwise} \end{cases}$$

Enforcement is a slightly different process, since it has to modify the systems under control. Indeed, enforcement is the process of retrieving the closest system $\mathbb{s}'$ such that $\mathbb{s}' \models \mathbb{Q}$. An enforcer could be defined as a function $\texttt{Enforcer} \in \mathsf{Specs} \times \mathsf{Sys} \to \mathsf{Sys}$ such that:

$$\texttt{Enforcer}(\mathbb{Q}, \mathbb{s}) = \mathbb{s}' \quad \text{and} \quad \mathbb{s}' \models \mathbb{Q} \quad \text{and} \quad \mathbb{s} \simeq \mathbb{s}'$$

for a given equivalence relation $\simeq \subseteq \mathsf{Sys} \times \mathsf{Sys}$. The definition of this latter is crucial, since it reflects the precision of the enforcement process. For instance, it is easy to make an enforcer for the trivial equivalence $\simeq \triangleq \mathsf{Sys} \times \mathsf{Sys}$.

Example 1. Given a specification $\mathbb{Q}$, it is not so hard to retrieve a system (maybe a trivial one) $\hat{\mathbb{s}}$ satisfying $\mathbb{Q}$. Then an enforcement for the equivalence relation $\simeq \subseteq \mathsf{Sys} \times \mathsf{Sys}$ is trivially definable as $\texttt{Enforcer}(\mathbb{Q}, \mathbb{s}) \triangleq \hat{\mathbb{s}}$.

Remark. Usually, a control mechanism is built specifically for a particular specification. Hence it is more realistic to say that a verifier (and similarly, an enforcer) is a function in $\mathsf{Sys} \to \{\text{“yes”}, \text{“no”}\}$, parametric on $\mathbb{Q}$, namely $\texttt{Verifier}^{\mathbb{Q}}(\mathbb{s}) \triangleq \llbracket \mathbb{s} \models \mathbb{Q} \,?\, \text{“yes”} : \text{“no”} \rrbracket$.

Both verification and enforcement mechanisms should be feasible, i.e. implementable in some way. Unfortunately, due to Rice's theorem, every non trivial (semantics) correctness condition is undecidable, hence verification mechanisms need approximations. For what concerns enforcement, it is always possible to enforce every correctness condition, as long as we sacrifice precision. In this thesis we do not deal with enforcement, we have just sketched the idea here. Hence, in the next subsection, we talk about approximate verification, in order to make the problem decidable.

3.2.1 Analysis and Approximations

Systems *analysis* is a related, but yet slightly different, concept compared to verification. As already said, in verification we want to answer the question: “does a system satisfy a given specification?”. In analysis, instead, we do not have the concept of specification to verify. Indeed, an analysis of a system returns the most precise information we are able to retrieve about the system. Usually, a system could fulfill more than one specification, hence, a system analysis returns the strongest specification fulfilled by the system. This means that analysis is an automatic means for *computing* $\{\mathcal{I}(\mathbb{s})\}$ and verification, instead, is an automatic means for *computing* $\{\mathcal{I}(\mathbb{s})\} \subseteq \mathbb{Q}$.

Unfortunately, in general, we are able to compute neither $\{\mathcal{I}(\mathbb{s})\}$ nor $\{\mathcal{I}(\mathbb{s})\} \subseteq \mathbb{Q}$. Indeed, we need approximations. A set of *abstract specification* $\mathsf{Specs}^\sharp$, which we are able to compute, represents concrete specifications in a simpler way, usually losing some information. Clearly, we have to define when an abstract specification is a sound approximation of a concrete one, namely we need a soundness function $\varsigma \in \mathsf{Specs}^\sharp \to \mathsf{Specs}$. This latter naturally induces a preorder on $\mathsf{Specs}^\sharp$ (a partial order if ς is injective), defined as: $\subseteq^\sharp \triangleq \{\langle \mathbb{Q}_1^\sharp, \mathbb{Q}_2^\sharp \rangle \mid \varsigma(\mathbb{Q}_1^\sharp) \subseteq \varsigma(\mathbb{Q}_2^\sharp)\}$[3]. The order $\subseteq^\sharp$ mimics the approximation order $\subseteq$ in the abstract, namely $\mathbb{Q}_1^\sharp \subseteq^\sharp \mathbb{Q}_2^\sharp$ implies that the set of systems satisfying $\varsigma(\mathbb{Q}_1^\sharp)$ is a subset of the set of systems satisfying $\varsigma(\mathbb{Q}_2^\sharp)$. Note that we can express this in the abstract interpretation framework, where the concrete objects domain is $\langle \mathsf{Specs}, \subseteq \rangle$, namely concrete specifications, the abstract objects domain is $\langle \mathsf{Specs}^\sharp, \subseteq^\sharp \rangle$, namely approximate specifications, and the concretization function γ is exactly ς.

Remark. Note that an abstraction is always (silently) performed in classic program analysis. Systems are represented with sets of executions and hence, technically, specifications are sets of sets of executions. Trace properties are sets of executions, namely they are already abstract specifications. In this case, the abstract approximation order remains set-inclusion and the soundness function is the powerset operator. Finally, in this setting the interpretation of a system, i.e. a set of executions, plays also the role of the strongest specification.

In this setting, the analysis aims to compute a sound abstract specification of a given system, namely it could be defined as a function $\texttt{Analyzer}^\sharp \in \mathsf{Sys} \to \mathsf{Specs}^\sharp$ such that:

$$\{\mathcal{I}(\mathbb{s})\} \subseteq \varsigma(\texttt{Analyzer}^\sharp(\mathbb{s}))$$

[3]This definition is very strong and, in practice, we settle for an order $\subseteq^\sharp \subseteq \{\langle \mathbb{Q}_1^\sharp, \mathbb{Q}_2^\sharp \rangle \mid \varsigma(\mathbb{Q}_1^\sharp) \subseteq \varsigma(\mathbb{Q}_2^\sharp)\}$. This is sufficient to guarantee soundness and it is much easier to achieve in practice: $\subseteq^\sharp$ becomes a semi-test.

Hence, even if we are not able to compute $\{\mathcal{I}(\mathbb{s})\}$, with an (abstract) analyzer we can compute an approximate version $\texttt{Analyzer}^\sharp(\mathbb{s})$ of $\{\mathcal{I}(\mathbb{s})\}$, which is sound. We can do a similar reasoning also for verification. In approximate verification we obtain an approximate answer to the question: "does a system satisfy a specification?". Indeed, an abstract verifier could answer "yes" or "maybe", where this latter means that the verifier is not able to say anything about the verification. The check $\{\mathcal{I}(\mathbb{s})\} \subseteq \mathbb{Q}$ is approximated totally in the abstract, given an under-approximation of the specification we want to verify. An abstract specification $\mathbb{Q}^\sharp$ is an under-approximation of a concrete specification $\mathbb{Q}$ when $\varsigma(\mathbb{Q}^\sharp) \subseteq \mathbb{Q}$. The verifier must be sound, hence it is a function $\texttt{Verifier}^\sharp \in \mathsf{Specs}^\sharp \times \mathsf{Sys} \to \{\text{"yes"}, \text{"maybe"}\}$ such that:

$$\texttt{Verifier}^\sharp(\mathbb{Q}^\sharp, \mathbb{s}) = \text{"yes"} \;\text{ implies }\; \{\mathcal{I}(\mathbb{s})\} \subseteq \varsigma(\mathbb{Q}^\sharp) \;\text{ (in turn implying } \mathbb{s} \models \mathbb{Q})$$

Note that we can define an (abstract) verifier in terms of an (abstract) analyzer as follows:

$$\texttt{Verifier}^\sharp(\mathbb{Q}^\sharp, \mathbb{s}) \triangleq \begin{cases} \text{"yes"} & \text{if } \texttt{Analyzer}^\sharp(\mathbb{s}) \subseteq^\sharp \mathbb{Q}^\sharp \\ \text{"maybe"} & \text{otherwise} \end{cases}$$

If the answer of the analyzer is "maybe" then we have to perform another verification process, refining $\mathbb{Q}^\sharp$ and/or changing the abstract specification domain $\mathsf{Specs}^\sharp$. In any case, it is not guaranteed to give a definitive answer to the problem.

3.3 Safety and Confidentiality Requirements

Specifications are very generic and, indeed, they could involve every aspect of a system. They could be syntactic, namely describing structural aspects of a system, or they could be semantic, namely involving the behavior of a system. Clearly, these latter are more interesting but, at the same time, more challenging to verify and enforce. In particular, from a practical point of view, in system verification/enforcement not all specifications are worth to be checked. Usually, the prominent specifications are those specifying some safety or confidentiality/integrity requirement. The first, basically, say that the system does not go[goes] in an error[goal] state, whichever the definition of error[goal] state is. For instance, a system supposed to run forever should not stop inadvertently its execution. Other examples are *partial correctness*, which guarantees that all terminating executions produce correct results, and *mutual exclusion*, which guarantees that concurrent processes do not enter their critical section at the same time. These specifications are "safety-like", in the sense that they require systems to never reach unwanted states. Conversely, "liveness-like" specifications require that a goal state is *eventually* reached. Examples of this kind are program *termination*, which guarantees that all program computations do not run forever, and *starvation freedom*, which guarantees that a system process will eventually enter its critical section.

The specifications expressing confidentiality/integrity requirements, aim to guarantee that sensitive information is managed in a correct way, during the execution of the system. They are the leading motivation for hyperproperties, and in turn for this thesis, hence we introduce them in a more detailed way in the next subsection.

3.3.1 Information Flow

Information flow control tries to guarantee confidentiality and integrity of the data manipulated during the execution of a system. For confidentiality, sensitive information must be prevented from flowing to public destinations, and dually, for integrity, untrusted information must be prevented from affecting, or flowing to, trusted destinations[4]. These two notions are dual, meaning that the reasoning for the first applies also for the second, modulo an isomorphic representation. For this reason, in the following we deal with confidentiality information flow control only.

Computing systems confidentiality relies on how information is propagated during systems execution. Historically, *access control* has been the main means of preventing information from being disseminated. As the name indicates, access control verifies the system rights at the entry-point. However, it is inadequate in many situations, in fact the system may leak/compromise information after the access check. Instead *information flow control* tracks how information propagates through the system during execution to make sure that the system handles the information securely. In this section when we use the term security we mean confidentiality.

One (or more) action of the system that uses the value of some object x to derive a value for another object y causes a *flow* from x to y. The system handles information in a secure way, w.r.t. a flow policy, if the flows it causes are permitted by the policy. A *flow policy* is usually represented by a lattice $\langle \mathcal{L}, \rightsquigarrow \rangle$, where $\mathcal{L}$ is a given set of security levels and $\rightsquigarrow$ is a flow relation specifying permissible flows between pairs of levels. Each object x is assigned to a security level $\Gamma(x) \in \mathcal{L}$, so $\Gamma(x) \rightsquigarrow \Gamma(y)$ means that a flow from object x to object y is permissible in the flow policy. With a little abuse of notation, we denote an information flow from object x to object y as $x \rightsquigarrow y$. So a system $\mathbb{s}$ is secure if and only if no executions of $\mathbb{s}$ result in a flow $x \rightsquigarrow y$, unless $\Gamma(x) \rightsquigarrow \Gamma(y)$.

An information flow $x \rightsquigarrow y$ could be of two types: explicit or implicit. It is *explicit* when the system actions generating it do not depend on the value of x. It is *implicit* when the system causes a flow from another object z to y but the execution depends on the value of x. Furthermore, an information flow $x \rightsquigarrow y$ is *direct* if a system action transfers directly the value (or a function of the value) of x to y. Otherwise it is *indirect*, i.e. the transfer of the value from x to y involves an intermediate object z.

`y := x`	explicit
`if x == 5 then y := z`	implicit
`y := abs(x)`	direct
`z := abs(x); y := z`	indirect

Figure 3.2: Examples of flow types $x \rightsquigarrow y$.

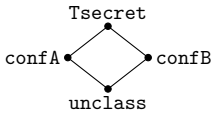
Secure information flow is comprised of two related aspects: information *confidentiality* and information *integrity*. The first says that information should not be disclosed by the system in a not permitted way (some kind of read-protection). The second says that information should not be altered by the system in a not permitted way (some kind of write-protection). Despite the fact that these two notions are duals, researchers in information

[4]The availability aspect of the CIA trinity [Stallings and Brown, 2007] is not taken into account in information flow control.

flow control have worked basically only on the confidentiality side, in particular on one of its formalization: Non-Interference.

In the last forty years, a lot of security conditions have been proposed (noninterference [Goguen and Meseguer, 1982], nondeducibility [Sutherland, 1986], restrictiveness [McCullough, 1987], separability [McLean, 1996], etc). They differ on the formal definitions but share the following common informal understanding of confidentiality: the lack of dependencies on confidential information. This is precisely the absence of *strong dependency* of [Cohen, 1977] and it is the base of confidentiality information flow specifications. The idea of Cohen is that there exists an information flow from x to y in a system $\mathbb{s}$ whenever variety in x is conveyed to y by the execution of $\mathbb{s}$ (strong dependency). So a system respects a flow policy, i.e. it is secure, if there is no strong dependency between objects x, y unless $\Gamma(x) \rightsquigarrow \Gamma(y)$.

Example 2. Suppose the flow policy is specified by the following Hasse diagram. The only information flows allowed (other the trivial reflexive ones) are: $\texttt{unlcass} \rightsquigarrow \texttt{confA}, \texttt{unlcass} \rightsquigarrow \texttt{confB}, \texttt{unlcass} \rightsquigarrow \texttt{Tsecret}, \texttt{confA} \rightsquigarrow \texttt{Tsecret}, \texttt{confB} \rightsquigarrow \texttt{Tsecret}$. Hence, for instance, we cannot have a flow from `confA` to `unclass`.



The majority of Non-Interference formulations take into account only two security levels: *private*, i.e. information that has to be kept secret, and *public*, i.e. information that could be freely released. So they represent a security lattice $\langle \{\mathsf{L}, \mathsf{H}\}, \rightsquigarrow \rangle$ where H (high) is the private level, L (low) is the public level and the only flows not permitted are those from H to L (so $\rightsquigarrow \triangleq \{\langle \mathsf{L}, \mathsf{L} \rangle, \langle \mathsf{L}, \mathsf{H} \rangle, \langle \mathsf{H}, \mathsf{H} \rangle\}$). Furthermore, in the classic definition, Non-Interference is expressed as the correlation between input and output data only and is given for deterministic systems. With these premises, a system is said to be *non-interfering* if the values of the public outputs do not depend on the values of the private inputs. Following the standard notation based on the equivalence relation $\overset{\mathsf{L}}{=}$, Non-Interference can be expressed as follows, with Σ denoting the set of system states and $[\![\mathbb{s}]\!]$ the execution of the system $\mathbb{s}$:

$$\forall \sigma_1, \sigma_2 \in \Sigma \,.\, \sigma_1 \overset{\mathsf{L}}{=} \sigma_2 \Rightarrow [\![\mathbb{s}]\!]\sigma_1 \overset{\mathsf{L}}{=} [\![\mathbb{s}]\!]\sigma_2 \tag{3.2}$$

More precisely, $[\![\mathbb{s}]\!]\sigma = \sigma'$ means that the system $\mathbb{s}$, executed on the state σ, yields the state σ'. The equivalence relation $\overset{\mathsf{L}}{=}$ says that two states are equivalent, modulo the public data. So a system $\mathbb{s}$ is said to be non-interfering if and only if for any two states σ_1, σ_2 having the same low data (written $\sigma_1 \overset{\mathsf{L}}{=} \sigma_2$), the executions of $\mathbb{s}$ in the initial states σ_1, σ_2 produce states with the same low data.

Remark. As already said, information flows can also be seen from the point of view of integrity, with a dual definition. In this case the values of the private outputs are not influenced by the values of the public inputs, i.e. they are not altered by public, untrusted, inputs. This is simply modeled inverting the flow relation, so $\rightsquigarrow \triangleq \{\langle \mathsf{L}, \mathsf{L} \rangle, \langle \mathsf{H}, \mathsf{L} \rangle, \langle \mathsf{H}, \mathsf{H} \rangle\}$. The definition of Non-Interference becomes:

$$\forall \sigma_1, \sigma_2 \in \Sigma \,.\, \sigma_1 \overset{\mathsf{H}}{=} \sigma_2 \Rightarrow [\![\mathbb{s}]\!]\sigma_1 \overset{\mathsf{H}}{=} [\![\mathbb{s}]\!]\sigma_2$$

where $\overset{\mathsf{H}}{=}$ has a similar meaning as $\overset{\mathsf{L}}{=}$, but on high data.

In a more general setting, the definition may additionally involve other aspects such as power consumption, computation time, termination, a different model of systems etc, and so there are a lot of variants of the notion of Non-Interference. One of these is Abstract Non-Interference [Giacobazzi and Mastroeni, 2004] and it states that there is no strong dependency between some properties of the secret inputs and some properties of the public outputs. The properties on outputs are those that an attacker can distinguish. The properties on inputs are those that must be kept secret. This is important because it deals naturally with *declassification*. In fact, for practical uses, the definition of non-interference as in Equation 3.2 is too restrictive: sometimes it is necessary to release some confidential information in order to make a system useful (*selective dependencies* of [Cohen, 1977]). Moreover, Abstract Non-Interference formalizes, in a unifying framework, also the other method to weak Non-Interference: constraining the attacker power, namely limiting what is observable about systems execution. As the name indicates, Abstract Non-Interference generalizes Non-Interference by means of abstract interpretation.

3.3.2 Different Flavors of Non-Interference

3.3.2.1 Qualitative vs Quantitative Information Flow

Let $\langle \mathcal{L}, \rightsquigarrow \rangle$ be a security lattice, where $\mathcal{L} = \{\ell_1, \ell_2, \dots \ell_n\}$ is a finite set of security levels and $\rightsquigarrow$ is the flow relation, stating which information flows are allowed. Given a system $\mathbb{s}$, its security typing Γ specifies the security clearance of every object occurring in $\mathbb{s}$. We can have an information flow from x to y if and only if $\Gamma(x) \rightsquigarrow \Gamma(y)$.

Given a security level $\ell \in \mathcal{L}$ we can define a relation on states stating that two states are ℓ-equivalent, namely that the two states agree on objects having security level at most ℓ. Formally, $\overset{\ell}{=} \subseteq \Sigma \times \Sigma$ is defined as:

$$\sigma \overset{\ell}{=} \sigma' \triangleq (\forall x \,.\, \Gamma(x) \rightsquigarrow \ell \Rightarrow \sigma(x) = \sigma'(x))$$

With these notions we can define the security specification ℓ-Non-Interference, stating that a system executed in two states ℓ-equivalent must terminate in two states ℓ-equivalent:

$$\forall \sigma, \sigma' \in \Sigma \,.\, \sigma \overset{\ell}{=} \sigma' \Rightarrow [\![\mathbb{s}]\!]\sigma \overset{\ell}{=} [\![\mathbb{s}]\!]\sigma'$$

We say that a system $\mathbb{s}$ is *secure*, w.r.t. $\langle \mathcal{L}, \rightsquigarrow \rangle$ and Γ, if and only if it does not exhibit forbidden information flows, namely if and only if it satisfies ℓ-Non-Interference for every security level $\ell \in \mathcal{L}$. Note that the Non-Interference check for the highest security level $\ell_\top$, i.e. the top element of the lattice, is not useful since every flow from $\ell \in \mathcal{L}$ to $\ell_\top$ is always allowed.

Example 3. The classic Non-Interference is obtained with the two security levels $\{\mathsf{L}, \mathsf{H}\}$ and the only allowed flow $\mathsf{L} \rightsquigarrow \mathsf{H}$. Then Non-Interference checks only L-equivalence, namely Non-Interference coincides with L-Non-Interference. Hence, a system is secure if and only if it satisfies L-Non-Interference.

Specifications of this kind are called *qualitative* information flow specifications, since they only express the fact that there is a leakage of confidential information. They are opposed to *quantitative* information flow specifications, designed to measure how much information have been, potentially, leaked. In particular, their aim is to retrieve how much an attacker

may learn (about confidential information) by observing the input/output behavior of a system on public objects. In a deterministic setting, all information in the output has to come from the input, and what is not provided by the public input has to be provided by the private input. Therefore, we can characterize how much of the information carried by the private input to a system can be learned observing the low output. This means that any variation of the output is due to a variation of the input.

The leakage is measured in terms of the number of private values that can be accurately distinguished by an attacker able to observe only public values. The idea is that if there are k such distinguishable values, then in a private object it is possible to encode an arbitrary number in $[0, k[$ that can be leaked through a public object. In other words, $\log_2 k$ bits of sensitive information could flow.

More formally, let $\Sigma_\ell \triangleq \{X \subseteq \Sigma \mid \forall \sigma, \sigma' \in X \,.\, \sigma \overset{\ell}{=} \sigma'\}$ be the set of all possible subsets of states having the same values for objects with security level at most ℓ. Then consider an object x such that $\Gamma(x) \rightsquigarrow \ell$. The set of sets $\{\{([\![\mathbb{s}]\!]\sigma)_{|x} \mid \sigma \in X\} \mid X \in \Sigma_\ell\}$ collects all the possible values the object x may have executing the system $\mathbb{s}$ on states ℓ-equivalent. Then, the leakage of $\mathbb{s}$ on x, w.r.t. an attacker with observation power ℓ, is given by:

$$\texttt{leak}_x^\ell(\mathbb{s}) \triangleq \log_2 \left(\max\left\{|\{([\![\mathbb{s}]\!]\sigma)_{|x} \mid \sigma \in X\}| \;\middle|\; X \in \Sigma_\ell\right\}\right)$$

This measures the number of bits of sensitive data (i.e. with clearance greater than ℓ) the attacker could retrieve observing the execution of $\mathbb{s}$ over x. Given a threshold $k \in \mathbb{N}$, we can say that a system is secure if it does not leak more than k bits, namely if it satisfies ℓ-Non-Interference^k:

$$\forall x \,.\, \Gamma(x) \rightsquigarrow \ell \Rightarrow \texttt{leak}_x^\ell(\mathbb{s}) \leq k$$

The qualitative notion of ℓ-Non-Interference coincides with ℓ-Non-Interference^0, since it requires that no information about sensitive data is leaked.

3.3.2.2 Declassification and Attacker's Power

The classic Non-Interference is based on a characterization of the attacker that does not impose any observational (or complexity) restriction on the attackers' power. In the literature, we can find mainly two approaches for weakening Non-Interference: constraining the power of the attacker (from the observational or the computational point of view); or allowing some confidential information to flow.

Despite the fact that we can see quantitative information flow as a form of declassification, indeed we can let that some bits of information could leak, it is arguable to have a more qualitative approach, aiming to discover which is the information that flows. Declassifying information means downgrading the sensitivity of data in order to accommodate with (intentional) information leakage. The complete separation between private and public objects assumed by Non-Interference is sometimes too strong. As a classic example, consider a login form taking a password from the user, a public object, and comparing it with the correct password, a private object. The output of the form, a public object, alerts the user whether the password inserted is correct or not. Clearly, from the point of view of Non-Interference there is a forbidden information flow. In this case the output of the form should be declassified.

Remark. Integrity is the dual of confidentiality and, indeed, there is also the dual concept of declassification, called *endorsement*. As an example, an untrusted input used to form a

database query can be safely considered trustworthy after a sanitization step. Again, the Non-Interference condition is violated due to the flow of information from untrusted input to some trusted output. In this case the sanitized input should be endorsed.

Both these aspects of Non-Interference weakening can be modeled in the abstract interpretation framework, by means of the notion of Abstract Non-Interference.

3.3.2.3 Abstract Non-Interference

Abstract Non-Interference [Giacobazzi and Mastroeni, 2004] is a possible way for weakening Non-Interference. In this context, an attacker is seen as an abstract interpreter analyzing the system with the intent of revealing properties of secret data, observing the public one. Furthermore the amount of information released can be modeled by the property which is allowed to flow from private to public objects.

Basically, Abstract Non-Interference states that there is no strong dependency between some properties of the private input data and some properties of the public output data. The property on output is the one that an attacker can distinguish. The property on input is the one that must be kept secret. This is important because it deals naturally with declassification. Moreover Abstract Non-Interference formalizes, in a unifying framework, also the other method for weakening Non-Interference: constraining the attacker power, namely limiting what is observable about system behaviors. So Non-Interference is made parametric on two abstractions, each one modeling different aspects of the information flow: the observer (attacker) and what is allowed to flow. In general, we can suppose that the observer may not have the same constraints in observing inputs and outputs. Therefore it is possible to consider three (potentially) different abstractions: one on the public input, one on the private input and one on the public output. The attacker is a pair of abstractions (we deliberately let the concept of abstraction vague for now, we define it precisely in Chapter 8) $\langle \eta, \rho \rangle$ which represents what can be observed about the public input/output of the system.

For the declassification, in the private input we can specify what information is allowed to flow. In this case, it is possible to select for which sets of inputs Non-Interference has to be checked. Let ϕ be the input property representing the inputs that need to be checked. This models the information that may flow since it is not interesting to check if its variation is visible through the output.

With these premises Abstract Non-Interference definition is given by a little modification of Equation 3.2:

$$\forall \sigma_1, \sigma_2 \in \Sigma \,.\, \phi(\sigma_1) = \phi(\sigma_2) \Rightarrow \rho([\![\mathbb{s}]\!](\eta(\sigma_1))) = \rho([\![\mathbb{s}]\!](\eta(\sigma_2))) \tag{3.3}$$

This means that a system $\mathbb{s}$ satisfies abstract Non-Interference relatively to an output observation ρ, an input property η to protect and an input property ϕ that may flow if, whenever the input values have the same property ϕ, then the best correct approximation of the system $\mathbb{s}$ semantics w.r.t. η in input and ρ in output does not report any change, meaning that the variation of η does not affect the observation ρ in $\mathbb{s}$.

4

Systems, Semantics and Specifications

Chapter 3 introduces the concept of correctness condition and it describes, in a general fashion, control mechanisms. The focus of the thesis is to develop verification mechanisms for computer programs, dealing with specifications formalized as hyperproperties. In the following chapter, we go more in detail on the definition of a control mechanism. In the first section we define the systems model we have chosen, while in the second section we introduce hyperproperties, i.e. our specifications of interest.

Remark. In this work we are interested in the behavior of systems, formalizing the possible executions of a system. Indeed, both system models and specifications deal with semantic aspects of systems.

4.1 Systems and Semantics

In order to represent systems, there are a plethora of different mathematical models we can use. We chose transition systems, since they are very expressive and well suitable to deal with programs semantics. In the following, we introduce transition systems and, at the end, we show how to retrieve a transition system out of a program written in a simple imperative language.

4.1.1 Transition Systems

We model systems behaviors as transition systems, which are a very general way for expressing discrete dynamic systems semantics. The idea is to model system executions by means of states, representing the most precise information we have about systems in a particular interval of time, and transitions between states.

Definition 20 (Transition System)**.** A *transition system* is a tuple $\langle \Sigma, \tau, \Upsilon, \Omega \rangle$ where

- Σ is a, possibly infinite, set of *states*;
- $\tau \subseteq \Sigma \times \Sigma$ is the *transition relation* between states, namely it describes the successor state(s) of a given one;
- $\Upsilon \subseteq \Sigma$ is a designated set of *initial states*;
- $\Omega \subseteq \Sigma$ is a designated set of *final states*, namely states σ such that $\forall \sigma' \in \Sigma \,.\, \langle \sigma, \sigma' \rangle \notin \tau$

Sometimes transition systems are defined without initial and final states, which is equivalent to say that $\Upsilon = \Sigma$ and $\Omega = \varnothing$. In the following, we use the notation $\sigma \; \tau \; \sigma'$, in place of $\langle \sigma, \sigma' \rangle \in \tau$, denoting the fact that the system changes state, from σ to σ'.

Remark. In the literature, *labeled* transition systems are often used, in order to model non-determinism in programs' semantics. They are transition systems augmented with a set Δ of *actions* and with a *labeled* transition relation $\tau \subseteq \Sigma \times \Delta \times \Sigma$, in place of the classic one. The meaning of $\langle \sigma, \delta, \sigma' \rangle \in \tau$ is that the system can transition from state σ to state σ' by executing the action δ. A transition system can be viewed as a labeled transition system where $\Delta \triangleq \{\star\}$ is a singleton. In this case, every transition relation in $\Sigma \times \Sigma$ is isomorphic to a labeled one in $\Sigma \times \{\star\} \times \Sigma$.

Let $\mathfrak{D}$ be the set of possible denotations of systems states. Then, given a system $\mathbb{s}$, $\Sigma^{\mathbb{s}} \subseteq \mathfrak{D}$ is the state space of $\mathbb{s}$ and $\Upsilon^{\mathbb{s}}$, $\Omega^{\mathbb{s}}$, $\tau^{\mathbb{s}}$ are the other components of the transition system. Very often, for the sake of simplicity, we omit the superscript $^{\mathbb{s}}$ in the transition system. In the following, we assume to know the transition system associated with a system $\mathbb{s}$. In Section 4.1.3 we will show how to retrieve the transition system associated to a given system (where the system is a program in a deterministic programming language).

4.1.1.1 System Semantics

A transition system is a mere static mathematical description of a system. Information about its behavior, i.e. semantic information, emerges when considering sequences of transitions. Before we define the trace semantics, which expresses the most information about a system, we have to introduce some notation about sequences.

Sequences. Given a set S, the set $S^{\vec{n}} \triangleq [0, n[\to S$, $n \in \mathbb{N}$, is the set of finite sequences[1] $\bar{s} = \bar{s}_0 \bar{s}_1 \dots \bar{s}_{n-1}$ of length $|\bar{s}| = n$ over S. We denote ϵ the empty sequence, so $S^{\vec{0}} \triangleq \{\epsilon\}$. The set of finite non-empty sequences is $S^{\vec{+}} \triangleq \bigcup_{0<n<\omega} S^{\vec{n}}$. The set $S^{\vec{\omega}} \triangleq \mathbb{N} \to S$ contains infinite sequences $\bar{s} = \bar{s}_0 \bar{s}_1 \dots$ of length $|s| = \omega$ over S. The set of non-empty sequences is $S^{\vec{\infty}} \triangleq S^{\vec{+}} \cup S^{\vec{\omega}}$. We denote with $S_\epsilon^{\vec{+}}$ and $S_\epsilon^{\vec{\infty}}$ the sets $S^{\vec{+}} \cup S^{\vec{0}}$ and $S^{\vec{\infty}} \cup S^{\vec{0}}$, respectively.

Given sequences $\bar{s}, \bar{s}' \in S^{\vec{\infty}}$, their *concatenation* is $\bar{s}\bar{s}'$ when $\bar{s} \in S^{\vec{+}}$ and it is just $\bar{s}$ when $\bar{s} \in S^{\vec{\omega}}$. The empty sequence is the identity element for concatenation, namely $\bar{s}\epsilon = \epsilon\bar{s} = \bar{s}$. Given $\bar{s}, \bar{s}' \in S^{\vec{\infty}}$, $\bar{s}'$ can be appended to $\bar{s}$ when $\bar{s}_{|\bar{s}|-1} = \bar{s}'_0$ (hence $\bar{s}$ is finite) and their *append* is $\bar{s} \frown \bar{s}' \triangleq \bar{s}_0 \bar{s}_1 \dots \bar{s}_{|\bar{s}|-1} \bar{s}'_1 \bar{s}'_2 \dots \bar{s}'_{|\bar{s}'|-1}$ ($|\bar{s}'|$ could be ω). In the case of $|\bar{s}| = \omega$, we define the append as $\bar{s} \frown \bar{s}' \triangleq \bar{s}$.

Finally, given $X \subseteq S^{\vec{\infty}}$, we denote with $X^{\vec{+}}$ its selection of finite non-empty sequences, namely $X^{\vec{+}} \triangleq X \cap S^{\vec{+}}$, and with $X^{\vec{\omega}}$ its selection of infinite non-empty sequences, namely $X^{\vec{\omega}} \triangleq X \cap S^{\vec{\omega}}$. Concatenation and append lift to sets as follow: for every $X, Y \subseteq S^{\vec{\infty}}$, their (set) concatenation is $XY \triangleq \{\bar{s}\bar{s}' \mid \bar{s} \in X \wedge \bar{s}' \in Y\}$ and their (set) append is $X \frown Y \triangleq \{\bar{s} \frown \bar{s}' \mid \bar{s} \in X \wedge \bar{s}' \in Y\}$.

Trace Semantics. An execution, called *trace*, of a system $\mathbb{s}$, whose associated transition system is $\langle \Sigma, \tau, \Upsilon, \Omega \rangle$, is a sequence of states in Σ where adjacent elements are in τ. We denote with $\tau^{\vec{n}} \triangleq \{\bar{\sigma} \in \Sigma^{\vec{n}} \mid \bar{\sigma}_{n-1} \in \Omega \wedge \forall i \in [0, n-1[\,.\, \bar{\sigma}_i \ \tau \ \bar{\sigma}_{i+1}\}$ the *finite final* traces of length n of $\mathbb{s}$ and with $\tau^{\vec{\omega}} \triangleq \{\bar{\sigma} \in \Sigma^{\vec{\omega}} \mid \forall i \in \mathbb{N} \,.\, \bar{\sigma}_i \ \tau \ \bar{\sigma}_{i+1}\}$ the *infinite* traces of $\mathbb{s}$. So we can

[1] A sequence is isomorphic to a function whose domain is a subset of natural numbers with cardinality equal to length of the sequence and whose co-domain is the set of symbols of the sequence. For instance, the sequence abc is isomorphic to the function $[0 \mapsto a \ 1 \mapsto b \ 2 \mapsto c]$.

define the most precise semantics of $\mathbb{s}$, namely the one which gives us the most information about the behavior of $\mathbb{s}$.

Definition 21 (Maximal Trace Semantics)**.** The *maximal trace semantics* $\tau^{\vec{\infty}} \in \wp(\Sigma^{\vec{\infty}})$ of $\mathbb{s}$, which is associated with the transition system $\langle \Sigma, \tau, \Upsilon, \Omega \rangle$, is:

$$\tau^{\vec{\infty}} \triangleq \tau^{\vec{+}} \cup \tau^{\vec{\omega}}$$

where $\tau^{\vec{+}} \triangleq \bigcup_{0<n<\omega} \tau^{\vec{n}}$ is called the *maximal finite trace semantics* and $\tau^{\vec{\omega}}$ is called the *infinite trace semantics*.

The maximal trace semantics models the terminating executions of the system, namely those which end in a final state (the ones in $\tau^{\vec{+}}$) and the divergent executions, namely those which do not end (the ones in $\tau^{\vec{\omega}}$). In this sense the term 'maximal' is used: the longest finite sequences, which are the terminating ones, or the infinite sequences, which are longer than any finite sequence.

The maximal trace semantics is constructive, in the sense of Definition 5. Its computational fixpoint definition is $\langle F^{\vec{\infty}}, \mathcal{D}^{\vec{\infty}}, \bot^{\vec{\infty}} \rangle$, where $F^{\vec{\infty}} \in \wp(\Sigma^{\vec{\infty}}) \to \wp(\Sigma^{\vec{\infty}})$, $\bot^{\vec{\infty}}$ and $\mathcal{D}^{\vec{\infty}} \triangleq \langle \wp(\Sigma^{\vec{\infty}}), \sqsubseteq, \sqcup \rangle$ are[2]:

- $F^{\vec{\infty}} \triangleq \lambda X \,.\, \tau^{\vec{1}} \cup \tau^{\dot{\vec{2}}} \frown X$
- $\bot^{\vec{\infty}} \triangleq \Sigma^{\vec{\omega}}$
- $X \sqsubseteq Y \triangleq X^{\vec{+}} \subseteq Y^{\vec{+}} \wedge Y^{\vec{\omega}} \subseteq X^{\vec{\omega}}$, for every $X, Y \subseteq \Sigma^{\vec{\infty}}$
- $\bigsqcup \mathcal{X} \triangleq \bigcup_{X \in \mathcal{X}} X^{\vec{+}} \cup \bigcap_{X \in \mathcal{X}} X^{\vec{\omega}}$, for every $\mathcal{X} \subseteq \wp(\Sigma^{\vec{\infty}})$

Here, $\tau^{\dot{\vec{n}}} \triangleq \{s \in \Sigma^{\vec{n}} \mid \forall i \in [0, n-1[\,.\, s_i \;\tau\; s_{i+1}\}$ denotes the finite traces of length n (not necessarily terminating)[3]. The maximal trace semantics is the least fixpoint, greater than $\Sigma^{\vec{\omega}}$, of $F^{\vec{\infty}}$, namely:

$$\tau^{\vec{\infty}} = \mathsf{lfp}^{\sqsubseteq}_{\bot^{\vec{\infty}}} F^{\vec{\infty}}$$

To reason about particular systems behaviors, it is not necessary to consider all aspects of systems executions. In fact, reasoning is facilitated by the design of simpler semantics, abstracting away from irrelevant information. Therefore, there are a wide variety of systems semantics, each one of them dedicated to a particular verification task. Abstract interpretation is a method for relating these semantics, as shown in [Cousot, 2002].

4.1.2 Hierarchy of Semantics

In [Cousot, 2002] the author showed that many well-known (program) semantics can be computed as abstract interpretations of the maximal trace semantics, and they can be organized in a hierarchy. At first glance, one could remain a little bit disoriented, since abstract interpretation, as introduced in Chapter 2, needs a notion of approximation between objects (here semantics). But, what does it mean that the semantics of a system is an approximation

[2] Actually, $\langle \wp(\Sigma^{\vec{\infty}}), \sqsubseteq \rangle$ is a complete lattice and $\sqcup$ is total.

[3] In particular, $\tau^{\dot{\vec{2}}}$ coincides with τ.

of another semantics? Indeed, what is left implicit in [Cousot, 2002] is that the approximation regards the collecting semantics of systems, which is actually a specification (we will explain better the concept of collecting semantics in Chapter 5). The author, and in general the field of comparative semantics, is mainly interested in trace properties, namely specifications asserting properties of executions rather than properties of systems. Hence, in this case, semantics, collecting semantics and trace properties lie all in the same domain. In the following we make explicit the steps left implicit in [Cousot, 2002].

The most concrete semantics of the hierarchy is $\tau^{\vec{\infty}}$ and, indeed, systems semantics are in $\wp(\Sigma^{\vec{\infty}})$. This latter is the systems representation domain $\mathbb{REP}^{\mathsf{Sys}}$ of Chapter 3. Specifications are in $\mathsf{Specs} \triangleq \wp(\wp(\Sigma^{\vec{\infty}}))$ and the strongest system specification is $\{\tau^{\vec{\infty}}\}$, which is the collecting semantics. As usual, the implication between specifications is set-inclusion, namely $\mathbb{Q}_1 \subseteq \mathbb{Q}_2$ means that all systems satisfying $\mathbb{Q}_1$ also satisfy $\mathbb{Q}_2$. Since we are interested in trace properties, hence specifications on systems traces, we can apply the following abstraction $\alpha_\cup \triangleq \lambda\mathcal{X}\,.\,\bigcup\mathcal{X}$, obtaining the new set of specifications $\mathsf{Specs}' \triangleq \{\alpha_\cup(\mathcal{X}) \mid \mathcal{X} \in \mathsf{Specs}\} = \wp(\Sigma^{\vec{\infty}})$. The soundness function is $\gamma_\wp \triangleq \lambda X\,.\,\wp(X)$, meaning that $\mathbb{Q}' \in \mathsf{Specs}'$ is a correct abstraction of $\mathbb{Q} \in \mathsf{Specs}$ if $\mathbb{Q} \subseteq \wp(\mathbb{Q}')$. Abstracting the collecting semantics on Specs into the collecting semantics on Specs', we obtain exactly the standard semantics of the system, namely, $\tau^{\vec{\infty}} = \alpha_\cup(\{\tau^{\vec{\infty}}\})$. In this setting, the approximation order $\subseteq$ in Specs' has a precise meaning: if $\tau^{\vec{\infty}}$ implies (i.e. it is contained in) a specification in Specs', then all trace properties implied by the specification are implied by $\tau^{\vec{\infty}}$ as well. In other words, the relation $\subseteq$ in $\langle \mathsf{Specs}', \subseteq\rangle$, domain of the standard semantics, is the satisfiability relation. With this in mind, we can answer the question mentioned at the beginning: a semantics is an approximation of another semantics if the first satisfies more trace properties than the second.

The semantics of the hierarchy are all constructive objects, namely they can be computed by fixpoint of a function over an ordered domain. It is not always possible to obtain semantics by fixpoint w.r.t. the standard inclusion order ($\subseteq$), the approximation order. In fact, in some cases the fixpoint operator is not $\subseteq$-monotone, and therefore we have to define a computational order forcing monotonicity, and therefore convergence of the fixpoint computation (this is indeed the case of maximal trace semantics). In this subsection we recall the original hierarchy of [Cousot, 2002], introducing the most important elements. In all the following cases, the computational fixpoint definition is built upon a complete lattice and the partial least upper bound operator is indeed total.

The maximal trace semantics is defined as the union of the maximal finite trace semantics and the infinite trace semantics. Both these latter are constructive. The computational fixpoint definition for the maximal finite trace semantics $\tau^{\vec{+}}$ is $\langle F^{\vec{+}}, \mathcal{D}^{\vec{+}}, \varnothing\rangle$, where $F^{\vec{+}} \in \wp(\Sigma^{\vec{+}}) \to \wp(\Sigma^{\vec{+}})$ is $F^{\vec{+}} \triangleq \lambda X\,.\,\tau^{\vec{1}} \cup \tau^{\dot{\vec{2}}} \frown X$ and $\mathcal{D}^{\vec{+}} \triangleq \langle \wp(\Sigma^{\vec{+}}), \subseteq, \cup\rangle$. The finite trace semantics is the least fixpoint, greater than $\varnothing$, of $F^{\vec{+}}$, namely:

$$\tau^{\vec{+}} = \mathsf{lfp}^{\subseteq}_{\varnothing} F^{\vec{+}}$$

The computational fixpoint definition for the infinite trace semantics $\tau^{\vec{\omega}}$ is $\langle F^{\vec{\omega}}, \mathcal{D}^{\vec{\omega}}, \Sigma^{\vec{\omega}}\rangle$, where $F^{\vec{\omega}} \in \wp(\Sigma^{\vec{\omega}}) \to \wp(\Sigma^{\vec{\omega}})$ is $F^{\vec{\omega}} \triangleq \lambda X\,.\,\tau^{\dot{\vec{2}}} \frown X$ and $\mathcal{D}^{\vec{\omega}} \triangleq \langle \wp(\Sigma^{\vec{\omega}}), \supseteq, \cap\rangle$. The infinite trace semantics is the least fixpoint, greater than $\Sigma^{\vec{\omega}}$, of $F^{\vec{\omega}}$, namely:

$$\tau^{\vec{\omega}} = \mathsf{lfp}^{\supseteq}_{\Sigma^{\vec{\omega}}} F^{\vec{\omega}}$$

Remark. We already said that the maximal trace semantics is the composition (or fusion, using [Cousot, 2002] terminology) of the maximal finite and infinite trace semantics, namely:

$$\tau^{\vec{\infty}} = \tau^{\vec{+}} \cup \tau^{\vec{\omega}} = \mathsf{lfp}^{\subseteq}_{\varnothing} F^{\vec{+}} \cup \mathsf{lfp}^{\supseteq}_{\Sigma^{\vec{\omega}}} F^{\vec{\omega}} = \mathsf{lfp}^{\sqsubseteq}_{\Sigma^{\vec{\omega}}} F^{\vec{\infty}}$$

But the composition is also at the level of the semantic operator (given $X \in \wp(\Sigma^{\vec{\infty}})$):

$$\begin{aligned} F^{\vec{+}}(X^{\vec{+}}) \cup F^{\vec{\omega}}(X^{\vec{\omega}}) &= (\tau^{\vec{1}} \cup \tau^{\dot{\vec{2}}} \frown X^{\vec{+}}) \cup (\tau^{\dot{\vec{2}}} \frown X^{\vec{\omega}}) = \\ &= \tau^{\vec{1}} \cup \tau^{\dot{\vec{2}}} \frown (X^{\vec{+}} \cup X^{\vec{\omega}}) = \tau^{\vec{1}} \cup \tau^{\dot{\vec{2}}} \frown X = F^{\vec{\infty}} \end{aligned}$$

It is trivial to note that the finite maximal and infinite trace semantics are abstractions of the maximal trace semantics: in the first we rule out infinite traces whilst in the second we rule out finite traces. Consider the functions $\alpha^{\vec{+}} \in \wp(\Sigma^{\vec{\infty}}) \to \wp(\Sigma^{\vec{+}})$ and $\alpha^{\vec{\omega}} \in \wp(\Sigma^{\vec{\infty}}) \to \wp(\Sigma^{\vec{\omega}})$, defined as $\alpha^{\vec{+}} \triangleq \lambda X \,.\, X^{\vec{+}}$ and $\alpha^{\vec{\omega}} \triangleq \lambda X \,.\, X^{\vec{\omega}}$. Then we have that $\tau^{\vec{+}} = \alpha^{\vec{+}}(\tau^{\vec{\infty}})$ and $\tau^{\vec{\omega}} = \alpha^{\vec{\omega}}(\tau^{\vec{\infty}})$. The abstraction is formalized by the Galois insertions:

$$\langle \wp(\Sigma^{\vec{\infty}}), \subseteq \rangle \underset{\alpha^{\vec{+}}}{\overset{\gamma^{\vec{+}}}{\leftrightarrows}} \langle \wp(\Sigma^{\vec{+}}), \subseteq \rangle \quad \langle \wp(\Sigma^{\vec{\infty}}), \subseteq \rangle \underset{\alpha^{\vec{\omega}}}{\overset{\gamma^{\vec{\omega}}}{\leftrightarrows}} \langle \wp(\Sigma^{\vec{\omega}}), \subseteq \rangle$$

$$\text{with } \gamma^{\vec{+}} \triangleq \lambda X \,.\, X \cup \Sigma^{\vec{\omega}} \text{ and } \gamma^{\vec{\omega}} \triangleq \lambda X \,.\, X \cup \Sigma^{\vec{+}}$$

Another important semantics is the maximal relational semantics τ^{∞} (called natural in [Cousot, 2002]). This latter associates, with program traces, an input-output relation by using a special symbol, $\circlearrowleft \notin \Sigma$, to denote non-termination. Also in this case, we have a semantics representing only finite executions, called finite relational semantics τ^{+}, and a semantics representing only infinite executions, called infinite relational semantics τ^{ω}. The finite relational semantics corresponds to an abstraction of the maximal finite trace semantics, where intermediate computation states are ignored. Consider the function $\alpha^{+} \in \wp(\Sigma^{\vec{+}}) \to \wp(\Sigma \times \Sigma)$, defined as $\alpha^{+} \triangleq \lambda X \,.\, \{\langle \bar{\sigma}_0, \bar{\sigma}_{n-1} \rangle \mid \bar{\sigma} \in X \cap \Sigma^{\vec{n}}\}$. Then we have that $\tau^{+} = \alpha^{+}(\tau^{\vec{+}})$. Similarly, the infinite relational semantics corresponds to an abstraction of the infinite trace semantics, where intermediate computation states are ignored. Consider the function $\alpha^{\omega} \in \wp(\Sigma^{\vec{\omega}}) \to \wp(\Sigma \times \Sigma_{\circlearrowleft})$, where $\Sigma_{\circlearrowleft} \triangleq \Sigma \cup \{\circlearrowleft\}$, defined as $\alpha^{\omega} \triangleq \lambda X \,.\, \{\langle \bar{\sigma}_0, \circlearrowleft \rangle \mid \bar{\sigma} \in X\}$. Then we have that $\tau^{\omega} = \alpha^{\omega}(\tau^{\vec{\omega}})$. The abstraction is formalized by the Galois insertions:

$$\langle \wp(\Sigma^{\vec{+}}), \subseteq \rangle \underset{\alpha^{+}}{\overset{\gamma^{+}}{\leftrightarrows}} \langle \wp(\Sigma \times \Sigma), \subseteq \rangle \quad \langle \wp(\Sigma^{\vec{\omega}}), \subseteq \rangle \underset{\alpha^{\omega}}{\overset{\gamma^{\omega}}{\leftrightarrows}} \langle \wp(\Sigma \times \{\circlearrowleft\}), \subseteq \rangle$$

$$\text{with } \gamma^{+} \triangleq \lambda X \,.\, \{\bar{\sigma} \in \Sigma^{\vec{n}} \mid \langle \bar{\sigma}_0, \bar{\sigma}_{n-1} \rangle \in X\} \text{ and } \gamma^{\omega} \triangleq \lambda X \,.\, \{\bar{\sigma} \in \Sigma^{\vec{\omega}} \mid \langle \bar{\sigma}_0, \circlearrowleft \rangle \in X\}$$

The computational fixpoint definition for the finite relational semantics is $\langle F^{+}, \mathcal{D}^{+}, \varnothing \rangle$, with $F^{+} \in \wp(\Sigma \times \Sigma) \to \wp(\Sigma \times \Sigma)$ defined as $F^{+} \triangleq \lambda X \,.\, \{\langle \sigma, \sigma \rangle \mid \sigma \in \tau^{\vec{1}}\} \cup \tau \circ X$ (note that here X is a relation) and $\mathcal{D}^{+} \triangleq \langle \wp(\Sigma \times \Sigma), \subseteq, \cup \rangle$. The finite trace semantics is the least fixpoint, greater than $\varnothing$, of F^{+}, namely:

$$\tau^{+} = \mathsf{lfp}^{\subseteq}_{\varnothing} F^{+}$$

The computational fixpoint definition for the infinite relational semantics is $\langle F^{\omega}, \mathcal{D}^{\omega}, \bot^{\vec{\omega}} \rangle$, where $F^{\omega} \in \wp(\Sigma \times \{\circlearrowleft\}) \to \wp(\Sigma \times \{\circlearrowleft\})$ is $F^{\omega} \triangleq \lambda X \,.\, \tau \circ X$ (again, here X is a relation),

$\mathcal{D}^{\omega} \triangleq \langle \wp(\Sigma \times \{\circlearrowleft\}), \supseteq, \cap \rangle$ and $\bot^{\omega} \triangleq \Sigma \times \{\circlearrowleft\}$. The infinite relational semantics is the least fixpoint, greater than $\bot^{\omega}$, of F^{ω}, namely:

$$\tau^{\omega} = \mathsf{lfp}^{\supseteq}_{\bot^{\omega}} F^{\omega}$$

Finally, the maximal relational semantics corresponds to an abstraction of the maximal trace semantics, where intermediate computation states are ignored. Consider the function $\alpha^{\infty} \in \wp(\Sigma^{\vec{\infty}}) \to \wp(\Sigma \times \Sigma_{\circlearrowleft})$, defined as $\alpha^{\infty} \triangleq \lambda X \,.\, \alpha^{+}(X^{\vec{+}}) \cup \alpha^{\omega}(X^{\vec{\omega}})$. Then we have $\tau^{\infty} = \alpha^{\infty}(\tau^{\vec{\infty}})$. The abstraction is formalized by the Galois insertion:

$$\langle \wp(\Sigma^{\vec{\infty}}), \subseteq \rangle \underset{\alpha^{\infty}}{\overset{\gamma^{\infty}}{\leftrightarrows}} \langle \wp(\Sigma \times \Sigma_{\circlearrowleft}), \subseteq \rangle \text{ with } \gamma^{\infty} \triangleq \lambda X \,.\, \gamma^{+}(X \cap \Sigma{\times}\Sigma) \cup \gamma^{\omega}(X \cap \bot^{\infty})$$

The computational fixpoint definition for the maximal relational semantics is $\langle F^{\infty}, \mathcal{D}^{\infty}, \bot^{\infty} \rangle$, where $F^{\infty} \in \wp(\Sigma \times \Sigma_{\circlearrowleft}) \to \wp(\Sigma \times \Sigma_{\circlearrowleft})$, $\mathcal{D}^{\infty} \triangleq \langle \wp(\Sigma \times \Sigma_{\circlearrowleft}), \preccurlyeq, \curlyvee \rangle$ and $\bot^{\infty}$ are:

- $F^{\infty} \triangleq \lambda X \,.\, \{\langle \sigma, \sigma \rangle \mid \sigma \in \tau^{\vec{1}}\} \cup \tau \circ X$
- $\bot^{\infty} \triangleq \Sigma \times \{\circlearrowleft\}$
- $X \preccurlyeq Y \triangleq (X \cap \Sigma{\times}\Sigma) \subseteq (Y \cap \Sigma{\times}\Sigma) \wedge (X \cap \bot^{\infty}) \supseteq (Y \cap \bot^{\infty})$, for every $X, Y \subseteq \Sigma{\times}\Sigma_{\circlearrowleft}$
- $\curlyvee \mathcal{X} \triangleq \bigcup_{X \in \mathcal{X}} (X \cap \Sigma{\times}\Sigma) \cup \bigcap_{X \in \mathcal{X}} (X \cap \bot^{\infty})$, for every $\mathcal{X} \subseteq \wp(\Sigma{\times}\Sigma_{\circlearrowleft})$

The maximal relational semantics is the least fixpoint, greater than $\bot^{\infty}$, of F^{∞}, namely:

$$\tau^{\infty} = \mathsf{lfp}^{\preccurlyeq}_{\bot^{\infty}} F^{\infty}$$

Also in this case, it is trivial to note that the finite and infinite relational semantics are abstractions of the maximal relational semantics. Consider the functions $\alpha^{+}_{\bullet} \in \wp(\Sigma{\times}\Sigma_{\circlearrowleft}) \to \wp(\Sigma \times \Sigma)$ and $\alpha^{\omega}_{\bullet} \in \wp(\Sigma \times \Sigma_{\circlearrowleft}) \to \wp(\Sigma \times \{\circlearrowleft\})$, defined as $\alpha^{+}_{\bullet} \triangleq \lambda X \,.\, X \cap \Sigma \times \Sigma$ and $\alpha^{\omega}_{\bullet} \triangleq \lambda X \,.\, X \cap \Sigma \times \{\circlearrowleft\}$. Then we have that $\tau^{+} = \alpha^{+}_{\bullet}(\tau^{\infty})$ and $\tau^{\omega} = \alpha^{\omega}_{\bullet}(\tau^{\infty})$. The abstraction is formalized by the Galois insertions:

$$\langle \wp(\Sigma \times \Sigma_{\circlearrowleft}), \subseteq \rangle \underset{\alpha^{+}_{\bullet}}{\overset{\gamma^{+}_{\bullet}}{\leftrightarrows}} \langle \wp(\Sigma \times \Sigma), \subseteq \rangle \quad \langle \wp(\Sigma \times \Sigma_{\circlearrowleft}), \subseteq \rangle \underset{\alpha^{\omega}_{\bullet}}{\overset{\gamma^{\omega}_{\bullet}}{\leftrightarrows}} \langle \wp(\Sigma \times \{\circlearrowleft\}), \subseteq \rangle$$
$$\text{with } \gamma^{+}_{\bullet} \triangleq \lambda X \,.\, X \cup \Sigma \times \{\circlearrowleft\} \text{ and } \gamma^{\omega}_{\bullet} \triangleq \lambda X \,.\, X \cup \Sigma \times \Sigma$$

In [Cousot, 2002], the author showed that the semantic operators computing all the previous semantics can be obtained by fixpoint transfer, starting from the maximal trace semantics. This is obtained due to the fact that all abstractions function are additive also into the corresponding computational domain (hence we have Galois insertions also between all computational domains). In Figure 4.1 we have a graphical representation of (a part of) the hierarchy, where an arrow from a semantics to another means that the second is an abstraction of the first.

4.1.2.1 Extending the Hierarchy

The original hierarchy contains a lot of semantics, but it lacks some important examples of semantics used in static analysis. For instance, in order to capture safety trace properties, on finite executions, the *partial trace semantics* (called prefix semantics in [Cousot and Cousot, 2012]) is the best choice. This latter is $\tau^{\vec{\propto}} \triangleq \bigcup_{0<n<\omega}\{\bar{\sigma} \in \tau^{\dot{n}} \mid \bar{\sigma}_0 \in \Upsilon\}$ and it models the finite prefixes of executions (not necessarily terminating), starting from the initial states.

The partial trace semantics $\tau^{\vec{\propto}}$ is constructive. Its computational fixpoint definition is $\langle F^{\vec{\propto}}, \mathcal{D}^{\vec{\propto}}, \varnothing\rangle$, where $F^{\vec{\propto}} \in \wp(\Sigma^{\vec{+}}) \to \wp(\Sigma^{\vec{+}})$ is defined as $F^{\vec{\propto}} \triangleq \lambda X\,.\,\Upsilon \cup X \frown \tau^{\dot{2}}$ and $\mathcal{D}^{\vec{\propto}} \triangleq \langle \wp(\Sigma^{\vec{\propto}}), \subseteq, \cup\rangle$. The partial trace semantics is the least fixpoint, greater than $\varnothing$, of $F^{\vec{\propto}}$, namely:

$$\tau^{\vec{\propto}} = \mathsf{lfp}^{\subseteq}_{\varnothing} F^{\vec{\propto}}$$

Remark. Usually, the partial trace semantics can be defined starting from an arbitrary set of initial states $I \subseteq \Upsilon$, namely we are interested in the semantics $\tau^{\vec{\propto}} \cap \{\bar{\sigma} \in \Sigma^{\vec{+}} \mid \bar{\sigma}_0 \in I\}$. This latter can be easily computed substituting Υ with I in the definition of the operator $F^{\vec{\propto}}$.

The partial trace semantics corresponds to an abstraction of the maximal trace semantics, adding all the prefixes of finite and infinite computations. Consider the function $\alpha^{\vec{\propto}} \in \wp(\Sigma^{\vec{\infty}}) \to \wp(\Sigma^{\vec{+}})$ defined as $\alpha^{\vec{\propto}} \triangleq \lambda X\,.\,\{\bar{\sigma} \in \Sigma^{\dot{n}} \mid 0 < n < \omega \wedge \exists \bar{\sigma}' \in \Sigma^{\vec{\infty}}_{\epsilon}\,.\,(\bar{\sigma}'_0 \in \Upsilon \wedge \bar{\sigma}\bar{\sigma}' \in X)\}$. Then we have $\tau^{\vec{\propto}} = \alpha^{\vec{\propto}}(\tau^{\vec{\infty}})$. The abstraction is formalized by the Galois insertion:

$$\langle \wp(\Sigma^{\vec{\infty}}), \subseteq\rangle \underset{\alpha^{\vec{\propto}}}{\overset{\gamma^{\vec{\propto}}}{\leftrightarrows}} \langle \wp(\Sigma^{\vec{\propto}}), \subseteq\rangle$$
$$\text{with } \gamma^{\vec{\propto}} \triangleq \lambda X\,.\,\{\bar{\sigma} \in \Sigma^{\dot{n}} \mid 0 < n < \omega \wedge \exists \bar{\sigma}' \in \Sigma^{\vec{+}}_{\epsilon}\,.\,\bar{\sigma}'\bar{\sigma} \in X\}$$

As done in the classic hierarchy, we can abstract traces to input/output pairs of states, obtaining the *partial relational semantics* (called relational invariance/reachability semantics in [Cousot and Cousot, 2012]) $\tau^{\propto} \triangleq \{\langle \bar{\sigma}_0, \bar{\sigma}_{n-1}\rangle \mid 0 < n < \omega \wedge \exists \bar{\sigma} \in \Sigma^{\dot{n}}\,.\,\bar{\sigma}_0 \in \Upsilon\}$. This semantics provides a relation between an initial state and a state reached during the execution of the system (current state).

The partial relational semantics corresponds to an abstraction of the partial trace semantics, where intermediate computation states are ignored. Consider the function $\alpha^{\propto} \in \wp(\Sigma^{\vec{\propto}}) \to \wp(\Sigma \times \Sigma)$ defined as $\alpha^{\propto} \triangleq \lambda X\,.\,\{\langle \bar{\sigma}_0, \bar{\sigma}_{n-1}\rangle \mid 0 < n < \omega \wedge \bar{\sigma} \in \Sigma^{\dot{n}} \cap X\}$. Then we have $\tau^{\propto} = \alpha^{\propto}(\tau^{\vec{\propto}})$. The abstraction is formalized by the Galois insertion:

$$\langle \wp(\Sigma^{\vec{\propto}}), \subseteq\rangle \underset{\alpha^{\propto}}{\overset{\gamma^{\propto}}{\leftrightarrows}} \langle \wp(\Sigma \times \Sigma), \subseteq\rangle \text{ with } \gamma^{\propto} \triangleq \lambda X\,.\,\{\bar{\sigma} \in \Sigma^{\dot{n}} \mid \langle \bar{\sigma}_0, \bar{\sigma}_{n-1}\rangle \in X\}$$

The partial relational semantics $\tau^{\propto}$ is constructive. Its computational fixpoint definition is $\langle F^{\propto}, \mathcal{D}^{\propto}, \varnothing\rangle$, where $F^{\propto} \in \wp(\Sigma \times \Sigma) \to \wp(\Sigma \times \Sigma)$ is $F^{\propto} \triangleq \lambda X\,.\,\{\langle \sigma, \sigma\rangle \mid \sigma \in \Upsilon\} \cup X \circ \tau$ and $\mathcal{D}^{\propto} \triangleq \langle \wp(\Sigma \times \Sigma), \subseteq, \cup\rangle$. The partial relational semantics is the least fixpoint, greater than $\varnothing$, of $F^{\propto}$, namely:

$$\tau^{\propto} = \mathsf{lfp}^{\subseteq}_{\varnothing} F^{\propto}$$

Remark. The partial trace and partial relational semantics described here are slightly different from the ones presented in [Cousot and Cousot, 2012]. Indeed, our semantics are prefix-closed but not suffix-closed, as happens for the cited semantics. This because the semantics of [Cousot and Cousot, 2012] start from a semantics different from the maximal

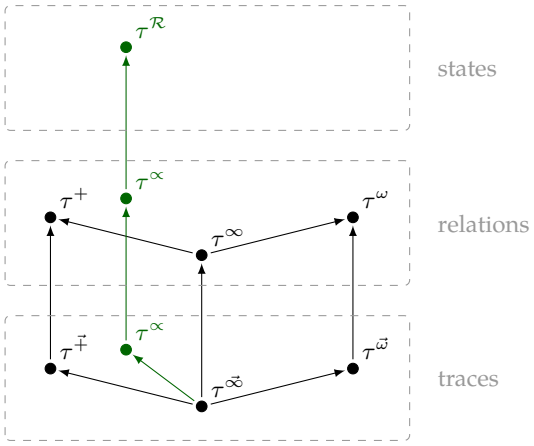


Figure 4.1: A part of the standard hierarchy of semantics, with extensions (in green).

trace semantics. Nevertheless, this semantics is isomorphic to the maximal trace semantics and our partial semantics are isomorphic to the one proposed in [Cousot and Cousot, 2012].

Finally, the last important semantics we want to introduce is the *state semantics* (called invariance/reachability semantics in [Cousot and Cousot, 2012]). This latter collects all states encountered during the execution of the system and it is useful to compute invariants. It is defined as $\tau^{\mathcal{R}} \triangleq \Upsilon \cup \{\sigma \in \Sigma \mid \exists n \in \mathbb{N} \setminus \{0\}\, \exists \bar{\sigma} \in \Sigma^{\dot{n}} \,.\, \bar{\sigma}_0 \in \Upsilon \wedge \bar{\sigma}_{n-1} = \sigma\}$.

The state semantics corresponds to an abstraction of the partial relational semantics, collecting only the last states. Consider the function $\alpha^{\mathcal{R}} \in \wp(\Sigma^{\propto}) \to \wp(\Sigma)$ defined as $\alpha^{\mathcal{R}} \triangleq \lambda X \,.\, \{\sigma' \mid \langle \sigma, \sigma' \rangle \in X\}$. Then we have $\tau^{\mathcal{R}} = \alpha^{\mathcal{R}}(\tau^{\propto})$. The abstraction is formalized by the Galois insertion:

$$\langle \wp(\Sigma \times \Sigma), \subseteq \rangle \underset{\alpha^{\mathcal{R}}}{\overset{\gamma^{\mathcal{R}}}{\leftrightarrows}} \langle \wp(\Sigma), \subseteq \rangle \text{ with } \gamma^{\mathcal{R}} \triangleq \lambda X \,.\, \{\langle \sigma, \sigma' \rangle \in \Sigma \times \Sigma \mid \sigma' \in X\}$$

As usual, the state semantics $\tau^{\mathcal{R}}$ is constructive. Its computational fixpoint definition is $\langle F^{\mathcal{R}}, \mathcal{D}^{\mathcal{R}}, \varnothing \rangle$, where $F^{\mathcal{R}} \in \wp(\Sigma) \to \wp(\Sigma)$ is $F^{\mathcal{R}} \triangleq \lambda X \,.\, \Upsilon \cup \{\sigma' \mid \exists \sigma \in X \,.\, \sigma \;\tau\; \sigma'\}$ and $\mathcal{D}^{\mathcal{R}} \triangleq \langle \wp(\Sigma), \subseteq, \cup \rangle$. The state semantics is the least fixpoint, greater than $\varnothing$, of $F^{\mathcal{R}}$, namely:

$$\tau^{\mathcal{R}} = \mathsf{lfp}^{\subseteq}_{\varnothing} F^{\mathcal{R}}$$

All the semantic operators computing the introduced semantics can be obtained by fixpoint transfer, starting from the maximal trace semantics. Again, this is obtained due to the fact that all abstraction function are additive also into the corresponding computational domain (hence we have Galois insertions also between all computational domains). In Figure 4.1 we have a graphical representation of the extended hierarchy, where an arrow from a semantics to another means that the second is an abstraction of the first.

4.1.3 The Programming Language

The systems we will take in consideration are computer programs. These latter are written in a simple deterministic imperative language Imp, whose grammar is the following:

$$\begin{aligned}
\mathsf{Aexp} \ni \mathsf{a} &::= \mathsf{x} \mid \mathsf{n} \mid \mathsf{a} + \mathsf{a} \mid \mathsf{a} - \mathsf{a} \mid \mathsf{a} * \mathsf{a} \mid (\mathsf{a}) \\
\mathsf{Bexp} \ni \mathsf{b} &::= \mathsf{tt} \mid \mathsf{ff} \mid \mathsf{a} = \mathsf{a} \mid \mathsf{a} \neq \mathsf{a} \mid \mathsf{a} < \mathsf{a} \mid \mathsf{a} \leq \mathsf{a} \mid \mathsf{b} \wedge \mathsf{b} \mid \mathsf{b} \vee \mathsf{b} \mid \neg \mathsf{b} \mid (\mathsf{b}) \\
\mathsf{Com} \ni \mathsf{c} &::= \mathsf{skip} \mid \mathsf{x} := \mathsf{a} \mid \mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\,\mathsf{P}\,\} \mid \mathsf{while}\ {}^{\mathtt{k}}\mathsf{b}\ \{\,\mathsf{P}\,\} \mid \mathsf{c}^{\mathtt{k}}\,.\,{}^{\mathtt{k}}\mathsf{c} \\
\mathsf{Imp} \ni \mathsf{P} &::= {}^{\mathtt{k}}\mathsf{c}^{\mathtt{k}}
\end{aligned}$$

We suppose to have an infinite supply of variables $\mathsf{x} \in \mathsf{Var}$ and labels $\mathtt{k} \in \mathsf{Lab}$. A program in Imp is just a command in Com attached with an initial and a final label. If $\mathsf{P} = {}^{\mathtt{i}}\mathsf{c}^{\mathtt{f}}$ then we call $\mathtt{i}$, i.e. the initial label, its entry-point and we call $\mathtt{f}$, i.e. the final label, its exit-point. Commands comprise a concatenation construct $\mathsf{c}^{\mathtt{l}}\,.\,{}^{\mathtt{l}}\mathsf{c}$, a "no-op" construct skip, an assignment construct $\mathsf{x} := \mathsf{a}$, a conditional construct $\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\,\mathsf{P}\,\}$ and an iteration construct $\mathsf{while}\ {}^{\mathtt{l}}\mathsf{b}\ \{\,\mathsf{P}\,\}$. Variables range over integer values, hence arithmetic expressions evaluate to values in $\mathbb{Z}$. Boolean expressions evaluate to boolean values in $\mathbb{B} \triangleq \{\mathbb{tt}, \mathbb{ff}\}$. The semantics of the language is given on top of *memories*, namely maps from variables to values, and *labels*, namely program control points. Let $\mathsf{Mem} \triangleq \mathsf{Var} \to \mathbb{Z}$ be the set of programs memories. Given $\mathsf{P} \in \mathsf{Imp}$, we denote with $\mathsf{Var}^{\mathsf{P}}$ and $\mathsf{Lab}^{\mathsf{P}}$ the set of variables and labels occurring in P, respectively. Similarly, we define the memories of P as those with domain equal to $\mathsf{Var}^{\mathsf{P}}$, namely the set $\mathsf{Mem}^{\mathsf{P}} \triangleq \{\mathsf{m} \in \mathsf{Mem} \mid \mathrm{dom}(\mathsf{m}) = \mathsf{Var}^{\mathsf{P}}\}$. A program *state* is a pair consisting in a label and a memory, hence the set of programs states is $\mathsf{Stat} \triangleq \mathsf{Lab} \times \mathsf{Mem}$. Given a program P, its set of states is $\mathsf{Stat}^{\mathsf{P}} \triangleq \mathsf{Lab}^{\mathsf{P}} \times \mathsf{Mem}^{\mathsf{P}}$.

We define the semantics of a program inductively from its syntax. In particular, it is built on top of the small-step operational semantics (SOS) of Imp, which is a sort of idealized interpreter of the language. This latter models the execution of programs step by step and it is specified by a set of inference rules modifying *configurations*. A *not-final* configuration $\langle \mathsf{s}, \mathsf{P} \rangle \in \mathsf{Stat} \times \mathsf{Imp}$ represents the current state s in which the program P has to be executed. A *final* configuration is just a state $\mathsf{s} \in \mathsf{Stat}$ and it represents the fact that the computation cannot go further. We denote with $\mathsf{Conf} \triangleq \mathsf{Stat} \times \mathsf{Imp} \cup \mathsf{Stat}$ the set of not-final and final configurations. The SOS inference rules in Fig. 4.3 describe, by means of the rewriting relation $\rightarrowtriangle \subseteq (\mathsf{Stat} \times \mathsf{Imp}) \times \mathsf{Conf}$, how not-final configurations evolve during time, until a final configuration is reached (if ever). We assume that programs are well-labeled. In a well-labeled program every concatenation of two programs is such that the exit-point of the first must be equal to the entry-point of the second, namely if $\mathsf{P} = {}^{\mathtt{i}}\mathsf{c}_1{}^{\mathtt{l}}\,.\,{}^{\mathtt{k}}\mathsf{c}_2{}^{\mathtt{f}}$ then $\mathtt{l} = \mathtt{k}$.

Remark. The SOS inference rules can rewrite a well-labeled program into a not well-labeled program. This is not a concern, because these not well-labeled programs are deliberately generated, and the rewriting relation $\rightarrowtriangle$ is able to deal with these particular cases. Indeed, the SOS needs to generate these not well-labeled programs in order to match the intended semantics of iteration commands.

The SOS rules rely on the operational semantics for arithmetic expressions $\Downarrow^{\mathbb{Z}}$ and for boolean expressions $\Downarrow^{\mathbb{B}}$. These latter are big-step semantics, since we are not interested in the intermediate steps of computation for expressions. In Fig. 4.2 we have the definition of

the (big-step) semantics for expressions. We denote with $\rightarrowtriangle^k$ the self-composition of the relation $\rightarrowtriangle$ a number $k > 0$ of times, hence $\langle \mathsf{s}, \mathsf{P}\rangle \rightarrowtriangle^k \mathsf{cnf} \in \mathsf{Conf}$ if there exists a sequence σ of $k+1$ configurations such that: $\sigma_0 = \langle \mathsf{s}, \mathsf{P}\rangle$, $\sigma_k = \mathsf{cnf}$ and $\forall i \in [0, k[\,.\, \sigma_i \rightarrowtriangle \sigma_{i+1}$. We denote with $\langle \mathsf{s}, \mathsf{P}\rangle \rightarrowtriangle^* \mathsf{cnf}$ the fact that there exists a $k > 0$ such that $\langle \mathsf{s}, \mathsf{P}\rangle \rightarrowtriangle^k \mathsf{cnf}$. A program $\mathsf{P} = {}^{\underline{\mathsf{i}}}\mathsf{c}^{\underline{\mathsf{f}}}$, starting in the state $\langle \underline{\mathsf{i}}, \mathsf{m}\rangle$, terminates, yielding the state $\langle \underline{\mathsf{f}}, \mathsf{n}\rangle$, if and only if $\langle\langle \underline{\mathsf{i}}, \mathsf{m}\rangle, \mathsf{P}\rangle \rightarrowtriangle^* \langle \underline{\mathsf{f}}, \mathsf{n}\rangle$. Conversely, it diverges (on $\langle \underline{\mathsf{i}}, \mathsf{m}\rangle$) if and only if it is possible to compose $\rightarrowtriangle$ from $\langle\langle \underline{\mathsf{i}}, \mathsf{m}\rangle, \mathsf{P}\rangle$ infinitely many times. We say that a program P' is a *derivative* of the program P, written $\mathsf{P} \dashrightarrow \mathsf{P}'$, when there exist $\mathsf{s}, \mathsf{s}' \in \mathsf{Stat}$ such that $\langle \mathsf{s}, \mathsf{P}\rangle \rightarrowtriangle^* \langle \mathsf{s}', \mathsf{P}'\rangle$.

4.1.3.1 Program Standard Semantics

In order to reason (semantically) about programs, we derive from the SOS of Imp the transition system associated to a given program. The (transition system) state space Σ^{P} of program $\mathsf{P} = {}^{\underline{\mathsf{i}}}\mathsf{c}^{\underline{\mathsf{f}}}$ is exactly $\mathsf{Stat}^{\mathsf{P}}$, its set of initial states Υ^{P} is $\{\langle \underline{\mathsf{i}}, \mathsf{m}\rangle \mid \mathsf{m} \in \mathsf{Mem}^{\mathsf{P}}\}$, its set of final states Ω^{P} is $\{\langle \underline{\mathsf{f}}, \mathsf{m}\rangle \mid \mathsf{m} \in \mathsf{Mem}^{\mathsf{P}}\}$ and the transition relation τ_{P} is the set $\{\langle \mathsf{s}, \mathsf{s}'\rangle \mid \exists \mathsf{P}' \in \mathsf{Imp}\,.\, \mathsf{P} \dashrightarrow \mathsf{P}' \wedge \langle \mathsf{s}, \mathsf{P}\rangle \rightarrowtriangle \langle \mathsf{s}', \mathsf{P}'\rangle\} \cup \{\langle \mathsf{s}', \mathsf{s}''\rangle \mid \exists \mathsf{P}', \mathsf{P}'' \in \mathsf{Imp}\,.\, \mathsf{P} \dashrightarrow \mathsf{P}' \wedge \mathsf{P}' \dashrightarrow \mathsf{P}'' \wedge \langle \mathsf{s}', \mathsf{P}'\rangle \rightarrowtriangle \langle \mathsf{s}'', \mathsf{P}''\rangle\}$. When P is clear from the context we write $\langle \Sigma, \Upsilon, \Omega, \tau\rangle$ instead of $\langle \Sigma^{\mathsf{P}}, \Upsilon^{\mathsf{P}}, \Omega^{\mathsf{P}}, \tau_{\mathsf{P}}\rangle$. As shown in Definition 21, we obtain the maximal trace semantics of P. By abstraction, we can retrieve all the semantics in the hierarchy of Figure 4.1.

Example 4. Consider the program ${}^{\underline{0}}\mathsf{x} := 0^{\underline{1}}\,.\,{}^{\underline{1}}\mathsf{while}\ {}^{\underline{2}}(\mathsf{y} < \mathsf{x})\ \{\ {}^{\underline{3}}\mathsf{y} := \mathsf{y} - 1^{\underline{4}}\ \}^{\underline{5}}$ in Imp. The corresponding transition system is $\langle \Sigma, \tau, \Upsilon, \Omega\rangle$, where $\Sigma \triangleq \{\underline{0}, \underline{1}, \ldots \underline{5}\} \times (\{\mathsf{x}, \mathsf{y}\} \to \mathbb{Z})$, the set of initial states is $\Upsilon \triangleq \{\underline{0}\} \times (\{\mathsf{x}, \mathsf{y}\} \to \mathbb{Z})$, the set of final states is $\Omega \triangleq \{\underline{5}\} \times (\{\mathsf{x}, \mathsf{y}\} \to \mathbb{Z})$ and the transition relation is (we denote with $[n\ m]$ the map $[\mathsf{x} \mapsto n\ \mathsf{y} \mapsto m]$):

$$\begin{aligned}\tau \triangleq\ & \{\langle\langle \underline{0}, [n\ m]\rangle, \langle \underline{1}, [0\ m]\rangle\rangle \mid n, m \in \mathbb{Z}\} \cup \{\langle\langle \underline{1}, [n\ m]\rangle, \langle \underline{2}, [n\ m]\rangle\rangle \mid n \in \mathbb{Z}\} \cup \\ & \{\langle\langle \underline{2}, [n\ m]\rangle, \langle \underline{3}, [n\ m]\rangle\rangle \mid m < n\} \cup \{\langle\langle \underline{2}, [n\ m]\rangle, \langle \underline{5}, [n\ m]\rangle\rangle \mid m \geq n\} \cup \\ & \{\langle\langle \underline{3}, [n\ m]\rangle, \langle \underline{4}, [n\ m-1]\rangle\rangle \mid n, m \in \mathbb{Z}\} \cup \{\langle\langle \underline{4}, [n\ m]\rangle, \langle \underline{2}, [n\ m]\rangle\rangle \mid n, m \in \mathbb{Z}\}\end{aligned}$$

Then the maximal trace semantics $\tau^{\vec{\infty}}$ of the transition system is:

$$\begin{aligned}& \{\langle \underline{5}, [n\ m]\rangle \in \Sigma^{\vec{+}} \mid n, m \in \mathbb{Z}\} \cup \{\langle \underline{2}, [n\ m]\backslash\langle \underline{5}, [n\ m]\rangle \in \Sigma^{\vec{+}} \mid m \geq n\} \cup \\ & \{\langle \underline{1}, [n\ m]\backslash\langle \underline{2}, [n\ m]\backslash\langle \underline{5}, [n\ m]\rangle \in \Sigma^{\vec{+}} \mid m \geq n\} \cup \\ & \{\langle \underline{0}, [n\ m]\backslash\langle \underline{1}, [0\ m']\backslash\langle \underline{2}, [0\ m']\backslash\langle \underline{5}, [0\ m']\rangle \in \Sigma^{\vec{+}} \mid m' \geq 0 \wedge n, m \in \mathbb{Z}\} \cup \\ & \{\langle \underline{4}, [n\ m]\backslash\langle \underline{2}, [n\ m]\backslash\langle \underline{5}, [n\ m]\rangle \in \Sigma^{\vec{+}} \mid m \geq n\} \cup \\ & \{\langle \underline{3}, [n\ m+1]\backslash\langle \underline{4}, [n\ m]\backslash\langle \underline{2}, [n\ m]\backslash\langle \underline{5}, [n\ m]\rangle \in \Sigma^{\vec{+}} \mid m \geq n\} \cup \\ & \left\{\langle \underline{0}, [n\ m]\backslash\langle \underline{1}, [0\ m]\backslash\langle \underline{2}, [0\ m]\backslash\langle \underline{3}, [0\ m]\backslash\langle \underline{4}, [0\ m-1]\backslash\langle \underline{2}, [0\ m-1]\rangle \ldots \in \Sigma^{\vec{\omega}} \;\middle|\; \begin{matrix} m < 0, \\ n \in \mathbb{Z} \end{matrix} \right\} \cup \\ & \{\langle \underline{1}, [0\ m]\backslash\langle \underline{2}, [0\ m]\backslash\langle \underline{3}, [0\ m]\backslash\langle \underline{4}, [0\ m-1]\backslash\langle \underline{2}, [0\ m-1]\rangle \ldots \in \Sigma^{\vec{\omega}} \mid m < 0\} \cup \\ & \{\langle \underline{2}, [0\ m]\backslash\langle \underline{3}, [0\ m]\backslash\langle \underline{4}, [0\ m-1]\backslash\langle \underline{2}, [0\ m-1]\rangle \ldots \in \Sigma^{\vec{\omega}} \mid m < 0\} \cup \\ & \{\langle \underline{3}, [0\ m]\backslash\langle \underline{4}, [0\ m-1]\backslash\langle \underline{2}, [0\ m-1]\rangle \ldots \in \Sigma^{\vec{\omega}} \mid m < 0\} \cup \ldots \\ & \{\langle \underline{2}, [n\ m]\backslash\langle \underline{3}, [n\ m]\backslash\langle \underline{4}, [n\ m-1]\backslash\langle \underline{2}, [n\ m-1]\rangle \ldots \in \Sigma^{\vec{\omega}} \mid n, m \in \mathbb{Z}\} \cup \\ & \{\langle \underline{3}, [n\ m]\backslash\langle \underline{4}, [n\ m-1]\backslash\langle \underline{2}, [n\ m-1]\rangle \ldots \in \Sigma^{\vec{\omega}} \mid n, m \in \mathbb{Z}\} \cup \ldots\end{aligned}$$

As we can note in Example 4, the maximal trace semantics is suffix-closed. Usually, such suffixes are not useful and, sometimes, they could also insert misleading information. Consider the very simple program $\overset{0}{}\mathsf{x} := 3\overset{1}{}\,.\,\overset{1}{}\mathsf{skip}\overset{2}{}$. Its transition system is composed by $\Sigma \triangleq \{\overset{0}{}, \overset{1}{}, \overset{2}{}\} \times (\{\mathsf{x}\} \to \mathbb{Z})$, $\Upsilon \triangleq \{\langle \overset{0}{}, [n]\rangle \mid n \in \mathbb{Z}\}$, $\Omega \triangleq \{\langle \overset{2}{}, [n]\rangle \mid n \in \mathbb{Z}\}$ and $\tau \triangleq \{\langle\langle \overset{0}{}, [n]\rangle, \langle \overset{1}{}, [3]\rangle\rangle \mid n \in \mathbb{Z}\} \cup \{\langle\langle \overset{1}{}, [n]\rangle, \langle \overset{2}{}, [n]\rangle\rangle \mid n \in \mathbb{Z}\}$[4]. The the *maximal trace semantics of the transition system* is:

$$\tau^{\vec{\infty}} \triangleq \{\langle \overset{2}{}, [n]\rangle \mid n \in \mathbb{Z}\} \cup \{\langle \overset{1}{}, [n]\rangle\langle \overset{2}{}, [n]\rangle \mid n \in \mathbb{Z}\} \cup \{\langle \overset{0}{}, [n]\rangle\langle \overset{1}{}, [3]\rangle\langle \overset{2}{}, [3]\rangle \mid n \in \mathbb{Z}\}$$

It is clear that the only useful complete traces are those in $\{\langle \overset{0}{}, [n]\rangle\langle \overset{1}{}, [3]\rangle\langle \overset{2}{}, [3]\rangle \mid n \in \mathbb{Z}\}$. Indeed, the traces in $\{\langle \overset{2}{}, [n]\rangle \mid n \in \mathbb{Z} \setminus \{3\}\}$ are not even reachable, meaning that it is not possible to generate them with a real execution of the program. For this reasons, we consider as *maximal traces of a program* only those traces starting in an initial state. Hence, from now on, we consider as maximal trace semantics of a program the set $\{\bar{\sigma} \in \tau^{\vec{\infty}} \mid \bar{\sigma}_0 \in \Upsilon\}$.

Transition systems allow us to relate and compare different semantics of programs in an unique setting, but they are not so useful for verification methods. Indeed, when we want to prove properties of programs, we need to compute these semantics with a computer. Hence, the semantics of interest should be defined by induction on programs syntax, without passing from a transition system representation. In the next chapters we will see how to compute semantics, directly from programs code.

Arithmetic expressions: $\Downarrow^{\mathbb{Z}} \subseteq (\mathsf{Aexp} \times \mathsf{Mem}) \times \mathbb{Z}$

$$\langle \mathsf{n}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n \qquad \langle \mathsf{x}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} \mathsf{m}(\mathsf{x}) \qquad \langle (\mathsf{a}), \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n \quad \text{if } \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n$$
$$\langle \mathsf{a}_1 + \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n \quad \text{if } \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \text{ and } \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \text{ and } n = n_1 + n_2$$
$$\langle \mathsf{a}_1 - \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n \quad \text{if } \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \text{ and } \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \text{ and } n = n_1 - n_2$$
$$\langle \mathsf{a}_1 * \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n \quad \text{if } \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \text{ and } \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \text{ and } n = n_1 * n_2$$

Boolean expressions: $\Downarrow^{\mathbb{B}} \subseteq (\mathsf{Bexp} \times \mathsf{Mem}) \times \mathbb{B}$

$$\langle \mathsf{tt}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt} \qquad \langle \mathsf{ff}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{ff} \qquad \langle (\mathsf{b}), \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b \quad \text{if } \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b$$
$$\langle \mathsf{a}_1 = \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b \quad \text{if } \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \text{ and } \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \text{ and } b = (\!| n_1 = n_2 \mathbin{?} \mathbb{tt} : \mathbb{ff} |\!)$$
$$\langle \mathsf{a}_1 \neq \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b \quad \text{if } \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \text{ and } \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \text{ and } b = (\!| n_1 \neq n_2 \mathbin{?} \mathbb{tt} : \mathbb{ff} |\!)$$
$$\langle \mathsf{a}_1 < \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b \quad \text{if } \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \text{ and } \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \text{ and } b = (\!| n_1 < n_2 \mathbin{?} \mathbb{tt} : \mathbb{ff} |\!)$$
$$\langle \mathsf{a}_1 \leq \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b \quad \text{if } \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \text{ and } \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \text{ and } b = (\!| n_1 \leq n_2 \mathbin{?} \mathbb{tt} : \mathbb{ff} |\!)$$
$$\langle \mathsf{b}_1 \wedge \mathsf{b}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b \quad \text{if } \langle \mathsf{b}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b_1 \text{ and } \langle \mathsf{b}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b_2 \text{ and } b = b_1 \wedge b_2$$
$$\langle \mathsf{b}_1 \vee \mathsf{b}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b \quad \text{if } \langle \mathsf{b}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b_1 \text{ and } \langle \mathsf{b}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b_2 \text{ and } b = b_1 \vee b_2$$
$$\langle \neg \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b \quad \text{if } \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} b' \text{ and } b = \neg b'$$

Figure 4.2: Big-step operational semantics for expressions.

[4] We denote with $[n]$ the memory $[\mathsf{x} \mapsto n]$

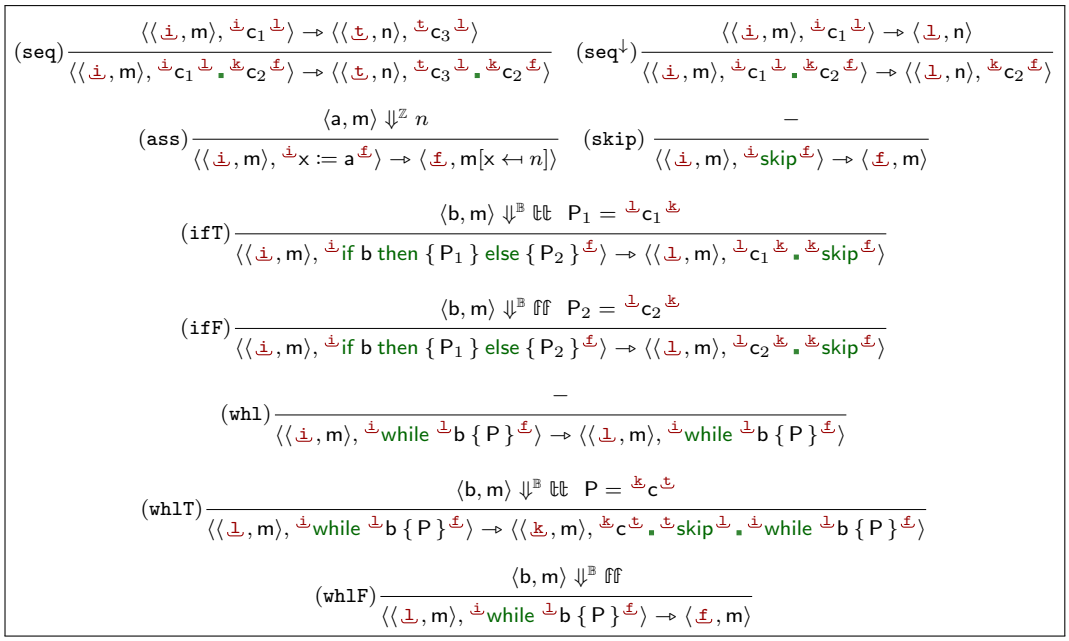


Figure 4.3: Small-step operational semantics of Imp.

4.2 Formalizing Specifications

In Section 4.1 we have fixed systems model and systems behavior. Similarly, in this section we define which are the specifications we are interested in. We have chosen *hyperproperties* [Clarkson and Schneider, 2010], which are the most flexible way for representing systems specifications, when systems are described by means of sets of executions.

4.2.1 Hyperproperties: Introduction

As introduced in the previous section, we model systems executions as traces of states, where a state is the most precise information we have about the system in a particular interval of time. We maintain the notation of the previous section, denoting with $\Sigma^{\mathbb{s}} \subseteq \mathfrak{D}$ the state space of a particular system $\mathbb{s}$, where $\mathfrak{D}$ is the set of all possible state denotations of all possible systems. Then, we have a lot of choices for execution denotations. For instance, maximal traces, input/output relations, states, etc. In this section we reason about specifications, in a way independent of the choice of executions denotations. Hence, in the following, we denote with $\mathbb{Den}$ a generic set of executions denotations. For instance, we have that $\mathbb{Den} = \Sigma^{\vec{\infty}}$ if we chose maximal traces. We model systems semantics as set of executions, namely the semantics domain is $\wp(\mathbb{Den})$. It is clear that, in this settings, a specification is modeled as a *set of sets of executions*, namely an element of $\wp(\wp(\mathbb{Den}))$. Specifications modeled in this way are called *hyperproperties* [Clarkson and Schneider, 2010] in the literature. The authors of [Clarkson and Schneider, 2010] had to introduce the prefix "hyper", in order to distinguish specifications described as set of sets of executions from specifications described as

sets of executions. These latter were called improperly systems properties and were used for decades as the general formalization for systems specifications. The terminology is improper since a set of executions cannot be termed as a property, in the mathematical sense, of systems. In fact they are properties of executions, since they are defined by means of a characteristic function with domain $\mathbb{Den}$, instead of $\wp(\mathbb{Den})$. In order to disambiguate, from now on we call hyperproperties the properties of systems, defined on $\wp(\wp(\mathbb{Den}))$, and we call *trace properties* the properties of executions, defined on $\wp(\mathbb{Den})$. Finally we will use "property" in order to denote the general concept of mathematical property. We synthesize these concepts in the following definitions.

- Given a set X, a (mathematical) *property* over X is just a subset of X.
- Given a set of systems executions denotations $\mathbb{Den}$, a *hyperproperty* is just a property over $\wp(\mathbb{Den})$, namely it is a subset of $\wp(\mathbb{Den})$.
- Given a set of systems executions denotations $\mathbb{Den}$, an *trace property* is just a property over $\mathbb{Den}$, namely it is a subset of $\mathbb{Den}$.

Note that hyperproperties generalize trace properties since, as we will see formally later, these latter are isomorphic to a subset of hyperproperties. From now on, we will denote specifications as hyperproperties. Recalling Chapter 3, we say that $\mathbb{s}$ satisfies an hyperproperty $\mathfrak{Hp} \in \wp(\wp(\mathbb{Den}))$, in symbols $\mathbb{s} \models \mathfrak{Hp}$, when $\mathcal{I}(\mathbb{s}) \in \mathfrak{Hp}$. Here $\mathcal{I}(\mathbb{s}) \in \mathbb{REP}^{\mathsf{Sys}} \triangleq \mathbb{Den}$ is the semantics of $\mathbb{s}$, namely an element of the hierarchy presented the previous section. Analogously, we can state when a system satisfies a trace property. We use again the same symbol $\models$ in order to denote the satisfiability relation for trace properties: a system $\mathbb{s}$ satisfies a trace property $\mathfrak{P} \in \wp(\mathbb{Den})$, written $\mathbb{s} \models \mathfrak{P}$, when $\mathcal{I}(\mathbb{s}) \subseteq \mathfrak{P}$.

It is clear that, being predicates on single executions, trace properties could express safety requirements like "the system does not go in an error state" but they cannot express, for instance, confidentiality requirements. These latter need to compare *different* executions of the system, hence a predicate on single executions is not sufficient. Hyperproperties, instead, could express *relations* between executions, since the level of set of sets allows to use predicates over multiple executions. This difference is depicted in Figure 4.4. More expressiveness comes at a cost: the verification of hyperproperties is significantly more complex than the verification of trace properties, as we will see in the next chapters.

We continuously maintain this parallel between hyperproperties and trace properties since the majority of works about specifications analysis/verification deal with these latter only. In the following sections we investigate hyperproperties from a theoretical point of view. First, by a topological point of view (Subsection 4.2.2) and then from an algebraic point of view (Subsection 4.2.3). We will exploit these theoretical findings in order to build verification methods for hyperproperties in Chapters 6, 7 and 8.

It is worth noting that in order to disprove that a system fulfills a trace property it is necessary to show one trace, which is the counterexample. Analogously, in order to disprove that a system fulfills a hyperproperty is necessary to show a set of traces (potentially all system traces). Finally, the trace property $\varnothing$, written $\mathfrak{P}^{\mathrm{false}}$, is the one which cannot be satisfied, by any system, i.e. $\nexists \mathbb{s} \,.\, \mathcal{I}(\mathbb{s}) \subseteq \mathfrak{P}^{\mathrm{false}}$ ($\varnothing$ is not a representation of any system). Dually, the trace property $\mathbb{Den}$, written $\mathfrak{P}^{\mathrm{true}}$, is the one which is satisfied by every system, i.e. $\forall \mathbb{s} \,.\, \mathcal{I}(\mathbb{s}) \subseteq \mathfrak{P}^{\mathrm{true}}$. Analogously, we can define $\mathfrak{Hp}^{\mathrm{false}}$ and $\mathfrak{Hp}^{\mathrm{true}}$ for hyperproperties as $\varnothing$ and $\wp(\mathbb{Den})$, respectively.

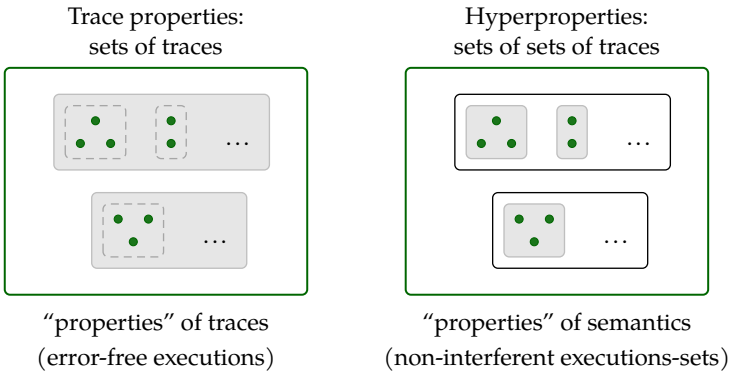


Figure 4.4: Trace properties and hyperproperties.

Remark. In [Clarkson and Schneider, 2010] the authors define $\mathfrak{Hp}^{\text{false}}$ as $\{\varnothing\}$ instead of $\varnothing$. From the point of view of verification, the two definitions are equivalent. In fact, both $\varnothing$ and $\{\varnothing\}$ are unsatisfiable: there is not a system $\mathbb{s}$ such that $\mathcal{I}(\mathbb{s}) \in \varnothing$ or $\mathcal{I}(\mathbb{s}) \in \{\varnothing\}$. We prefer to use $\varnothing$ since, by an algebraic point of view, it is more general (see Section 4.2.3).

Examples of Hyperproperties. As we have already said, information flows are expressible only with hyperproperties. Now we give other examples of useful specifications, which are not expressible with trace properties. *Processes synchronization requirements* are hyperproperties. In a symmetric mutual exclusion protocol, for every pair of mutual exclusive requests to a critical section the respective grant to enter the section should be answered eventually, but not at the same time. *Service Level Agreements* are hyperproperties. For instance, the average time, over all executions, to respond to requests must be lower than a given threshold. *Error resistant codes* requirements are hyperproperties. Error resistant codes enable the transmission of data over noisy channels. A typical model of errors bounds the number of flipped bits that may happen for a given code word length. Then, error correction coding schemes must guarantee that all code words have a minimal Hamming distance. This is an hyperproperty, since for every pair of executions with different input data, the encoder must produce code words with, at least, a given Hamming distance. Finally, also *cryptographic protocols* requirements are hyperproperties. For instance, in a secret-sharing scheme a (secret) sensitive data is divided in k parts and each part is distributed to a distinct agent. All k of these shares are required to reconstruct the original secret data. In this setting, a system should not, across all of its executions, output all k parts.

4.2.2 Hyperproperties: Topological Characterization

The aim of this subsection is to give a topological characterization of trace properties and hyperproperties, parametric on the execution denotations domain. First we deal with trace properties and then we pass to hyperproperties.

Among all the trace properties, there are some with particular characteristics [Alpern and Schneider, 1985]. The *safety* (*trace*) *properties* express the fact that "nothing bad happens", namely the systems satisfying a safety trace property do not exhibit bad behaviors.

This means that for every trace that is not in a safety trace property, there exists a finite prefix (the "bad thing") which cannot be extended to a trace satisfying the property. The most useful feature of safety trace properties is that to check if the specification does not hold for an execution, and so for the system, it is sufficient to observe only a finite prefix of the trace. The *liveness* (*trace*) *properties* express the fact that "something good eventually happens", namely the systems satisfying a liveness trace property are those that, eventually, exhibit a good behavior. This means that a finite trace is part of a liveness trace (a good behavior) if it can be extended to an infinite trace satisfying the property. It is well known that every trace property can be defined as the intersection of a safety and a liveness trace property [Alpern and Schneider, 1985], using topological arguments.

A Primer on Topologies. A *topology* over a set X consists in a family of subsets of X which defines its *open sets*. $\mathfrak{O}_X \subseteq \wp(X)$ is a family of open sets if and only if it is closed under union (i.e. $\forall \mathcal{Y} \subseteq \mathfrak{O}_X \,.\, \bigcup \mathcal{Y} \in \mathfrak{O}_X$), it is closed under binary intersection (i.e. $Y_1, Y_2 \in \mathfrak{O}_X \Rightarrow Y_1 \cap Y_2 \in \mathfrak{O}_X$) and it covers X (i.e. $\bigcup \mathfrak{O}_X = X$). The dual of an open set is a closed set, so a family of open sets defines automatically a family of *closed sets*, namely $\mathfrak{C}_X = \{X \setminus O \mid O \in \mathfrak{O}_X\}$. Given $Y \subseteq X$, the *interior* of Y, written $\iota(Y)$, is the largest open set contained in Y and the *closure* of Y, written $\rho(Y)$, is the smallest closed set containing Y, i.e.

$$\rho(Y) = \bigcap \{C \in \mathfrak{C}_X \mid Y \subseteq C\} \text{ and } \iota(Y) = \bigcup \{O \in \mathfrak{O}_X \mid O \subseteq Y\}$$

A set $D \subseteq X$ is said *dense* if and only if $\rho(D) = X$, so in a topology there is also a family of dense sets, i.e. $\mathfrak{D}_X = \{D \subseteq X \mid \rho(D) = X\}$. Every element of $\wp(X)$ can be specified as the intersection of a closed set and a dense set. We have not found any explicit proof in the literature, so we give a simple one here. Note that the proof also provides a way for retrieving the closed and the dense sets whose intersection is the given element of $\wp(X)$.

Theorem 8 (Decomposition)**.** $\forall Y \in \wp(X)\, \exists C \in \mathfrak{C}_X, D \in \mathfrak{D}_X \,.\, Y = C \cap D$.

Proof. First, consider the following equalities:

$$\begin{array}{ll}
Y = & \| \text{ extensivity of } \rho \\
\quad \rho(Y) \cap Y & \\
\quad = & \| \text{ set-theory} \\
\quad (\rho(Y) \cap Y) \cup \varnothing & \\
\quad = & \| \text{ set-theory} \\
\quad (\rho(Y) \cap Y) \cup (\rho(Y) \cap (X \setminus \rho(Y))) & \\
\quad = & \| \text{ distributivity of } \cap \\
\rho(Y) \cap (Y \cup (X \setminus \rho(Y))) &
\end{array}$$

Then we can note that $\rho(Y) \in \mathfrak{C}_X$, i.e. it is a closed set, and $Y \cup (X \setminus \rho(Y)) \in \mathfrak{D}_X$, i.e. it is a

dense set. The first is closed by definition, while for the second we have:

$$
\begin{array}{ll}
\rho(Y \cup (X \setminus \rho(Y))) & \\
\supseteq & \| \text{ property of closures} \\
\rho(Y) \cup \rho(X \setminus \rho(Y)) & \\
\supseteq & \| \text{ extensivity of } \rho \\
\rho(Y) \cup (X \setminus \rho(Y)) & \\
= X & \| \text{ set-theory}
\end{array}
$$

This implies $\rho(Y \cup (X \setminus \rho(Y))) = X$, since $\rho(Z) \subseteq X$ for every $Z \in \wp(X)$. □

Definition 22. A function $\kappa \in \wp(X) \to \wp(X)$ is a *Kuratowski Closure Operator*, KCO in short, if and only if all the following hold:

1. $\kappa(\varnothing) = \varnothing$
2. $\forall Y \subseteq X \,.\, Y \subseteq \kappa(Y)$
3. $\forall Y_1, Y_2 \subseteq X \,.\, \kappa(Y_1 \cup Y_2) = \kappa(Y_1) \cup \kappa(Y_2)$ [this implies $Y_1 \subseteq Y_2 \Rightarrow \kappa(Y_1) \subseteq \kappa(Y_2)$]
4. $\forall Y \subseteq X \,.\, \kappa(\kappa(Y)) = \kappa(Y)$

A KCO over $\wp(X)$ induces a topology on X where the KCO is the closure of X [Kuratowski, 1967]. So, in order to define a topology on X, it is sufficient to specify its closed sets, namely a closure (or a KCO) over $\wp(X)$.

Extra Notation. Limited to this subsection, we define some concepts and notations, in order to make the presentation more readable. We denote the set $\Sigma_\epsilon^{\dagger}$ of non-empty finite sequences over Σ just as Σ^+. Similarly, we denote the set $\Sigma_\epsilon^{\vec{\omega}}$ of non-empty infinite sequences as Σ^ω and the set $\Sigma_\epsilon^{\vec{\infty}}$ of non-empty sequences (finite or infinite) as Σ^∞.

The sequence $\bar{\sigma} \in \Sigma^+$ is a *prefix* of $\bar{\sigma}'' \in \Sigma^\infty$, in symbols $\bar{\sigma} \leq^{\text{pf}} \bar{\sigma}''$, if there exists $\bar{\sigma}' \in \Sigma^\infty$ such that $\bar{\sigma}\bar{\sigma}' = \sigma''$. When we deal with sets of sequences, we can extend the definition of prefix to sets as follows. A set of sequences $X \subseteq \Sigma^+$ is a *prefixset* of $Y \subseteq \Sigma^\infty$, in symbols $X \trianglelefteq^{\text{pf}} Y$, if for all $\bar{\sigma} \in X$ there exists $\bar{\sigma}' \in Y$ such that $\bar{\sigma} \leq^{\text{pf}} \bar{\sigma}'$ [Clarkson and Schneider, 2010]. A set of infinite sequences $\{\bar{\sigma}^0, \bar{\sigma}^1, \dots\}$ converges to the *limit sequence* $\bar{\sigma}$ if the length of the maximal prefix common to each $\bar{\sigma}^k$ and to $\bar{\sigma}$ goes to infinity as k goes to infinity [Chang, Manna, and Pnueli, 1992].

Example 5. Consider the set of sequences $\{a^n b^\omega \mid n \geq 0\}$. It converges to a^ω, since the sequence of longest prefixes common to a^ω and to $\sigma_k = a^k b^\omega$ (i.e. a^k) gets increasingly longer with k.

Given two sets X and Y and a function $f \in X \to \wp(Y)$, we denote with $f^\uparrow \in \wp(X) \to \wp(Y)$ the function $f^\uparrow \triangleq \lambda Z \,.\, \bigcup\{f(x) \mid x \in Z\}$, called the set image-lift of f.

4.2.2.1 Safety and Liveness Dichotomy

As already mentioned, among all, there are two particular kinds of trace properties: the safety and the liveness. Informally, the first model the fact that "nothing bad will happen" and the second model the fact that "something good will eventually happen". In other words, a system violates a safety trace property if it eventually performs the "bad thing" and a system violates a liveness trace property if it never performs the "good thing". It is clear that, if a system does not satisfy a safety trace property, the violation must occur during its execution and hence the violation must arise in a *finite* amount of time. Due to this fact, safety trace properties are identified as the ones which can be *monitored*, i.e. checked at runtime. For liveness trace properties things are more complicated because the checker must observe the system execution entirely, hence it needs a, *possibly infinite*, amount of time.

Finite executions can be seen as particular cases of infinite ones: we can repeat infinitely many times the final state of a finite execution in order to obtain an infinite execution equivalent to the finite one. This has led researchers to model system executions and trace properties as set of infinite sequences of systems states. This choice has also two other important motivations: reasoning about trace properties can be done with well studied formalisms modeling semantics by considering infinite sequences (like linear temporal logics and Büchi automata), and it allows us to give a topological characterization of trace properties. It turns out that safety trace properties correspond to the closed sets in the Cantor topology over infinite sequences and liveness trace properties correspond to the dense sets. Hence, by using the decomposition theorem (Theorem 8), we can specify an arbitrary trace property as the intersection of a safety and a liveness one. This means that we can reduce the check of a generic trace property to the check of its safety and liveness parts.

While trace properties over finite sequences can be easily expressed as infinite sequences, in practice we deal with systems which exhibit finite behaviors. So it is natural to wonder what happens if we allow finite sequences in trace properties definition. Something in this direction was already done [Roşu, 2012], but only for safety trace properties. In this section, we give a trace properties characterization on the following execution domains: only finite Σ^+, only infinite Σ^ω and mixed Σ^∞.

Remark. Clearly the concepts of safety and liveness are useful only for execution denotations strictly more precise than I/O traces. Indeed, these properties embody the notion of computation history, i.e. executions keep track of past states. In a trace composed by just two elements, or by a single state, the concept of history is not definable.

Safety. Let us start with safety first and denote by Safety^+, Safety^ω and Safety^∞ the safety trace properties over finite, infinite and mixed executions, respectively. In [Alpern and Schneider, 1985], the authors define safety trace properties S on Σ^ω in a refutational way: if $\bar{\sigma} \notin S$ then there exists a finite prefix $\bar{\sigma}'$ of $\bar{\sigma}$, written $\bar{\sigma}' \leq^{\text{pf}} \bar{\sigma}$, such that $\bar{\sigma}'\bar{\sigma}'' \notin S$ for every $\bar{\sigma}'' \in \Sigma^\omega$. This means that, if an infinite execution violates the property, then the bad thing must have occurred in one of its finite prefixes and the violation cannot ever be recovered in the future. An alternative, and equivalent, definition due to Roşu [Roşu, 2012] is the following: for a safety trace property $S \in \wp(\Sigma^\omega)$, $\bar{\sigma} \in S$ if and only if $\mathtt{prf}(\bar{\sigma}) \subseteq \mathtt{prf}^\uparrow(S)$, where $\mathtt{prf} \in \Sigma^\infty \to \wp(\Sigma^+)$ is the function $\lambda\bar{\sigma} \,.\, \{\bar{\sigma}' \mid \exists\bar{\sigma}'' \in \Sigma^\infty \,.\, \bar{\sigma}'\bar{\sigma}'' = \bar{\sigma}\}$ returning the set of prefixes of a given sequence. Furthermore, Roşu in [Roşu, 2012] discusses the known definitions of safety trace properties over finite, infinite and mixed executions, and their

equivalence with the following:

- $\mathsf{Safety}^{+} \triangleq \{S \in \wp(\Sigma^{+}) \mid S = \mathtt{prf}^{\uparrow}(S)\}$
- $\mathsf{Safety}^{\omega} \triangleq \{S \in \wp(\Sigma^{\omega}) \mid \bar{\sigma} \in S \Leftrightarrow \mathtt{prf}(\bar{\sigma}) \subseteq \mathtt{prf}^{\uparrow}(S)\}$
- $\mathsf{Safety}^{\infty} \triangleq \{S \in \wp(\Sigma^{\infty}) \mid \bar{\sigma} \in S \Leftrightarrow \mathtt{prf}(\bar{\sigma}) \subseteq S\}$

Although safety trace properties essentially capture the fact that, in order to disprove the property, it is sufficient to show a finite counterexample, the definitions of safety on finite, infinite and mixed sequences are different. For finite sequences it is sufficient the prefix-closure, namely a safety on Σ^{+} must contain all the prefixes of its sequences. The same definition cannot be applied to Σ^{ω}, indeed none of its prefixes are in the property. In this case we have to reason "at the limit" and say that a safety on Σ^{ω} contains all its limit sequences, namely the infinite sequences which approximate the prefixes. Finally, as expected, the safety on Σ^{∞} combines both aspects of finite and infinite sequences.

Liveness. To the best of our knowledge, there are no works reasoning about liveness trace properties over finite and mixed executions. One can think that it is not meaningful to define liveness on finite executions, but we believe it is not the case. Take as example termination, i.e. the set of systems executions which do not run forever. Clearly this trace property is liveness, where the good thing is exactly termination. We can model this trace property on finite sequences only, as the set Σ^{+}. So, let us denote with $\mathsf{Liveness}^{+}$, $\mathsf{Liveness}^{\omega}$ and $\mathsf{Liveness}^{\infty}$ the liveness trace properties over finite, infinite and mixed executions, respectively. The original definition of Alpern and Schneider [Alpern and Schneider, 1985] involves infinite sequences only, and it states that a trace property $L \in \wp(\Sigma^{\omega})$ is a liveness trace property if and only if for every finite sequence $\bar{\sigma} \in \Sigma^{+}$ there exists an infinite sequence $\bar{\sigma}' \in \Sigma^{\omega}$ such that $\bar{\sigma}\bar{\sigma}'$ is in L. This means that every finite execution can be extended to an infinite one satisfying the property. We believe that this intuition can be easily adapted to finite and mixed sequences as well.

Definition 23. Given $\mathfrak{y} \in \{+, \omega, \infty\}$:

$$\mathsf{Liveness}^{\mathfrak{y}} \triangleq \{L \in \wp(\Sigma^{\mathfrak{y}}) \mid \forall \bar{\sigma} \in \Sigma^{+}\, \exists \bar{\sigma}' \in L \,.\, \bar{\sigma} \leq^{\mathrm{pf}} \bar{\sigma}'\}$$

Liveness trace properties capture the fact that in order to disprove the property it is necessary to show an infinite counterexample. It is worth noting that our definition of liveness on infinite executions is indeed equivalent to the one of Alpern and Schneider. Moreover, $\forall \bar{\sigma} \in \Sigma^{+}\, \exists \bar{\sigma}' \in L \,.\, \bar{\sigma} \leq^{\mathrm{pf}} \bar{\sigma}'$ is equivalent to $\Sigma^{+} \subseteq \mathtt{prf}^{\uparrow}(L)$.

Finally, we can note that, as usual, the trace property $\mathfrak{P}^{\mathrm{false}}$ is a safety property for Σ^{+}, Σ^{ω} and Σ^{∞} but it is a liveness trace property for none of them. Analogously, $\Sigma^{+} \in \mathsf{Safety}^{+} \cap \mathsf{Liveness}^{+}$, $\Sigma^{\omega} \in \mathsf{Safety}^{\omega} \cap \mathsf{Liveness}^{\omega}$ and $\Sigma^{\infty} \in \mathsf{Safety}^{\infty} \cap \mathsf{Liveness}^{\infty}$ (here Σ^{+}, Σ^{ω} and Σ^{∞} are the $\mathfrak{P}^{\mathrm{true}}$ trace properties).

4.2.2.2 Hypersafety and Hyperliveness Dichotomy

Similarly, among all hyperproperties, there are some with particular characteristics [Clarkson and Schneider, 2010]. The *safety hyperproperties* (or hypersafety) are the lift to sets of safety trace properties. This means that for every set of traces which is not a member of a

safety hyperproperty, there exists a finite prefixset of finite traces (the "bad thing") which cannot be extended to a set of traces satisfying the hyperproperty. The *liveness hyperproperties* (or hyperliveness) are the lift to sets of liveness trace properties. This means that a set of finite traces is a member of a hyperliveness traces-set (the "good thing") if it can be extended to a set of infinite traces satisfying the hyperproperty. In [Clarkson and Schneider, 2010] the authors proved that every hyperproperty can be defined as the intersection of a safety and a liveness hyperproperty (on infinite traces only).

Hypersafety. In their seminal paper [Clarkson and Schneider, 2010], Clarkson and Schneider introduce hyperproperties and they extend the safety/liveness dichotomy on sets of sets of sequences. Their work, as well as all other works about hyperproperties, deals with infinite executions only. In this section, we mimic what we have done for trace properties in the hyper case.

Let us denote by $\mathsf{HyperSafety}^{+}$, $\mathsf{HyperSafety}^{\omega}$ and $\mathsf{HyperSafety}^{\infty}$ the safety hyperproperties over finite, infinite and mixed executions, respectively. In [Clarkson and Schneider, 2010] the authors define hypersafety in a refutational way: if $X \notin \mathcal{S}$ then there exists a *finite* set of finite sequences $O \trianglelefteq^{\text{pf}} X$ such that every possible $X' \in \wp(\Sigma^{\omega})$ which extends O (i.e. $O \trianglelefteq^{\text{pf}} X'$) is not in $\mathcal{S}$. This is basically the concept of safety lifted to sets, where the "bad thing" is exactly the set O. Here we define hypersafety for finite, infinite and mixed executions, lifting to sets the definitions of safety. The function $\texttt{sprf} \in \wp(\Sigma^{\infty}) \to \wp(\wp(\Sigma^{+}))$ is $\lambda X\,.\,\{Y \mid Y \trianglelefteq^{\text{pf}} X\} = \lambda X\,.\,\{Y \mid \forall \bar{\sigma} \in Y\, \exists \bar{\sigma}' \in X\,.\,\bar{\sigma} \leq^{\text{pf}} \bar{\sigma}'\}$ and it returns the set of prefixsets of X. Note that `sprf` does not constrain prefixsets to have finite size, indeed in our definitions we allow the "bad thing" set to be infinite.

- $\mathsf{HyperSafety}^{+} \triangleq \{\mathcal{S} \in \wp(\wp(\Sigma^{+})) \mid \mathcal{S} = \texttt{sprf}^{\uparrow}(\mathcal{S})\}$
- $\mathsf{HyperSafety}^{\omega} \triangleq \{\mathcal{S} \in \wp(\wp(\Sigma^{\omega})) \mid X \in \mathcal{S} \Leftrightarrow \texttt{sprf}(X) \subseteq \texttt{sprf}^{\uparrow}(\mathcal{S})\}$
- $\mathsf{HyperSafety}^{\infty} \triangleq \{\mathcal{S} \in \wp(\wp(\Sigma^{\infty})) \mid X \in \mathcal{S} \Leftrightarrow \texttt{sprf}(X) \subseteq \mathcal{S}\}$

Hypersafety essentially captures the fact that, in order to disprove the hyperproperty, it is sufficient to show a counterexample-set of finite traces. Hence, also in the hyper case, we have the link between safety and the concept of monitorability. In fact, as a safety trace property can be disproved at runtime observing *one execution* until the "bad thing" happens, an hypersafety can be disproved at runtime observing a *set of executions* until the "bad thing" happens. Note that, this set can have an unbounded (or infinite) number of elements hence, in general, the monitorability of an hypersafety is unfeasible. But there are some exceptions. For k-hypersafety, i.e. safety hyperproperties for which the bad thing never involves more than k traces (see [Clarkson and Schneider, 2010] for details), the set of traces we need to monitor can be restricted to $k \in \mathbb{N}$ (i.e. a finite number of) elements.

In order to characterize hypersafety for finite sequences, it is sufficient the prefixset-closure, namely an hypersafety on Σ^{+} must contain all the prefixsets of its sequences. The same definition cannot be applied to Σ^{ω}, indeed none of its prefixsets are in the property. In this case, again, we have to reason "at the limit" and say that an hypersafety on Σ^{ω} contains all its sets of limit sequences, namely the sets of infinite sequences which approximate the prefixsets. Finally, as expected, the hypersafety on Σ^{∞} combine both aspects of finite and infinite sequences. Furthermore, it is worth noting that our definition of hypersafety on

infinite executions is indeed equivalent to the one of Clarkson and Schneider, if we constrain `sprf` to collect only finite prefixsets.

Hyperliveness. Let us now denote by $\mathsf{HyperLiveness}^{+}$, $\mathsf{HyperLiveness}^{\omega}$ and $\mathsf{HyperLiveness}^{\infty}$ the liveness hyperproperties over finite, infinite and mixed executions, respectively. In [Clarkson and Schneider, 2010], the original definition states that an hyperproperty $\mathcal{L} \in \wp(\wp(\Sigma^{\omega}))$ is hyperliveness if and only if for every *finite* set of finite sequences $O \in \wp(\Sigma^{+})$ there exists a set of infinite sequences $X \in \wp(\Sigma^{\omega})$, which extends O (i.e. $O \preceq^{\mathrm{pf}} X$), such that X is in $\mathcal{L}$. Also in this case, the definition is basically the concept of liveness lifted to sets. Here we give an alternative definition, which turns out to be parameterizable on finite, infinite and mixed executions, as it happens for trace properties case. As we have done for hypersafety, in our definitions we relax the constraint that the observable O must to be a finite set.

Definition 24. Given $\mathfrak{y} \in \{+, \omega, \infty\}$:

$$\mathsf{HyperLiveness}^{\mathfrak{y}} = \{\mathcal{L} \in \wp(\wp(\Sigma^{\mathfrak{y}})) \mid \forall O \in \wp(\Sigma^{+})\, \exists X \in \mathcal{L}\,.\, O \preceq^{\mathrm{pf}} X\}$$

Hyperliveness captures the fact that, in order to disprove the hyperproperty, it is necessary to show a set of infinite counterexamples. It is worth noting that our definition of hyperliveness on infinite executions is indeed equivalent to the one of Clarkson and Schneider if we constrain every O to be finite. Furthermore, the condition $\forall O \in \wp(\Sigma^{+})\, \exists X \in \mathcal{L}\,.\, O \preceq^{\mathrm{pf}} X$ is equivalent to: $\wp(\Sigma^{+}) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{L})$.

Finally, we can note that, as expected, the hyperproperty $\mathfrak{Hp}^{\mathrm{false}}$ is hypersafety for Σ^{+}, Σ^{ω} and Σ^{∞} but it is hyperliveness for none of them. Analogously, $\wp(\Sigma^{+}) \in \mathsf{HyperSafety}^{+} \cap \mathsf{HyperLiveness}^{+}$, $\wp(\Sigma^{\omega}) \in \mathsf{HyperSafety}^{\omega} \cap \mathsf{HyperLiveness}^{\omega}$ and $\wp(\Sigma^{\infty}) \in \mathsf{HyperSafety}^{\infty} \cap \mathsf{HyperLiveness}^{\infty}$ (here $\wp(\Sigma^{+})$, $\wp(\Sigma^{\omega})$ and $\wp(\Sigma^{\infty})$ are the $\mathfrak{Hp}^{\mathrm{true}}$ hyperproperties).

4.2.2.3 Topologies for Trace Properties and Hyperproperties

When dealing with infinite computations there is a topological interpretation of safety and liveness, also for the hyper case (see [Clarkson and Schneider, 2010]). In this section, we give topological characterizations of safety/liveness trace properties and hyperproperties which take in consideration finite and mixed computations, other than the infinite ones. For doing so, we define a KCO for each domain (finite, infinite, mixed for trace properties and finite, infinite, mixed for hyperproperties) and then we prove that safety and liveness are closed and dense sets, respectively, in the topology induced by the KCO.

Trace Properties. The function $\mathit{PrfCl} \in \wp(\Sigma^{+}) \to \wp(\Sigma^{+})$, defined as $\mathit{PrfCl} \triangleq \lambda X\,.\,\mathtt{prf}^{\uparrow}(X)$, is a closure on $\wp(\Sigma^{+})$ (indeed it is a KCO). So we have a topology on Σ^{+}, where:

- $\mathfrak{C}_{\Sigma^{+}} = \mathit{PrfCl}^{\uparrow}(\wp(\Sigma^{+})) = \{X \subseteq \Sigma^{+} \mid X = \mathit{PrfCl}(X)\}$ are the closed sets
- $\mathfrak{D}_{\Sigma^{+}} = \{X \subseteq \Sigma^{+} \mid \mathit{PrfCl}(X) = \Sigma^{+}\}$ are the dense sets

Lemma 1. $\mathsf{Safety}^{+} = \mathfrak{C}_{\Sigma^{+}}$ *and* $\mathsf{Liveness}^{+} = \mathfrak{D}_{\Sigma^{+}}$.

Proof. Since elements $X \in \mathsf{Safety}^+$ are prefix-closed, i.e. $X = \mathtt{prf}^\uparrow(X)$, Safety^+ is equal to $\mathfrak{C}_{\Sigma^+}$ by definition. As we already noted, $\forall \bar{\sigma} \in \Sigma^+ \, \exists \bar{\sigma}' \in X \,.\, \bar{\sigma} \leq^{\mathrm{pf}} \bar{\sigma}'$ is equivalent to $\Sigma^+ \subseteq \mathtt{prf}^\uparrow(X)$. But $\mathtt{prf}^\uparrow(X) \subseteq \Sigma^+$, for every $X \in \wp(\Sigma^+)$. Hence $\mathit{PrfCl}(X) = \Sigma^+$, i.e. $X \in \mathfrak{D}_{\Sigma^+}$, if and only if $X \in \mathsf{Liveness}^+$. □

Theorem 9. $\forall \mathfrak{P} \in \wp(\Sigma^+) \, \exists S \in \mathsf{Safety}^+, L \in \mathsf{Liveness}^+ \,.\, \mathfrak{P} = S \cap L$

Proof. It follows from Lemma 1 and Theorem 8. □

Let $\mathtt{li\tilde{m}} \in \wp(\Sigma^\omega) \to \wp(\Sigma^\omega)$ be the function $\lambda X \,.\, \{\bar{\sigma} \in \Sigma^\omega \mid \forall \bar{\sigma}' \in \Sigma^+ \,.\, (\bar{\sigma}' \leq^{\mathrm{pf}} \bar{\sigma} \Rightarrow \bar{\sigma}' \in \mathtt{prf}^\uparrow(X))\}$ returning the set of limit sequences of X. The function $\mathit{LimCl} \in \wp(\Sigma^\omega) \to \wp(\Sigma^\omega)$, defined as $\mathit{LimCl} \triangleq \lambda X \,.\, \mathtt{li\tilde{m}}^\omega(X)$, is a closure on $\wp(\Sigma^\omega)$ (indeed it is a KCO). So we have a topology on Σ^ω, where:

- $\mathfrak{C}_{\Sigma^\omega} = \mathit{LimCl}^\uparrow(\wp(\Sigma^\omega)) = \{X \subseteq \Sigma^\omega \mid X = \mathit{LimCl}(X)\}$ are the closed sets
- $\mathfrak{D}_{\Sigma^\omega} = \{X \subseteq \Sigma^\omega \mid \mathit{LimCl}(X) = \Sigma^\omega\}$ are the dense sets

Lemma 2. $\mathsf{Safety}^\omega = \mathfrak{C}_{\Sigma^\omega}$ *and* $\mathsf{Liveness}^\omega = \mathfrak{D}_{\Sigma^\omega}$.

Proof. Our definitions of safety and liveness (on Σ^ω) are equivalent to the one of [Alpern and Schneider, 1985] and *LimCl* is the limit operator of [Emerson, 1983], so our characterization is equivalent to the usual topological definition of safety/liveness trace properties over Σ^ω. □

Theorem 10. $\forall \mathfrak{P} \in \wp(\Sigma^\omega) \, \exists S \in \mathsf{Safety}^\omega, L \in \mathsf{Liveness}^\omega \,.\, \mathfrak{P} = S \cap L$

Proof. It follows from Lemma 2 and Theorem 8. □

Let $\mathtt{li\tilde{m}}^\infty \in \wp(\Sigma^\infty) \to \wp(\Sigma^\infty)$, defined as $\lambda X \,.\, X \cup \{\bar{\sigma} \in \Sigma^\omega \mid \forall \bar{\sigma}' \in \Sigma^+ \,.\, (\bar{\sigma}' \leq^{\mathrm{pf}} \bar{\sigma} \Rightarrow \bar{\sigma}' \in X)\}$[5], the version on mixed sequences of $\mathtt{li\tilde{m}}$. The function $\mathit{LimPrfCl} \in \wp(\Sigma^\infty) \to \wp(\Sigma^\infty)$, defined as $\mathit{LimPrfCl} \triangleq \lambda X \,.\, \mathtt{li\tilde{m}}^\infty \circ \mathtt{prf}^\uparrow(X)$, is a closure on $\wp(\Sigma^\infty)$ (indeed it is a KCO). So we can define a topology on Σ^∞, where:

- $\mathfrak{C}_{\Sigma^\infty} = \mathit{LimPrfCl}^\uparrow(\wp(\Sigma^\infty)) = \{X \subseteq \Sigma^\infty \mid X = \mathit{LimPrfCl}(X)\}$ are the closed sets
- $\mathfrak{D}_{\Sigma^\infty} = \{X \subseteq \Sigma^\infty \mid \mathit{LimPrfCl}(X) = \Sigma^\infty\}$ are the dense sets

Lemma 3. $\mathsf{Safety}^\infty = \mathfrak{C}_{\Sigma^\infty}$ *and* $\mathsf{Liveness}^\infty = \mathfrak{D}_{\Sigma^\infty}$.

Proof. The proof can be found in Appendix A. □

Theorem 11. $\forall \mathfrak{P} \in \wp(\Sigma^\infty) \, \exists S \in \mathsf{Safety}^\infty, L \in \mathsf{Liveness}^\infty \,.\, \mathfrak{P} = S \cap L$

Proof. It follows from Lemma 3 and Theorem 8. □

[5] $\mathtt{li\tilde{m}}^\infty(X)$ is the Eilenberg-limit [Eilenberg, 1974] of X, i.e. the set $\{\bar{\sigma} \in \Sigma^\omega \mid |\mathtt{prf}(\bar{\sigma}) \cap X| = \infty\}$.

Hyperproperties. The function $\mathit{SprfCl} \in \wp(\wp(\Sigma^+)) \to \wp(\wp(\Sigma^+))$, defined as $\mathit{SprfCl} \triangleq \lambda\mathcal{X} \,.\, \mathtt{sprf}^{\uparrow}(\mathcal{X})$, is a closure on $\wp(\wp(\Sigma^+))$ (indeed it is a KCO). So we can define a topology on $\wp(\Sigma^+)$, where:

- $\mathfrak{C}_{\wp(\Sigma^+)} = \mathit{SprfCl}^{\uparrow}(\wp(\wp(\Sigma^+))) = \{\mathcal{X} \subseteq \wp(\Sigma^+) \mid \mathcal{X} = \mathit{SprfCl}(\mathcal{X})\}$ are the closed sets
- $\mathfrak{D}_{\wp(\Sigma^+)} = \{\mathcal{X} \subseteq \wp(\Sigma^+) \mid \mathit{SprfCl}(\mathcal{X}) = \wp(\Sigma^+)\}$ are the dense sets

Lemma 4. $\mathsf{HyperSafety}^+ = \mathfrak{C}_{\wp(\Sigma^+)}$ *and* $\mathsf{HyperLiveness}^+ = \mathfrak{D}_{\wp(\Sigma^+)}$.

Proof. Elements $\mathcal{X} \in \mathsf{HyperSafety}^+$ are prefixset-closed, i.e. $\mathcal{X} = \mathtt{sprf}^{\uparrow}(\mathcal{X})$, so $\mathsf{HyperSafety}^+$ is equal to $\mathfrak{C}_{\wp(\Sigma^+)}$ by definition. As we already noted, $\forall X \in \wp(\Sigma^+) \,\exists X' \in \mathcal{X} \,.\, X \trianglelefteq^{\mathrm{pf}} X'$ is equivalent to $\wp(\Sigma^+) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$. But $\mathtt{sprf}^{\uparrow}(\mathcal{X}) \subseteq \wp(\Sigma^+)$, for all $\mathcal{X} \in \wp(\wp(\Sigma^+))$. Hence $\mathit{SprfCl}(\mathcal{X}) = \wp(\Sigma^+)$, i.e. $\mathcal{X} \in \mathfrak{D}_{\wp(\Sigma^+)}$, if and only if $\mathcal{X} \in \mathsf{HyperLiveness}^+$. □

Theorem 12. $\forall \mathfrak{H}\mathfrak{p} \in \wp(\wp(\Sigma^+)) \,\exists \mathcal{S} \in \mathsf{HyperSafety}^+, \mathcal{L} \in \mathsf{HyperLiveness}^+ \,.\, \mathfrak{H}\mathfrak{p} = \mathcal{S} \cap \mathcal{L}$

Proof. It follows from Lemma 4 and Theorem 8. □

Let $\mathtt{slim} \in \wp(\wp(\Sigma^\omega)) \to \wp(\wp(\Sigma^\omega))$ be $\lambda\mathcal{X} \,.\, \{Y \in \wp(\Sigma^\omega) \mid \forall Y' \in \wp(\Sigma^+) \,.\, (Y' \trianglelefteq^{\mathrm{pf}} Y \Rightarrow Y' \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X}))\}$, i.e. the function returning the sets of limit sequences of $\mathcal{X}$. Then the function $\mathit{SlimCl} \in \wp(\wp(\Sigma^\omega)) \to \wp(\wp(\Sigma^\omega))$, defined as $\mathit{SlimCl} \triangleq \lambda\mathcal{X} \,.\, \mathtt{slim}^{\omega}(\mathcal{X})$, is a closure on $\wp(\wp(\Sigma^\omega))$ (indeed it is a KCO). So we can define a topology on $\wp(\Sigma^\omega)$, where:

- $\mathfrak{C}_{\wp(\Sigma^\omega)} = \mathit{SlimCl}^{\uparrow}(\wp(\wp(\Sigma^\omega))) = \{\mathcal{X} \subseteq \wp(\Sigma^\omega) \mid \mathcal{X} = \mathit{SlimCl}(\mathcal{X})\}$ are the closed sets
- $\mathfrak{D}_{\wp(\Sigma^\omega)} = \{\mathcal{X} \subseteq \wp(\Sigma^\omega) \mid \mathit{SlimCl}(\mathcal{X}) = \wp(\Sigma^\omega)\}$ are the dense sets

Lemma 5. $\mathsf{HyperSafety}^\omega = \mathfrak{C}_{\wp(\Sigma^\omega)}$ *and* $\mathsf{HyperLiveness}^\omega = \mathfrak{D}_{\wp(\Sigma^\omega)}$.

Proof. The proof can be found in Appendix A. □

Theorem 13. $\forall \mathfrak{H}\mathfrak{p} \in \wp(\wp(\Sigma^\omega)) \,\exists \mathcal{S} \in \mathsf{HyperSafety}^\omega, \mathcal{L} \in \mathsf{HyperLiveness}^\omega \,.\, \mathfrak{H}\mathfrak{p} = \mathcal{S} \cap \mathcal{L}$

Proof. It follows from Lemma 5 and Theorem 8. □

Let $\mathtt{slim}^{\infty} \in \wp(\wp(\Sigma^\infty)) \to \wp(\wp(\Sigma^\infty))$ be $\lambda\mathcal{X} \,.\, \mathcal{X} \cup \{Y \in \wp(\Sigma^\infty) \mid \forall Y' \in \wp(\Sigma^+) \,.\, (Y' \trianglelefteq^{\mathrm{pf}} Y \Rightarrow Y' \in \mathcal{X})\}$, i.e. the version on mixed sequences of $\mathtt{slim}^{\omega}$. Note that here Y is a subset of finite and infinite sequences, not only infinite ones, so we maintain the power to express mixed sets. The function $\mathit{SlimSprfCl} \in \wp(\wp(\Sigma^\infty)) \to \wp(\wp(\Sigma^\infty))$, defined as $\mathit{SlimSprfCl} \triangleq \lambda\mathcal{X} \,.\, \mathtt{slim}^{\infty} \circ \mathtt{sprf}^{\uparrow}(\mathcal{X})$, is a closure on $\wp(\wp(\Sigma^\infty))$ (it is a KCO). So we can define a topology on $\wp(\Sigma^\infty)$, where:

- $\mathfrak{C}_{\wp(\Sigma^\infty)} = \mathit{SlimSprfCl}^{\uparrow}(\wp(\wp(\Sigma^\infty))) = \{\mathcal{X} \subseteq \wp(\Sigma^\infty) \mid \mathcal{X} = \mathit{SlimSprfCl}(\mathcal{X})\}$ are the closed sets
- $\mathfrak{D}_{\wp(\Sigma^\infty)} = \{\mathcal{X} \subseteq \wp(\Sigma^\infty) \mid \mathit{SlimSprfCl}(\mathcal{X}) = \wp(\Sigma^\infty)\}$ are the dense sets

Lemma 6. $\mathsf{HyperSafety}^\infty = \mathfrak{C}_{\wp(\Sigma^\infty)}$ *and* $\mathsf{HyperLiveness}^\infty = \mathfrak{D}_{\wp(\Sigma^\infty)}$.

Proof. The proof can be found in Appendix A. □

Theorem 14. $\forall \mathfrak{H}\mathfrak{p} \in \wp(\wp(\Sigma^\infty)) \,\exists \mathcal{S} \in \mathsf{HyperSafety}^\infty, \mathcal{L} \in \mathsf{HyperLiveness}^\infty \,.\, \mathfrak{H}\mathfrak{p} = \mathcal{S} \cap \mathcal{L}$

Proof. It follows from Lemma 6 and Theorem 8. □

4.2.3 Hyperproperties: Algebraic Characterization

As already said, hyperproperties are sets of sets of executions, hence hyperproperties are in $\wp(\wp(\mathbb{D}\mathbb{e}\mathbb{n}))$, for a given execution denotations domain $\mathbb{D}\mathbb{e}\mathbb{n}$. We denote by $\mathtt{GEN}^{\mathtt{H}}$ the set of all (generic) hyperproperties, namely $\mathtt{GEN}^{\mathtt{H}} \triangleq \wp(\wp(\mathbb{D}\mathbb{e}\mathbb{n}))$. Setting some constraints on the algebraic structure of hyperproperties, we can define some subsets of $\mathtt{GEN}^{\mathtt{H}}$ with particular characteristics. In the next chapters we will show that these restrictions can be useful for verification purposes. The first is the subset-closure [Clarkson and Schneider, 2010].

Definition 25 (Subset-Closed Hyperproperties)**.** An hyperproperty $\mathfrak{Hp} \in \mathtt{GEN}^{\mathtt{H}}$ is *subset-closed* when for every $X \in \wp(\mathbb{D}\mathbb{e}\mathbb{n})$ we have that if X is in $\mathfrak{Hp}$ then every $Y \subseteq X$ is in $\mathfrak{Hp}$.

We denote with $\mathtt{SSC}^{\mathtt{H}}$ the set of all subset-closed hyperproperties, namely $\mathtt{SSC}^{\mathtt{H}}$ is the set $\{\mathfrak{Hp} \in \mathtt{GEN}^{\mathtt{H}} \mid X \in \mathfrak{Hp} \Rightarrow (\forall Y \subseteq X \,.\, Y \in \mathfrak{Hp})\}$, with typical elements $\mathfrak{cHp} \in \mathtt{SSC}^{\mathtt{H}}$. Basically subset-closed hyperproperties are downward-closed, w.r.t. $\subseteq$, sets. The second constraint we *add* is the closure by unions.

Definition 26 (Trace Hyperproperties)**.** A subset-closed hyperproperty $\mathfrak{cHp} \in \mathtt{SSC}^{\mathtt{H}}$ is a *trace hyperproperty* when it is closed under union, namely for every $\mathcal{X} \subseteq \wp(\mathbb{D}\mathbb{e}\mathbb{n})$ we have that if $\mathcal{X}$ is contained in $\mathfrak{cHp}$ then $\bigcup \mathcal{X}$ is in $\mathfrak{cHp}$.

We denote with $\mathtt{TRC}^{\mathtt{H}}$ the set of all trace hyperproperties, namely $\mathtt{TRC}^{\mathtt{H}}$ is the set $\{\mathfrak{cHp} \in \mathtt{SSC}^{\mathtt{H}} \mid \mathcal{X} \subseteq \mathfrak{cHp} \Rightarrow \bigcup \mathcal{X} \in \mathfrak{Hp}\}$, with typical elements $\mathfrak{tHp} \in \mathtt{TRC}^{\mathtt{H}}$.

From an algebraic, i.e. structural, point of view these constraints are equivalent to the following. Without constraints a generic hyperproperty $\mathfrak{Hp} \in \mathtt{GEN}^{\mathtt{H}}$ is identifiable with just a POSET $\langle \mathfrak{Hp}, \subseteq \rangle$. Adding subset-closure means that a subset-closed hyperproperty $\mathfrak{cHp} \in \mathtt{SSC}^{\mathtt{H}}$ is identifiable with a CPO $\langle \mathfrak{cHp}, \subseteq, \cup, \varnothing \rangle$. Indeed every $\mathfrak{cHp}$ has a minimum $\varnothing$, but not necessarily a maximum and the subset-closure guarantees the existence of the supremum of $\subseteq$-chains in $\mathfrak{cHp}$. Finally, a trace hyperproperty $\mathfrak{tHp} \in \mathtt{TRC}^{\mathtt{H}}$ is identifiable with a complete BA $\langle \mathfrak{tHp}, \subseteq, \cup, \cap, \varnothing, \bigcup \mathfrak{tHp}, \setminus \rangle$. Figure 4.7 shows a graphical interpretation of these results. In this setting, the relation $\subseteq$ is the approximation order between hyperproperties (see Chapter 3), in fact if a system satisfies $\mathfrak{Hp}_1$ and $\mathfrak{Hp}_1 \subseteq \mathfrak{Hp}_2$ we have that the system satisfies $\mathfrak{Hp}_2$ as well.

Trace hyperproperties coincide exactly with trace properties. Indeed, let $\mathtt{TRC}^{\mathtt{P}}$ be the set of all trace properties, namely it is the set $\wp(\mathbb{D}\mathbb{e}\mathbb{n})$. Then we have an isomorphism between $\mathtt{TRC}^{\mathtt{H}}$ and $\mathtt{TRC}^{\mathtt{P}}$. This proves that trace properties are particular cases of hyperproperties.

Making a link with the previous subsection, we have that all hypersafety are subset-closed [Clarkson and Schneider, 2010]. Furthermore, all trace hyperproperties are subset-closed, so we have that also some hyperliveness are in $\mathtt{SSC}^{\mathtt{H}}$. The classic liveness properties are trace hyperproperties, hence the corresponding liveness trace hyperproperties are subset-closed. We will see in Chapter 7 that there exists other hyperliveness in $\mathtt{SSC}^{\mathtt{H}}$. Hence we have the diagram in Figure 4.5.

4.2.3.1 Relations between Hyperproperties

Now, we show the relations existing among the notions of hyperproperties we have introduced. Moreover, we describe the algebraic structures of hyperproperties domains. All these results are independent of the choice of the execution denotations in $\mathbb{D}\mathbb{e}\mathbb{n}$.

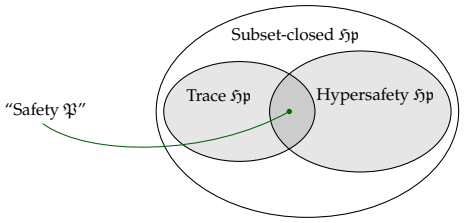


Figure 4.5: Subset-closed hyperproperties.

It is straightforward to note that $\mathtt{TRC}^{\mathtt{H}} \subsetneq \mathtt{SSC}^{\mathtt{H}} \subsetneq \mathtt{GEN}^{\mathtt{H}}$ and that $\mathtt{SSC}^{\mathtt{H}}$ (hence $\mathtt{TRC}^{\mathtt{H}}$) do not contain $\varnothing$. Indeed the empty set has no members, so it cannot be subset-closed. In addition, the unique singleton subset-closed is $\{\varnothing\}$. Now let ρ_{T} be the function $\lambda\mathcal{X} \,.\, \gamma_{\wp} \circ \alpha_{\cup}(\mathcal{X})$, where $\alpha_{\cup} \triangleq \lambda X \,.\, \bigcup X$ and $\gamma_{\wp} \triangleq \lambda X \,.\, \wp(X)$, and let ρ_{S} be the function $\lambda\mathcal{X} \,.\, \{X \mid \exists Y \in \mathcal{X} \,.\, X \subseteq Y\}$. It is easy to note that they are both upper closure operators of $\mathtt{GEN}^{\mathtt{H}}$.

Theorem 15. *Subset-closed hyperproperties are the fixpoints of ρ_{S} and trace hyperproperties are the fixpoints of ρ_{T}, namely $\mathtt{SSC}^{\mathtt{H}} = \rho_{\mathrm{S}}(\mathtt{GEN}^{\mathtt{H}})$ and $\mathtt{TRC}^{\mathtt{H}} = \rho_{\mathrm{T}}(\mathtt{GEN}^{\mathtt{H}})$.*

Proof. $\mathtt{SSC}^{\mathtt{H}} = \rho_{\mathrm{S}}(\mathtt{GEN}^{\mathtt{H}})$ is trivial. For what concerns $\mathtt{TRC}^{\mathtt{H}} = \rho_{\mathrm{T}}(\mathtt{GEN}^{\mathtt{H}})$, for every $\mathcal{X} \in \mathtt{GEN}^{\mathtt{H}}$ we have $\rho_{\mathrm{S}}(\mathcal{X}) \subseteq \rho_{\mathrm{T}}(\mathcal{X})$, hence $\rho_{\mathrm{T}}(\mathtt{GEN}^{\mathtt{H}}) \subseteq \rho_{\mathrm{T}}(\mathtt{GEN}^{\mathtt{H}})$. This proves that $\rho_{\mathrm{T}}(\mathcal{X})$ is subset-closed, for every $\mathcal{X} \in \mathtt{GEN}^{\mathtt{H}}$. Now we have to prove that $\rho_{\mathrm{T}}(\mathcal{X})$ is closed under union. If $\bigcup\mathcal{Y} \subseteq \rho_{\mathrm{T}}(\mathcal{X})$, then $\forall Z \in \mathcal{Y} \,.\, Z \subseteq \bigcup\mathcal{X}$. This latter implies that $\bigcup\mathcal{Y} \subseteq \bigcup\mathcal{X}$ and hence $\bigcup\mathcal{Y} \in \rho_{\mathrm{T}}(\mathcal{X})$. □

Corollary. *A hyperproperty $\mathfrak{Hp}$ is a trace hyperproperty if and only if $\wp(\bigcup\mathfrak{Hp}) = \mathfrak{Hp}$.*

Note that $\langle \mathtt{SSC}^{\mathtt{H}}, \subseteq, \cup, \cap, \{\varnothing\}, \wp(\mathbb{D}\mathtt{em})\rangle$ is a complete lattice, where the bottom is $\{\varnothing\}$ because $\varnothing$ is contained in every subset-closed set and the top is $\wp(\mathbb{D}\mathtt{em})$ because it is the top of $\mathtt{GEN}^{\mathtt{H}}$ and it is subset-closed. For the same reasons they are the bottom and the top of the complete lattice $\langle \mathtt{TRC}^{\mathtt{H}}, \subseteq, \cup, \cap, \{\varnothing\}, \wp(\mathbb{D}\mathtt{em})\rangle$, which is the sublattice of $\mathtt{SSC}^{\mathtt{H}}$ (and $\mathtt{GEN}^{\mathtt{H}}$) comprising its complete boolean algebras. Finally, it is straightforward to note that $\mathtt{TRC}^{\mathtt{H}}$ is isomorphic, through $\langle \alpha_{\cup}, \gamma_{\wp}\rangle$, to $\mathtt{TRC}^{\mathtt{P}}$[6]. The big picture is depicted by the commutative diagram in Figure 4.6. Recall that the approximation order plays the role of implication. So the strongest hyperproperty, i.e. the one which implies any other hyperproperty, is $\varnothing$ for $\mathtt{GEN}^{\mathtt{H}}$ and $\{\varnothing\}$ for $\mathtt{SSC}^{\mathtt{H}}, \mathtt{TRC}^{\mathtt{H}}$. Dually, the weakest hyperproperty, i.e. the one which is implied by any other one, is $\wp(\mathbb{D}\mathtt{em})$ for $\mathtt{GEN}^{\mathtt{H}}, \mathtt{SSC}^{\mathtt{H}}, \mathtt{TRC}^{\mathtt{H}}$. Regarding $\mathtt{TRC}^{\mathtt{P}}$, it is isomorphic to $\mathtt{TRC}^{\mathtt{H}}$ so the strongest trace property is $\alpha_{\cup}(\{\varnothing\}) = \varnothing$ and the weakest is $\alpha_{\cup}(\wp(\mathbb{D}\mathtt{em})) = \mathbb{D}\mathtt{em}$, as expected.

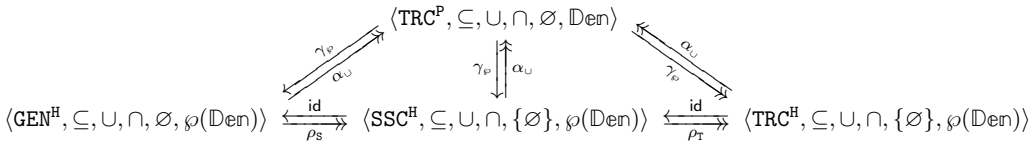


Figure 4.6: Relations between hyperproperties

[6]The adjunction $\langle \alpha_{\cup}, \gamma_{\wp}\rangle$ and its link with systems properties were already introduced in [Assaf et al., 2017] (their $\langle \alpha_{\mathrm{hpp}}, \gamma_{\mathrm{hpp}}\rangle$) and even before in [Cousot and Cousot, 2012] (their $\langle \alpha_{\Theta}, \gamma_{\Theta}\rangle$).

4.2.3.2 Trace Decomposition

Finally, we can provide a further characterization of subset-closed hyperproperties as union of trace hyperproperties.

Proposition 1. *Every subset-closed hyperproperty* $\mathfrak{csHp}$ *can be decomposed into a disjunction of trace hyperproperties, namely, given a* $\mathfrak{csHp} \in \mathtt{SSC}^{\mathtt{H}}$*, we have:*

$$\mathfrak{csHp} = \bigcup_{Y \in \max_{\subseteq}(\mathfrak{csHp})} \wp(Y) \;\; \textit{with} \;\; \max_{\subseteq}(\mathcal{X}) \triangleq \{X \in \mathcal{X} \mid \forall X' \in \mathcal{X} \,.\, X \subseteq X' \Rightarrow X = X'\}$$

where $\max_{\subseteq}(\mathcal{X})$ *is the set of supremum of* $\subseteq$*-chains in* $\mathcal{X}$*.*

Clearly, for all Y in $\max_{\subseteq}(\mathfrak{csHp})$, it holds $\wp(\bigcup \wp(Y)) = \wp(Y)$ so $\wp(Y)$ is a trace hyperproperty. Hence any subset-closed hyperproperty can be characterized as $\mathfrak{csHp} = \bigcup_{i \in \Delta} \mathfrak{tHp}_i$. This implies that, in order to verify $\mathfrak{csHp}$, it is sufficient to verify just one of these $\mathfrak{tHp}_i$. In fact, if $\mathbb{s} \models \mathfrak{tHp}_i$, i.e. $\mathcal{I}(\mathbb{s}) \in \mathfrak{tHp}_i$, then $\mathcal{I}(\mathbb{s}) \in \mathfrak{csHp}$ and hence $\mathbb{s} \models \mathfrak{csHp}$.

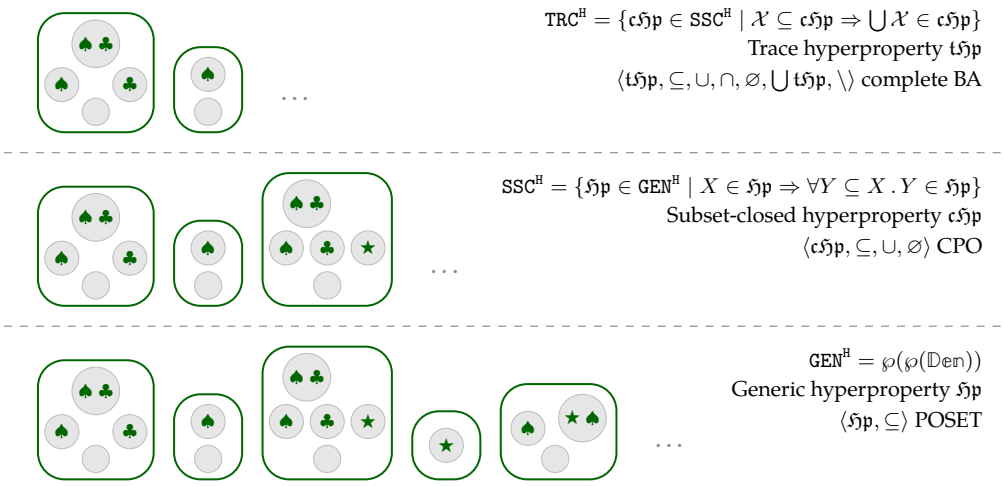


Figure 4.7: Hyperproperties algebraic structure.

5 Classic Program Analysis

In Chapter 3 we have seen the difference between systems verification and analysis, noting that it is possible to perform verification by using an analyzer. In this chapter we focus on the problem of program verification of trace properties. Our systems are computer programs and, given a specification formalized as a trace property, we want to check whether a program satisfies the specification or not. This is what is done in classic program analysis, in the next chapter we will deal with the problem of program verification of the more general hyperproperties. In the sake of simplicity, we take Imp, described in Section 4.1.3, as programming language and, in order to perform verification, we use analysis, as described in Section 3.2.

In general, the verification problem is undecidable, hence we need to move to approximate verification. We use abstract interpretation, in order to design sound approximations, so we need to fix some key concepts. First, we have the *standard semantics*, which is the most precise mathematical *representation* of the program. This is essentially the system model of Chapter 3 and it is used to retrieve the interpretation $\mathcal{I}(\mathsf{P})$ of a system (program) P. In particular, our standard semantics is the transition system $\langle \Sigma^{\mathsf{P}}, \Upsilon^{\mathsf{P}}, \Omega^{\mathsf{P}}, \tau_{\mathsf{P}} \rangle$ associated to the program P (in turn generated by the SOS of Imp). This choice is not mandatory, we could choose as standard semantics a function, a graph or other mathematical entities.

Second, we have the *collecting semantics*, which is the most precise specification that the program satisfies. This latter is derived from the interpretation $\mathcal{I}(\mathsf{P})$ that, in this context, we call *base semantics* and it is denoted as $\mathcal{S}^{\mathsf{P}}_{\text{base}}$. The collecting semantics, potentially, involves the loss of some information about the program standard semantics. Indeed, in [Cousot and Cousot, 1992], the authors say that the collecting semantics is

> *"a version of the standard semantics reduced to essentials in order to ignore irrelevant details about program execution."*

In other words, the collecting semantics is the most precise mathematical description of the program behavior, sufficiently expressive to prove all the possible specifications we are interested in. Since, in this chapter, we are interested in trace properties, the collecting semantics should be a set of execution denotations, representing every possible execution of the program. In this context, the hierarchy of semantics introduced in the previous chapter is very useful. When we set the execution denotations domain, expressing what we are interested in about the program behavior, then the corresponding element in the hierarchy is the base semantics of a program. Furthermore, since we are interested in trace properties, we have for free the collecting semantics, which is indeed equal to the base semantics. In the case of hyperproperties, as we will see in the next chapter, the base semantics and the collecting semantics do not coincide. Suppose we are interested in trace properties expressing a relation between the input and the output of a program (ignoring non-termination),

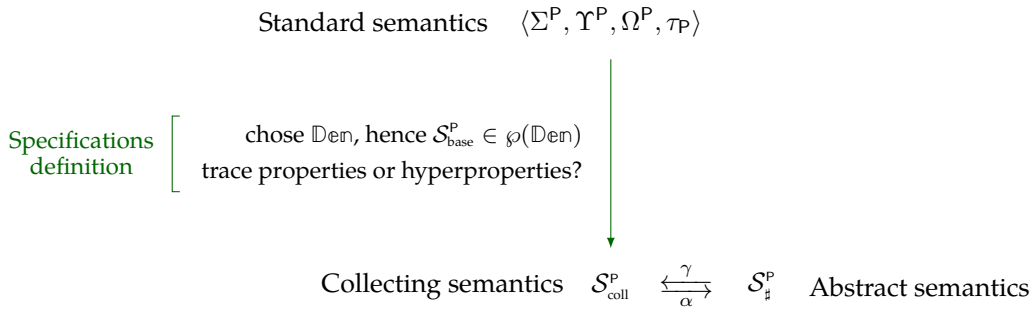


Figure 5.1: Standard, collecting and abstract semantics.

then the execution denotations domain is $\mathbb{Den} = \Sigma^+$. As a consequence, we have that the base semantics is τ^+_{P}. This latter is also the collecting semantics, since all trace properties we are interested in are provable using τ^+_{P}. In other words, τ^+_{P} is complete for trace properties in Σ^+. In the following, we will denote the collecting semantics as $\mathcal{S}^{\mathsf{P}}_{\text{coll}}$, which coincides with $\mathcal{S}^{\mathsf{P}}_{\text{base}}$, and we refer to the semantics of the hierarchy as collecting semantics as well. The hierarchy helps us also in proofs. Indeed, when we define a collecting semantics we need to prove that it exactly computes the trace property we want to verify over the standard semantics. If the execution denotations domain $\mathbb{Den}$ we chose is an abstraction of the most concrete domain $\Sigma^{\vec{\infty}}$, then we have also that the collecting semantics in the hierarchy corresponding to $\mathbb{Den}$ is an abstraction of $\tau^{\vec{\infty}}_{\mathsf{P}}$. This guarantees that the collecting semantics is correct, without exhibiting any proof.

Unfortunately, in general, the collecting semantics is not computable. Hence as a last ingredient, we need the *abstract collecting semantics*, which is an approximation of the collecting semantics, describing only the computable information we can model about a program behavior. Clearly, going abstract we gain computability but we lose precision, namely the abstract semantics is, in general, not complete. In Figure 5.1 we have a graphical representation of the types of semantics we need for program verification. From the standard semantics we chose how to denote the executions generated by the program, i.e. we set $\mathbb{Den}$. Then we have the base semantics, which is the interpretation of the program in $\mathbb{Den}$, corresponding to all possible executions of the program. The collecting semantics is retrieved, depending whether we want to verify trace properties or hyperproperties. Finally, the verification process is made feasible with a computable abstract collecting semantics.

The verification method we take into account is based on over-approximations, this guarantees soundness and, with some assumptions, also decidability. In the chapter, we use the terms specification and trace property interchangeably.

5.1 The Trace Properties Verification Process

The verification process for trace properties is quite straightforward. Setting the execution denotations domain $\mathbb{Den}$, everything we need lies in the domain $\wp(\mathbb{Den})$, i.e. the base semantics, the collecting semantics and trace properties are formalized as subsets of $\mathbb{Den}$. In fact, a trace property is modeled as the set of all executions satisfying it. We denote the set

of all possible trace properties (over $\mathbb{D}\mathtt{en}$) as $\mathtt{TRC}^{\mathsf{P}} \triangleq \wp(\mathbb{D}\mathtt{en})$. Then $\mathcal{S}^{\mathsf{P}}_{\text{base}} \in \wp(\mathbb{D}\mathtt{en})$ is the base semantics of P, chosen from the hierarchy of 4.1, and $\mathcal{S}^{\mathsf{P}}_{\text{coll}} = \mathcal{S}^{\mathsf{P}}_{\text{base}}$ is its collecting semantics. This latter is the strongest program trace property of P over $\mathbb{D}\mathtt{en}$.

Example 6. Suppose we are interested in invariance trace properties. Then $\mathtt{TRC}^{\mathsf{P}} \triangleq \wp(\Sigma)$, where Σ is the set of all possible state denotations, of all programs. Then the base semantics and the strongest program trace property of P (i.e. its collecting semantics) is exactly $\tau^{\mathcal{R}}$, which the most precise invariant satisfied by P.

Examples of trace properties, for $\mathbb{D}\mathtt{en} = \Sigma^{\vec{\infty}}$, are *termination* $\mathtt{Term} \triangleq \Sigma^{\vec{+}}$ and $\mathtt{Even}^{\mathsf{x}} \triangleq \{\bar{\sigma} \in \Sigma^{\vec{\infty}} \mid \forall i > 0 \,.\, (\bar{\sigma}_i = \langle \underline{k}, \mathsf{m}\rangle \wedge \mathsf{m}(\mathsf{x}) \in 2\mathbb{N})\}$ (saying that variable x is always even after initialization). Then we have that P satisfies a trace property $\mathfrak{P} \in \mathtt{TRC}^{\mathsf{P}}$, as usual written $\mathsf{P} \models \mathfrak{P}$, if and only if $\mathcal{S}^{\mathsf{P}}_{\text{coll}} \subseteq \mathfrak{P}$. Hence, by definition, $\mathfrak{P}$ is fulfilled for a system P if and only if $\mathfrak{P}$ is fulfilled for each one of its executions, i.e. $\mathsf{P} \models \mathfrak{P}$ if and only if $\forall \bar{\sigma} \in \mathcal{S}^{\mathsf{P}}_{\text{coll}} \,.\, \bar{\sigma} \in \mathfrak{P}$ (validation). This is quite useful because in order to disprove that a program fulfills a trace property we just need one counterexample, i.e. $\mathsf{P} \not\models \mathfrak{P}$ if and only if $\exists \bar{\sigma} \in \mathcal{S}^{\mathsf{P}}_{\text{coll}} \,.\, \bar{\sigma} \notin \mathfrak{P}$ (confutation). The program in Example 7 satisfies $\mathtt{Even}^{\mathsf{x}}$ but not Term, since $\tau^{\vec{\infty}} \subseteq \mathtt{Even}^{\mathsf{x}}$, while $\tau^{\vec{\infty}} \not\subseteq \mathtt{Term}$.

For practical uses, namely in order to implement a verification mechanism, it is not convenient to define the collecting semantics by means of transition systems. Usually, the collecting semantics is defined inductively from programs syntax. A common choice is to compute the collecting semantics with a denotational semantics (which is compositional by nature), directly defined on the program's code. This means to have a semantic operator $[\![\mathsf{P}]\!] \in \wp(\mathbb{D}\mathtt{en}) \to \wp(\mathbb{D}\mathtt{en})$, possibly involving fixpoint computations, such that $[\![\mathsf{P}]\!]I = \mathcal{S}^{\mathsf{P}}_{\text{coll}}$, for a given $I \in \wp(\mathbb{D}\mathtt{en})$. We will see an example for the maximal trace semantics in Subsection 5.1.1 and for the state semantics is Subsection 5.2.1.

Example 7. Consider the following program:

$$
\begin{array}{l}
\overset{\underline{0}}{}\mathsf{x} := 4 \overset{\underline{1}}{}\,.\,\overset{\underline{1}}{} \\
\mathsf{if}\ \mathsf{y} = 1\ \mathsf{then}\ \{\ \overset{\underline{2}}{}\mathsf{x} := 2 * \mathsf{y} \overset{\underline{3}}{}\ \} \\
\mathsf{else}\ \{ \\
\qquad \overset{\underline{4}}{}\mathsf{while}\ \overset{\underline{5}}{}(\mathsf{tt})\ \{\ \overset{\underline{6}}{}\mathsf{x} := 6 \overset{\underline{7}}{}\ \} \overset{\underline{8}}{} \\
\}\overset{\underline{9}}{}
\end{array}
$$

Let us denote memories $[\mathsf{x} \mapsto n\ \mathsf{y} \mapsto m]$ simply by $[n\ m]$. The maximal trace semantics $\tau^{\vec{\infty}}$ (of the program) and maximal relational semantics τ^{∞} (of the program) are:

$$
\begin{aligned}
\tau^{\vec{\infty}} = \{&\langle \underline{0},[n\ 1]\rangle\langle \underline{1},[4\ 1]\rangle\langle \underline{2},[4\ 1]\rangle\langle \underline{3},[2\ 1]\rangle\langle \underline{9},[2\ 1]\rangle \mid n \in \mathbb{Z}\} \cup \\
&\cup \{\langle \underline{0},[n\ m]\rangle\langle \underline{1},[4\ m]\rangle\langle \underline{4},[4\ m]\rangle\langle \underline{5},[4\ m]\rangle\langle \underline{6},[4\ m]\rangle\langle \underline{7},[6\ m]\rangle\langle \underline{8},[6\ m]\rangle\bar{\sigma} \mid n, m \in \mathbb{Z} \wedge m \neq 1\} \\
\text{where } \bar{\sigma} \in \Sigma^{\vec{\omega}} \text{ is } &\langle \underline{5},[6\ m]\rangle\langle \underline{6},[6\ m]\rangle\langle \underline{7},[6\ m]\rangle\langle \underline{8},[6\ m]\rangle\langle \underline{5},[6\ m]\rangle\langle \underline{6},[6\ m]\rangle\langle \underline{7},[6\ m]\rangle\langle \underline{8},[6\ m]\rangle \ldots
\end{aligned}
$$

$$
\tau^{\infty} = \{\langle\langle \underline{0},[n\ 1]\rangle,\langle \underline{9},[2\ 1]\rangle\rangle \mid n \in \mathbb{Z}\} \cup \{\langle\langle \underline{0},[n\ m]\rangle,\circlearrowleft\rangle \mid n, m \in \mathbb{Z} \wedge m \neq 0\}
$$

Even if we suppose that the collecting semantics is definable starting from program syntax, in general we cannot compute it. The key factor of static analysis is that it must

terminate, at some point, and the collecting semantics does not guarantees this constraint. Furthermore, the collecting semantics is computed over infinite mathematical objects (even a specification $\mathfrak{P}$ is, in general, infinite) so it is not even representable in a computer. We need indeed approximations, in order to make the analysis feasible. Using the theory of abstract interpretation we can define sound approximations of $[\![\mathsf{P}]\!]$ and, in turn of $\mathcal{S}^{\mathsf{P}}_{\text{coll}}$. Recalling Chapter 2, suppose we have an abstract domain $\langle \mathtt{TRC}^{\mathsf{P}}_{\sharp}, \subseteq^{\sharp} \rangle$ approximating concrete trace properties, which forms, paired with $\langle \mathtt{TRC}^{\mathsf{P}}, \subseteq \rangle$ the following Galois connection:

$$\langle \mathtt{TRC}^{\mathsf{P}}, \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathtt{TRC}^{\mathsf{P}}_{\sharp}, \subseteq^{\sharp} \rangle$$

Remember that the Galois connection states the relative precision between abstract specifications, which preserves the relative precision of concrete specifications. Mathematically, we have that $\mathfrak{P}_1 \subseteq \mathfrak{P}_2 \Rightarrow \alpha(\mathfrak{P}_1) \subseteq^{\sharp} \alpha(\mathfrak{P}_2)$ and $\mathfrak{P}^{\sharp}_1 \subseteq^{\sharp} \mathfrak{P}^{\sharp}_2 \Rightarrow \gamma(\mathfrak{P}^{\sharp}_1) \subseteq \gamma(\mathfrak{P}^{\sharp}_2)$. The goal of the analysis is to derive an abstract collecting semantics $\mathcal{S}^{\mathsf{P}}_{\sharp} \in \mathtt{TRC}^{\mathsf{P}}_{\sharp}$ which is sound, i.e. such that it approximates $\mathcal{S}^{\mathsf{P}}_{\text{coll}}$. As we have seen, the collecting semantics is computed by a semantics operator $[\![\mathsf{P}]\!] \in \mathtt{TRC}^{\mathsf{P}} \to \mathtt{TRC}^{\mathsf{P}}$, potentially involving fixpoint computations. Hence, suppose that $\mathcal{S}^{\mathsf{P}}_{\text{coll}} = [\![\mathsf{P}]\!]I$, for a given $I \in \mathtt{TRC}^{\mathsf{P}}$. Then, what we need is an abstract operator $[\![\mathsf{P}]\!]^{\sharp} \in \mathtt{TRC}^{\mathsf{P}}_{\sharp} \to \mathtt{TRC}^{\mathsf{P}}_{\sharp}$, such that, given $I^{\sharp} \in \mathtt{TRC}^{\mathsf{P}}_{\sharp}$ satisfying the condition $I \subseteq \gamma(I^{\sharp})$:

$$[\![\mathsf{P}]\!]I \subseteq \gamma([\![\mathsf{P}]\!]^{\sharp}I^{\sharp}) \text{ or, equivalently, } \alpha([\![\mathsf{P}]\!]I) \subseteq^{\sharp} [\![\mathsf{P}]\!]^{\sharp}I^{\sharp}$$

Then the check $\gamma([\![\mathsf{P}]\!]^{\sharp}I^{\sharp}) \subseteq \mathfrak{P}$ implies $\mathsf{P} \models \mathfrak{P}$. Note that, in order to be effectively implementable, the verification must be done totally in the abstract domain, using an under-approximation of $\mathfrak{P}$. Formally, assuming $\gamma(\mathfrak{P}^{\sharp}) \subseteq \mathfrak{P}$, for a given abstract element $\mathfrak{P}^{\sharp}$, the check becomes: $[\![\mathsf{P}]\!]^{\sharp}I^{\sharp} \subseteq^{\sharp} \mathfrak{P}^{\sharp}$ (which implies the verification due to monotonicity of γ).

If the collecting semantics is computed on a domain different than the approximation domain (as happens for the maximal trace semantics), namely if the semantic operator is not defined on $\langle \mathtt{TRC}^{\mathsf{P}}, \subseteq \rangle$, we have a little extra work to do. Indeed, supposing that α is additive also between $\langle \mathtt{TRC}^{\mathsf{P}}, \leq \rangle$, the computational domain, and $\langle \mathtt{TRC}^{\mathsf{P}}_{\sharp}, \subseteq^{\sharp} \rangle$, we have that its left adjoint always exists. It can be defined as $\gamma^* \triangleq \alpha^- = \lambda Y^{\sharp} \,.\, \bigvee\{X \mid \alpha(X) \subseteq^{\sharp} Y^{\sharp}\}$[1]. Then we have the Galois connection:

$$\langle \mathtt{TRC}^{\mathsf{P}}, \leq \rangle \underset{\alpha}{\overset{\gamma^*}{\leftrightarrows}} \langle \mathtt{TRC}^{\mathsf{P}}_{\sharp}, \subseteq^{\sharp} \rangle$$

Then we can reason as before. Given $I^{\sharp} \in \mathtt{TRC}^{\mathsf{P}}_{\sharp}$ such that $I \leq \gamma^*(I^{\sharp})$:

$$[\![\mathsf{P}]\!]I \leq \gamma^*([\![\mathsf{P}]\!]^{\sharp}I^{\sharp}) \text{ or, equivalently, } \alpha([\![\mathsf{P}]\!]I) \subseteq^{\sharp} [\![\mathsf{P}]\!]^{\sharp}I^{\sharp}$$

Then the check $\gamma([\![\mathsf{P}]\!]^{\sharp}I^{\sharp}) \subseteq \mathfrak{P}$ implies again $\mathsf{P} \models \mathfrak{P}$, in fact:

$$\begin{array}{l}
\gamma([\![\mathsf{P}]\!]^{\sharp}I^{\sharp}) \subseteq \mathfrak{P} \\
\quad \Downarrow \quad \| \text{ reductivity of } \alpha\gamma^* \text{and monotonicity of } \gamma \\
\gamma(\alpha\gamma^*([\![\mathsf{P}]\!]^{\sharp}I^{\sharp})) \subseteq \gamma([\![\mathsf{P}]\!]^{\sharp}I^{\sharp}) \subseteq \mathfrak{P} \\
\quad \Downarrow \quad \| \text{ soundness of } [\![\mathsf{P}]\!]^{\sharp} \text{w.r.t. } \gamma^* \text{and monotonicity of } \alpha,\gamma \\
\gamma\alpha([\![\mathsf{P}]\!]I) \subseteq \gamma(\alpha\gamma^*([\![\mathsf{P}]\!]^{\sharp}I^{\sharp})) \subseteq \gamma([\![\mathsf{P}]\!]^{\sharp}I^{\sharp}) \subseteq \mathfrak{P} \\
\quad \Downarrow \quad \| \text{ extensivity of } \gamma\alpha \\
[\![\mathsf{P}]\!]I \subseteq \gamma\alpha([\![\mathsf{P}]\!]I) \subseteq \gamma(\alpha\gamma^*([\![\mathsf{P}]\!]^{\sharp}I^{\sharp})) \subseteq \gamma([\![\mathsf{P}]\!]^{\sharp}I^{\sharp}) \subseteq \mathfrak{P}
\end{array}$$

[1] Note that we can reason dually, supposing γ co-additive and defining $\alpha^* \triangleq \gamma^+ = \lambda X \,.\, \bigcap^{\sharp}\{Y^{\sharp} \mid X \sqsubseteq \gamma(Y^{\sharp})\}$.

The chosen abstract properties must be machine-representable and the (abstract) computation of the abstract collecting semantics must terminate, namely the verification check is decidable. Galois connection-based abstract interpretations are very useful, from the point of view of analysis design. Indeed, as seen in Chapter 2, Galois connections compose, namely we can always add an abstraction on top of another, and the result is a Galois connection. Furthermore, Galois connections allow us to derive the abstract collecting semantics from the concrete collecting semantics. In fact, the best correct approximation is aways sound. If we have the concrete semantic operator $[\![\mathsf{P}]\!]$, then its best correct approximation $[\![\mathsf{P}]\!]^{\mathrm{bca}} \triangleq \alpha \circ [\![\mathsf{P}]\!] \circ \gamma$ is a correct approximation (also if $[\![\mathsf{P}]\!]$ involves fixpoint computations). In general, even $[\![\mathsf{P}]\!]^{\mathrm{bca}}$ is not computable. But it is useful, since it gives us some hints on how to build the abstract semantics. In any case, once we have derived the abstract semantics $[\![\mathsf{P}]\!]^\sharp$, in order to prove soundness we just need to prove that it approximates $[\![\mathsf{P}]\!]^{\mathrm{bca}}$, namely to prove that $[\![\mathsf{P}]\!]^{\mathrm{bca}} \mathrel{\dot{\subseteq}}^\sharp [\![\mathsf{P}]\!]^\sharp$.

Note that we do not always have a Galois connection, for instance in the case of the Polyhedra abstract domain, for numerical analyses. In these cases, we have to rely on weaker forms of abstract interpretations, based on concretization functions or on abstraction functions. We lose the ability to retrieve the best abstraction of a given concrete property and we lose clearly the ability to retrieve the abstract semantics, by derivation, from the concrete one, exploiting the best correct approximation (which, in general, does not exist). Nevertheless, we still have compositionality.

5.1.1 Computing the Maximal Trace Semantics

As already said, transition systems are very useful from a theoretical point of view but dealing with them in practice is not amenable. So, in order to compute the collecting semantics of interest, we resort to some semantic operators, defined directly on programs syntax. A common approach is to compute the collecting semantics with a denotational semantics inductively defined on the program's code. This is not the only choice, we can use systems of equations, rewriting systems, etc.

We adopt the denotational semantics approach and we show now how to compute the collecting semantics over the denotations domain $\Sigma^{\vec{\infty}}$, namely we show how to compute the maximal trace semantics of programs.

Consider the semantic operator $[\![\mathsf{P}]\!]^{\vec{\infty}} \in \wp(\Sigma^{\vec{\infty}}) \to \wp(\Sigma^{\vec{\infty}})$, computing maximal traces backward, similarly to the general operator $F^{\vec{\infty}}$ defined in Subsection 4.1.1. It is defined for every $\mathsf{P} \in \mathsf{Imp}$ as follow:

$$[\![\mathsf{P}]\!]^{\vec{\infty}}\varnothing \triangleq \varnothing \qquad [\![{}^{\mathtt{i}}\mathsf{c}_1{}^{\mathtt{l}} \,.\, {}^{\mathtt{k}}\mathsf{c}_2{}^{\mathtt{f}}]\!]^{\vec{\infty}} X \triangleq [\![{}^{\mathtt{i}}\mathsf{c}_1{}^{\mathtt{l}}]\!]^{\vec{\infty}} \circ [\![{}^{\mathtt{k}}\mathsf{c}_2{}^{\mathtt{f}}]\!]^{\vec{\infty}} X$$

$$[\![{}^{\mathtt{i}}\mathsf{skip}^{\mathtt{f}}]\!]^{\vec{\infty}} X \triangleq \{\langle \mathtt{i}, \mathsf{m}\rangle\langle \mathtt{f}, \mathsf{m}\rangle\bar{\sigma} \mid \langle \mathtt{f}, \mathsf{m}\rangle\bar{\sigma} \in X \wedge \bar{\sigma} \in \Sigma^{\vec{\infty}}_\epsilon\}$$

$$[\![{}^{\mathtt{i}}\mathsf{x} := \mathsf{a}^{\mathtt{f}}]\!]^{\vec{\infty}} X \triangleq \left\{ \langle \mathtt{i}, \mathsf{m}\rangle\langle \mathtt{f}, \mathsf{m}[\mathsf{x} \mapsto n]\rangle\bar{\sigma} \;\middle|\; \begin{array}{l} \mathsf{m} \in \mathsf{Mem} \wedge \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n \wedge \\ \langle \mathtt{f}, \mathsf{m}[\mathsf{x} \mapsto n]\rangle\bar{\sigma} \in X \wedge \bar{\sigma} \in \Sigma^{\vec{\infty}}_\epsilon \end{array} \right\}$$

$$[\![{}^{\mathtt{i}}\mathsf{if\ b\ then\ \{\,P_1\,\}\ else\ \{\,P_2\,\}}^{\mathtt{f}}]\!]^{\vec{\infty}} X \triangleq$$

$$\left\{ \langle \mathtt{i}, \mathsf{m}\rangle\bar{\sigma} \;\middle|\; \begin{array}{l} \langle \mathtt{l}, \mathsf{m}\rangle\bar{\sigma} \in [\![\mathsf{P}_1]\!]^{\vec{\infty}}\{\langle \mathtt{k}, \mathsf{n}\rangle\langle \mathtt{f}, \mathsf{n}\rangle\bar{\sigma}' \mid \langle \mathtt{f}, \mathsf{n}\rangle\bar{\sigma} \in X \wedge \bar{\sigma}' \in \Sigma^{\vec{\infty}}_\epsilon\} \\ \wedge \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt} \wedge \bar{\sigma} \in \Sigma^{\vec{\infty}} \wedge \mathsf{P}_1 = {}^{\mathtt{l}}\mathsf{c}^{\mathtt{k}} \end{array} \right\} \cup$$

$$\cup \left\{ \langle \underline{\mathtt{i}}, \mathsf{m}\rangle\langle \underline{\mathtt{l}}, \mathsf{m}\rangle\bar{\sigma} \;\middle|\; \begin{array}{l} \langle \underline{\mathtt{l}}, \mathsf{m}\rangle\bar{\sigma} \in [\![\mathsf{P}_2]\!]^{\vec{\infty}}\{\langle \underline{\mathtt{k}}, \mathsf{n}\rangle\langle \underline{\mathtt{f}}, \mathsf{n}\rangle\bar{\sigma}' \mid \langle \underline{\mathtt{f}}, \mathsf{n}\rangle\bar{\sigma} \in X \wedge \bar{\sigma}' \in \Sigma^{\vec{\infty}}_{\epsilon}\} \\ \wedge\, \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{ff} \wedge \bar{\sigma} \in \Sigma^{\vec{\infty}} \wedge \mathsf{P}_2 = {}^{\underline{\mathtt{l}}}\mathsf{c}^{\underline{\mathtt{k}}} \end{array} \right\}$$

$$[\![{}^{\underline{\mathtt{i}}}\mathsf{while}\ {}^{\underline{\mathtt{l}}}\mathsf{b}\ \{\ \mathsf{P}\ \}^{\underline{\mathtt{f}}}]\!]^{\vec{\infty}}X \triangleq \{\langle \underline{\mathtt{i}}, \mathsf{m}\rangle\bar{\sigma} \mid \bar{\sigma} \in \mathsf{lfp}^{\sqsubseteq}_{\Sigma^{\vec{\omega}}}\, \mathsf{F}^{\vec{\infty}}\} \text{ where:}$$

$$\mathsf{F}^{\vec{\infty}}(T) \triangleq \left\{ \langle \underline{\mathtt{l}}, \mathsf{m}\rangle\langle \underline{\mathtt{k}}, \mathsf{m}\rangle\bar{\sigma} \;\middle|\; \begin{array}{l} \langle \underline{\mathtt{k}}, \mathsf{m}\rangle\bar{\sigma} \in [\![\mathsf{P}]\!]^{\vec{\infty}}\{\langle \underline{\mathtt{t}}, \mathsf{n}\rangle\bar{\sigma}' \mid \langle \underline{\mathtt{l}}, \mathsf{n}\rangle\bar{\sigma} \in T \wedge \bar{\sigma}' \in \Sigma^{\vec{\infty}}_{\epsilon}\} \\ \wedge\, \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt} \wedge \bar{\sigma} \in \Sigma^{\vec{\infty}} \wedge \mathsf{P} = {}^{\underline{\mathtt{k}}}\mathsf{c}^{\underline{\mathtt{t}}} \end{array} \right\} \cup$$

$$\cup \left\{ \langle \underline{\mathtt{l}}, \mathsf{m}\rangle\langle \underline{\mathtt{f}}, \mathsf{m}\rangle\bar{\sigma} \;\middle|\; \langle \underline{\mathtt{f}}, \mathsf{m}\rangle\bar{\sigma} \in X \wedge \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{ff} \wedge \bar{\sigma} \in \Sigma^{\vec{\infty}} \right\}$$

The function $\mathsf{F}^{\vec{\infty}} \in \wp(\Sigma^{\vec{\infty}}) \to \wp(\Sigma^{\vec{\infty}})$ is monotone on the CPO $\langle \wp(\Sigma^{\vec{\infty}}), \sqsubseteq, \sqcup, \Sigma^{\vec{\omega}}\rangle$ (the one defined for $\tau^{\vec{\infty}}$ in Subsection 4.1.1), hence its least fixpoint $\bigsqcup_{n<\omega} \mathsf{F}^{\vec{\infty}\,n}(\Sigma^{\vec{\omega}})$ exists. Indeed, the maximal trace semantics $\tau^{\vec{\infty}}$ of P coincides with $[\![\mathsf{P}]\!]^{\vec{\infty}}F$. Here F is the set $\{\langle \underline{\mathtt{f}}, \mathsf{m}\rangle \mid \mathsf{m} \in \mathsf{Mem}^{\mathsf{P}}\}$, assuming $\mathsf{P} = {}^{\underline{\mathtt{i}}}\mathsf{c}^{\underline{\mathtt{f}}}$. Note that this operator computes the maximal traces semantics without the traces not starting in an initial state, as pointed out at the end of Subsection 4.1.3. As already pointed out, this semantics is, in general, not implementable: in the function $\mathsf{F}^{\vec{\infty}}$ we have that ω is a transfinite ordinal and hence it is not guaranteed that the fixpoint is reached in finite time.

5.2 Invariants Verification

In this section, we will go deeper in the definition of a program analysis for trace properties, based on abstract interpretation. The concepts introduced here are important, since we will adapt them for the verification of hyperproperties in the next chapter. In the following example, our focus is on *invariance properties*, sometimes called state properties, or simply invariants. These latter are particular trace properties totally dispensing from the execution history: they are predicates on the reachable states. In the (extended) hierarchy, they are exactly defined in terms of the state semantics $\tau^{\mathcal{R}} \subseteq \Sigma$. Examples of invariance properties are "there are no division-by-zero errors in the program" or "variable j is always within a given range" (if j is an index of an array, it is important to check if it could generate array-out-of-bound errors), etc. As for the maximal trace semantics, we need a way for expressing this semantics recursively on program syntax.

5.2.1 Computing the State Semantics

Again we use a denotational semantics, in order to compute the state semantics. Consider the semantic operator $[\![\mathsf{P}]\!]^{\mathcal{R}} \in \wp(\Sigma) \to \wp(\Sigma)$, computing the reachable states forward, similarly to the general operator $F^{\mathcal{R}}$ defined in Subsection 4.1.2. It is defined for every $\mathsf{P} \in \mathsf{Imp}$ as follow:

$$[\![\mathsf{P}]\!]^{\mathcal{R}}\varnothing \triangleq \varnothing \qquad [\![{}^{\underline{\mathtt{i}}}\mathsf{c}_1\,{}^{\underline{\mathtt{l}}}\,;\,{}^{\underline{\mathtt{k}}}\mathsf{c}_2{}^{\underline{\mathtt{f}}}]\!]^{\mathcal{R}}X \triangleq X \cup [\![{}^{\underline{\mathtt{k}}}\mathsf{c}_2{}^{\underline{\mathtt{f}}}]\!]^{\mathcal{R}} \circ [\![{}^{\underline{\mathtt{i}}}\mathsf{c}_1\,{}^{\underline{\mathtt{l}}}]\!]^{\mathcal{R}}X$$

$$[\![{}^{\underline{\mathtt{i}}}\mathsf{skip}^{\underline{\mathtt{f}}}]\!]^{\mathcal{R}}X \triangleq X \cup \{\langle \underline{\mathtt{f}}, \mathsf{m}\rangle \mid \langle \underline{\mathtt{i}}, \mathsf{m}\rangle \in X\}$$

$$[\![{}^{\underline{\mathtt{i}}}\mathsf{x} := \mathsf{a}^{\underline{\mathtt{f}}}]\!]^{\mathcal{R}}X \triangleq X \cup \{\langle \underline{\mathtt{f}}, \mathsf{m}[\mathsf{x} \mapsto n]\rangle \mid \langle \underline{\mathtt{i}}, \mathsf{m}\rangle \wedge \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\}$$

$$[\![{}^{\underline{\mathtt{i}}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\ \mathsf{P}_1\ \}\ \mathsf{else}\ \{\ \mathsf{P}_2\ \}^{\underline{\mathtt{f}}}]\!]^{\mathcal{R}}X \triangleq X \cup Y_1 \cup Y_2 \cup \{\langle \underline{\mathtt{f}}, \mathsf{m}\rangle \mid \langle \underline{\mathtt{t}}, \mathsf{m}\rangle \in Y_1 \cup Y_2\} \text{ with:}$$

$$\mathsf{P}_1 = {}^{\underline{\mathsf{l}}}\mathsf{c}^{\underline{\mathsf{k}}} \quad \mathsf{P}_2 = {}^{\underline{\mathsf{h}}}\mathsf{c}^{\underline{\mathsf{d}}} \quad \underline{\mathsf{t}} \in \{\underline{\mathsf{k}}, \underline{\mathsf{d}}\}$$

$$Y_1 = [\![\mathsf{P}_1]\!]^{\mathcal{R}}\{\langle \underline{\mathsf{l}}, \mathsf{m}\rangle \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt} \wedge \langle \underline{\mathsf{i}}, \mathsf{m}\rangle \in X\}$$

$$Y_2 = [\![\mathsf{P}_2]\!]^{\mathcal{R}}\{\langle \underline{\mathsf{h}}, \mathsf{m}\rangle \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{ff} \wedge \langle \underline{\mathsf{i}}, \mathsf{m}\rangle \in X\}$$

$$[\![{}^{\underline{\mathsf{i}}}\mathsf{while}\ {}^{\underline{\mathsf{l}}}\mathsf{b}\ \{\ \mathsf{P}\ \}^{\underline{\mathsf{f}}}]\!]^{\mathcal{R}} X \triangleq X \cup \mathsf{lfp}^{\subseteq}_{\varnothing}\, \mathsf{F}^{\mathcal{R}} \cup \{\langle \underline{\mathsf{f}}, \mathsf{m}\rangle \mid \langle \underline{\mathsf{l}}, \mathsf{m}\rangle \in \mathsf{lfp}^{\subseteq}_{\varnothing}\, \mathsf{F}^{\mathcal{R}} \wedge \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{ff}\}$$ with:

$$\mathsf{F}^{\mathcal{R}}(T) \triangleq \{\langle \underline{\mathsf{l}}, \mathsf{m}\rangle \mid \langle \underline{\mathsf{i}}, \mathsf{m}\rangle \in X\} \cup Y \cup \{\langle \underline{\mathsf{l}}, \mathsf{m}\rangle \mid \langle \underline{\mathsf{t}}, \mathsf{m}\rangle \in Y\}$$ where:

$$\mathsf{P} = {}^{\underline{\mathsf{k}}}\mathsf{c}^{\underline{\mathsf{t}}} \quad Y = [\![\mathsf{P}]\!]^{\mathcal{R}}\{\langle \underline{\mathsf{k}}, \mathsf{m}\rangle \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt} \wedge \langle \underline{\mathsf{l}}, \mathsf{m}\rangle \in T\}$$

The function $\mathsf{F}^{\mathcal{R}} \in \wp(\Sigma) \to \wp(\Sigma)$ is monotone on the CPO $\langle \wp(\Sigma), \subseteq, \cup, \varnothing\rangle$, hence its least fixpoint $\bigcup_{n<\omega} \mathsf{F}^{\mathcal{R}^n}(\varnothing)$ exists. Indeed, the state semantics $\tau^{\mathcal{R}}$ of P coincides with $[\![\mathsf{P}]\!]^{\mathcal{R}} I$. Here I is the set $\{\langle \underline{\mathsf{i}}, \mathsf{m}\rangle \mid \mathsf{m} \in \mathsf{Mem}^{\mathsf{P}}\}$, assuming $\mathsf{P} = {}^{\underline{\mathsf{i}}}\mathsf{c}^{\underline{\mathsf{f}}}$.

In static analysis, very often this semantics is computed in an alternative, yet isomorphic, form. The state semantics can be isomorphically represented, exploiting the following Galois isomorphism:

$$\langle \wp(\Sigma), \subseteq\rangle \underset{\alpha_\ell}{\overset{\gamma_\ell}{\leftrightarrows}} \langle \mathsf{Lab} \to \wp(\mathsf{Mem}), \dot{\subseteq}\rangle$$

$$\alpha_\ell(X) \triangleq \lambda \underline{\mathsf{l}}\,.\,\{\mathsf{m} \in \mathsf{Mem} \mid \langle \underline{\mathsf{l}}, \mathsf{m}\rangle \in X\} \qquad \gamma_\ell(f) \triangleq \{\langle \underline{\mathsf{l}}, \mathsf{m}\rangle \in \Sigma \mid \mathsf{m} \in f(\underline{\mathsf{l}})\}$$

We denote with $\tau^{\mathcal{R}}_\ell$ the state semantics $\tau^{\mathcal{R}}_\ell \triangleq \alpha_\ell(\tau^{\mathcal{R}})$. This semantics associates a *memory invariant* for every program control point. It could be computed inductively on programs syntax. Usually it is defined either with a semantic operator $[\![\mathsf{P}]\!]^{\mathcal{R}}_\ell \in (\mathsf{Lab} \to \wp(\mathsf{Mem})) \to (\mathsf{Lab} \to \wp(\mathsf{Mem}))$, similar to $[\![\mathsf{P}]\!]^{\mathcal{R}}$, or with a system of equations:

$$\left(\mathsf{x}_{\underline{\mathsf{k}}} = \mathsf{F}^{\mathrm{eq}}_{\underline{\mathsf{k}}}(\mathsf{x}_{\underline{\mathsf{0}}}, \mathsf{x}_{\underline{\mathsf{1}}}, \ldots \mathsf{x}_{\underline{\mathsf{k}}}, \ldots \mathsf{x}_{\underline{\mathsf{h}}})\right)_{\underline{\mathsf{k}} \in \mathsf{Lab}^{\mathsf{P}}} \text{ assuming } \mathsf{Lab}^{\mathsf{P}} = \{\underline{\mathsf{0}}, \underline{\mathsf{1}}, \ldots, \underline{\mathsf{h}}\}$$

The least solution of the system of equations coincides with $[\![\mathsf{P}]\!]^{\mathcal{R}}_\ell \alpha_\ell(I)$ and, in turn, we have that $\gamma_\ell([\![\mathsf{P}]\!]^{\mathcal{R}}_\ell \alpha_\ell(I))$ is equal to $[\![\mathsf{P}]\!]^{\mathcal{R}} I$. Both this ways of computing the state semantics have to represent in memory the invariant for every program control point, hence they are computationally expensive. It is then a common practice to use a *post-condition* semantics $\tau^{\curvearrowright} \in \wp(\mathsf{Mem})$, computing the invariant only of the last control point. This allows to reduce the memory space used by the analyzer by a factor $|\mathsf{Lab}^{\mathsf{P}}|$, but clearly we loose the information of the intermediate control points. This is an abstraction of the state semantics, through the following Galois connection, parametric on the program $\mathsf{P} = {}^{\underline{\mathsf{i}}}\mathsf{c}^{\underline{\mathsf{f}}}$:

$$\langle \wp(\Sigma), \subseteq\rangle \underset{\alpha_{\underline{\mathsf{f}}}}{\overset{\gamma_{\underline{\mathsf{f}}}}{\leftrightarrows}} \langle \wp(\mathsf{Mem}), \subseteq\rangle$$

$$\alpha_{\underline{\mathsf{f}}} \triangleq \lambda X\,.\,\{\mathsf{m} \mid \langle \underline{\mathsf{f}}, \mathsf{m}\rangle \in X\} \qquad \gamma_{\underline{\mathsf{f}}} \triangleq \lambda X\,.\,\{\langle \underline{\mathsf{f}}, \mathsf{m}\rangle \mid \mathsf{m} \in X\} \cup (\mathsf{Lab}^{\mathsf{P}} \setminus \{\underline{\mathsf{f}}\}) \times \mathsf{Mem}^{\mathsf{P}}$$

Then we have that $\tau^{\curvearrowright} = \alpha_{\underline{\mathsf{f}}}(\tau^{\mathcal{R}})$. The semantics can be computed directly on programs syntax by the operator $[\![\mathsf{P}]\!]^{\curvearrowright} \in \wp(\mathsf{Mem}) \to \wp(\mathsf{Mem})$, defined for every $\mathsf{P} \in \mathsf{Imp}$ as follow:

$$[\![\mathsf{P}]\!]^{\curvearrowright}\varnothing \triangleq \varnothing \qquad [\![{}^{\underline{\mathsf{i}}}\mathsf{c}_1{}^{\underline{\mathsf{l}}}\,.\,{}^{\underline{\mathsf{k}}}\mathsf{c}_2{}^{\underline{\mathsf{f}}}]\!]^{\curvearrowright} X \triangleq [\![{}^{\underline{\mathsf{k}}}\mathsf{c}_2{}^{\underline{\mathsf{f}}}]\!]^{\curvearrowright} \circ [\![{}^{\underline{\mathsf{i}}}\mathsf{c}_1{}^{\underline{\mathsf{l}}}]\!]^{\curvearrowright} X \qquad [\![{}^{\underline{\mathsf{i}}}\mathsf{skip}^{\underline{\mathsf{f}}}]\!]^{\curvearrowright} X \triangleq X$$

$$[\![{}^{\underline{\mathsf{i}}}\mathsf{x} := \mathsf{a}^{\underline{\mathsf{f}}}]\!]^{\curvearrowright} X \triangleq \{\mathsf{m}[\mathsf{x} \leftarrow n] \mid \mathsf{m} \in X \wedge \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\}$$

$$[\![{}^{\underline{\mathsf{i}}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\ \mathsf{P}_1\ \}\ \mathsf{else}\ \{\ \mathsf{P}_2\ \}^{\underline{\mathsf{f}}}]\!]^{\curvearrowright} X \triangleq [\![\mathsf{P}_1]\!]^{\curvearrowright}\{\mathsf{m} \in X \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}\{\mathsf{m} \in X \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{ff}\}$$

$$[\![{}^{\underline{\mathsf{i}}}\mathsf{while}\ {}^{\underline{\mathsf{l}}}\mathsf{b}\ \{\ \mathsf{P}\ \}^{\underline{\mathsf{f}}}]\!]^{\curvearrowright} X \triangleq \{\mathsf{m} \in \mathsf{lfp}^{\subseteq}_{\varnothing}\, \mathsf{F}^{\curvearrowright} \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{ff}\}$$ where:

$$\mathsf{F}^{\curvearrowright}(T) \triangleq X \cup [\![\mathsf{P}]\!]^{\curvearrowright}\{\mathsf{m} \in T \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\}$$

Again, $\mathsf{F}^{\curvearrowright} \in \wp(\mathsf{Mem}) \to \wp(\mathsf{Mem})$ is monotone on the CPO $\langle \wp(\mathsf{Mem}), \subseteq, \cup, \varnothing \rangle$, hence its least fixpoint $\bigcup_{n<\omega} \mathsf{F}^{\curvearrowright n}(\varnothing)$ exists. It is easy to note that for every program $\mathsf{P} = {}^{\mathtt{i}}\mathsf{c}^{\mathtt{f}}$ we have that $[\![\mathsf{P}]\!]^{\curvearrowright}\{\mathsf{m} \mid \langle \mathtt{i}, \mathsf{m}\rangle \in \tau^{\mathcal{R}}\} = \{\mathsf{m} \mid \langle \mathtt{f}, \mathsf{m}\rangle \in \tau^{\mathcal{R}}\}$. Basically this semantics, given a pre-condition (on memories) outputs its strongest post-condition (on memories).

The post-condition semantics can be used to compute $\tau^{\mathcal{R}}_{\ell}$. Indeed, if a program $\mathsf{P} = {}^{\mathtt{i}}\mathsf{c}^{\mathtt{f}}$ is not a sub-program (of another program), then $\tau^{\mathcal{R}}_{\ell}(\mathtt{i}) = \mathsf{Mem}^{\mathsf{P}}$, by definition. If a program $\mathsf{P} = {}^{\mathtt{i}}\mathsf{c}^{\mathtt{f}}$ is contained in a bigger program, we have the following cases. If the containing program is ${}^{\mathtt{l}}\mathsf{c}_1{}^{\mathtt{k}}.\mathsf{P}$, then $\tau^{\mathcal{R}}_{\ell}(\mathtt{i}) = \tau^{\mathcal{R}}_{\ell}(\mathtt{k})$. If the containing program is ${}^{\mathtt{l}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\mathsf{P}\}\ \mathsf{else}\ \{\mathsf{P}_2\}^{\mathtt{k}}$, then $\tau^{\mathcal{R}}_{\ell}(\mathtt{i}) = \{\mathsf{m} \in \tau^{\mathcal{R}}_{\ell}(\mathtt{l}) \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\}$. If the containing program is ${}^{\mathtt{l}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\mathsf{P}_1\}\ \mathsf{else}\ \{\mathsf{P}\}^{\mathtt{k}}$, then $\tau^{\mathcal{R}}_{\ell}(\mathtt{i}) = \{\mathsf{m} \in \tau^{\mathcal{R}}_{\ell}(\mathtt{l}) \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{ff}\}$. Finally, if the containing program is ${}^{\mathtt{l}}\mathsf{while}\ {}^{\mathtt{t}}\mathsf{b}\ \{\mathsf{P}\}^{\mathtt{k}}$, then $\tau^{\mathcal{R}}_{\ell}(\mathtt{i}) = \{\mathsf{m}\tau^{\mathcal{R}}_{\ell}(\mathtt{t}) \in \ \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\}$ where $\tau^{\mathcal{R}}_{\ell}(\mathtt{t}) = \mathsf{lfp}^{\subseteq}_{\varnothing} \mathsf{F}^{\curvearrowright}$ and $\mathsf{F}^{\curvearrowright}(T) \triangleq \tau^{\mathcal{R}}_{\ell}(\mathtt{l}) \cup [\![\mathsf{P}]\!]^{\curvearrowright}\{\mathsf{m} \in T \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\}$. In any case, $\tau^{\mathcal{R}}_{\ell}(\mathtt{f}) = [\![\mathsf{P}]\!]^{\curvearrowright}\tau^{\mathcal{R}}_{\ell}(\mathtt{i})$.

5.2.2 Application: Non-Relational Numerical Analysis

Suppose we are interested in non-relational numerical invariants of variables. For instance, suppose to be interested in verifying that a variable x is always inside a given range, i.e. $n \leq \mathsf{m}(\mathsf{x}) \leq m$, given the numerical bounds $n, m \in \mathbb{N}$. Numerical invariants are state properties of memories, which have to hold for any program label[2], hence the execution denotations domain is Σ (recall that $\Sigma = \mathsf{Lab} \times \mathsf{Mem}$). A state property is in $\wp(\Sigma)$ and indeed, $\mathtt{inv}^{[n,m]}_{\mathsf{x}} \triangleq \{\langle \mathtt{l}, \mathsf{m}\rangle \mid n \leq \mathsf{m}(\mathsf{x}) \leq m\}$ is the property stated before.

In this case the strongest program property of P is the state semantics, $\tau^{\mathcal{R}}$. This latter is computable with the semantic operator $[\![\mathsf{P}]\!]^{\mathcal{R}}$ (the computational domain $\langle \wp(\Sigma), \subseteq, \cup, \varnothing\rangle$ coincides with the approximation domain). In order to define the abstract collecting semantics we proceed as follows.

First, we change the representation of sets of states, as done in the previous section. Hence, $\alpha_{\ell}(\tau^{\mathcal{R}})$ is a map associating a set of memories to every program control point. Indeed, $\alpha_{\ell}(\tau^{\mathcal{R}})(\mathtt{l})$ is the most precise (memories) invariant of the program, at label $\mathtt{l}$. We could compute $\alpha_{\ell}(\tau^{\mathcal{R}})$ inductively on program's syntax, with the semantic operator $[\![\mathsf{P}]\!]^{\mathcal{R}}_{\ell}$. This function can be derived by calculus, approximating the best correct approximation of $[\![\mathsf{P}]\!]^{\mathcal{R}}$, namely the function $\alpha_{\ell} \circ [\![\mathsf{P}]\!]^{\mathcal{R}} \circ \gamma_{\ell}$. We prefer to use an abstract post-conditions semantics and compute the numerical invariant for each program control point as done at the end of the previous subsection, but in the abstract.

In order to do so, we have to define a numerical abstract domain approximating $\wp(\mathsf{Mem})$. Basically, we need to define a Galois connection (if possible)

$$\langle \wp(\mathsf{Mem}), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \preccurlyeq\rangle$$

where A is computer-representable, the test $a_1 \preccurlyeq a_2$ is decidable and the (binary) join $a_1 \curlyvee a_2$ is computable[3]. For instance, consider the Intervals domain defined as follow.

[2] Note that one can define invariants sensitive to programs label. In that case the invariant is a state property of program states (with labels).

[3] In reality, in order to define an implementable static analysis, we need other assumptions.

Definition 27 (Intervals Domain)**.** The domain of Intervals has as carrier set

$$\text{Intv} \triangleq \{[a,b] \mid a \in \mathbb{Z} \cup \{-\infty\} \wedge b \in \mathbb{Z} \cup \{+\infty\} \wedge a \leq b\} \cup [+\infty, -\infty]$$

Consider the usual ordering $\leq$ between integers, extended with the pairs in $\{\langle -\infty, n\rangle \mid n \in \mathbb{Z}\} \cup \{\langle n, +\infty\rangle \mid n \in \mathbb{Z}\} \cup \{\langle -\infty, -\infty\rangle, \langle -\infty, +\infty\rangle, \langle +\infty, +\infty\rangle\}$. Then the relation $\sqsubseteq^\imath \subseteq \text{Intv} \times \text{Intv}$, defined as $[a,b] \sqsubseteq^\imath [c,d] \triangleq (c \leq a \wedge b \leq d)$, is a partial order. The domain has arbitrary least upper bounds and greatest lower bounds, defined as:

$$\bigsqcup\nolimits^\imath \{[a_i, b_i]\}_{i\in\Delta} \triangleq [\min\{a_i\}_{i\in\Delta}, \max\{b_i\}_{i\in\Delta}]$$

$$\bigsqcap\nolimits^\imath \{[a_i, b_i]\}_{i\in\Delta} \triangleq \begin{cases} [+\infty, -\infty] & \text{if } \max\{a_i\}_{i\in\Delta} \leq \min\{b_i\}_{i\in\Delta} \\ [\max\{a_i\}_{i\in\Delta}, \min\{b_i\}_{i\in\Delta}] & \text{otherwise} \end{cases}$$

Where $\max X = +\infty$ if and only if $+\infty \in X$, $\max X = -\infty$ if and only if $X = \{-\infty\}$, $\min X = -\infty$ if and only if $-\infty \in X$ and $\min X = +\infty$ if and only if $\{+\infty\}$. The domain has a minimum $[+\infty, -\infty]$, that we denote $\bot^\imath$, and a maximum $[-\infty, +\infty]$, that we denote $\top^\imath$. The domain is, indeed, a complete lattice.

Remark. The Intervals domain is a complete lattice but it is not ACC. For instance, the increasing chain $\{[0,n] \mid n \in \mathbb{N}\}$ is infinite. This means that an analysis with the Intervals domain may not terminate in finite time. In order to force convergence, this domain needs an extrapolation operator, as explained in Chapter 2. For instance, a possible widening for Intervals [Cousot and Cousot, 1976] is

$$[a,b] \triangledown^\imath [c,d] \triangleq [\!(\, a \leq c \,?\, a \,:\, -\infty \,)\!], \!(\, d \leq b \,?\, b \,:\, +\infty \,)\!]$$
$$\bot^\imath \triangledown^\imath [a,b] \triangleq [a,b] \qquad [a,b] \triangledown^\imath \bot^\imath \triangleq [a,b]$$

In the following, in order to keep things simple, we avoid the introduction of such an operator in the abstract semantics.

This domain composes the following Galois connection with $\wp(\mathbb{Z})$, i.e. sets of values:

$$\langle \wp(\mathbb{Z}), \subseteq\rangle \underset{\alpha_\imath}{\overset{\gamma_\imath}{\leftrightarrows}} \langle \text{Intv}, \sqsubseteq^\imath\rangle$$
$$\alpha_\imath(X) \triangleq [\!(\, X \neq \varnothing \,?\, [\min X, \max X] \,:\, \bot^\imath \,)\!] \quad \gamma_\imath([a,b]) \triangleq \{n \in \mathbb{Z} \mid a \leq n \leq b\}$$

Before to apply the Intervals abstraction we need an intermediate Non-Relational Abstraction (Section 2.3). This latter abstracts each variable independently, thus forgetting any relation between variables. Let $\dot{\mathsf{Mem}} \triangleq \mathsf{Var} \to \wp(\mathbb{Z})$. We have the Galois insertion

$$\langle \wp(\mathsf{Mem}), \subseteq\rangle \underset{\alpha_{nr}}{\overset{\gamma_{nr}}{\leftrightarrows\!\!\!\!\!\!\rightarrow}} \langle \dot{\mathsf{Mem}}, \dot{\subseteq}\rangle$$
$$\alpha_{nr}(X) \triangleq \lambda \mathsf{x} \,.\, \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X\} \quad \gamma_{nr}(\dot{\mathsf{m}}) \triangleq \{\mathsf{m} \mid \forall \mathsf{x} \in \mathsf{Var} \,.\, \mathsf{m}(\mathsf{x}) \in \dot{\mathsf{m}}(\mathsf{x})\}$$

Then we can extend the Intervals abstractions to memories. Applying the Pointwise Construction introduced in Section 2.3, we have that $\langle \mathsf{Mem}^\imath, \dot{\sqsubseteq}^\imath, \dot{\sqcup}^\imath, \dot{\sqcap}^\imath, \mathsf{m}^\imath_\circ, \mathsf{m}^\imath_\bullet\rangle$ is a complete lattice, where $\mathsf{Mem}^\imath \triangleq \mathsf{Var} \to \text{Intv}$, $\mathsf{m}^\imath_\circ \triangleq \lambda \mathsf{x} \,.\, \bot^\imath$ and $\mathsf{m}^\imath_\bullet \triangleq \lambda \mathsf{x} \,.\, \top^\imath$. Applying the Pointwise Abstraction introduced in Section 2.3, we have the following Galois connection:

$$\langle \dot{\mathsf{Mem}}, \dot{\subseteq}\rangle \underset{\dot{\alpha}_\imath}{\overset{\dot{\gamma}_\imath}{\leftrightarrows}} \langle \mathsf{Mem}^\imath, \dot{\sqsubseteq}^\imath\rangle \quad \dot{\alpha}_\imath(\dot{\mathsf{m}}) \triangleq \lambda \mathsf{x} \,.\, \alpha_\imath \circ \dot{\mathsf{m}}(\mathsf{x}) \quad \dot{\gamma}_\imath(\mathsf{m}^\imath) \triangleq \lambda \mathsf{x} \,.\, \gamma_\imath \circ \mathsf{m}^\imath(\mathsf{x})$$

Finally, by composition, we have the Galois connection $(\wp(\mathsf{Mem}), \alpha_i^{\mathsf{m}}, \gamma_i^{\mathsf{m}}, \mathsf{Mem}^\imath)$, where $\alpha_i^{\mathsf{m}} \triangleq \dot{\alpha}_\imath \circ \alpha_{nr}$ and $\gamma_i^{\mathsf{m}} \triangleq \gamma_{nr} \circ \dot{\gamma}_\imath$. Hence $\langle \mathsf{Mem}^\imath, \dot{\sqsubseteq}^\imath, \dot{\sqcup}^\imath, \dot{\sqcap}^\imath, \mathsf{m}^\imath_\circ, \mathsf{m}^\imath_\bullet \rangle$ is the abstract domain of computation, where we define the abstract post-conditions semantics $[\![\mathsf{P}]\!]_i^{\curvearrowright} \in \mathsf{Mem}^\imath \to \mathsf{Mem}^\imath$. As before, we can derive this function by calculus, approximating the best correct approximation of $[\![\mathsf{P}]\!]^{\curvearrowright}$, namely the function $\alpha_i^{\mathsf{m}} \circ [\![\mathsf{P}]\!]^{\curvearrowright} \circ \gamma_i^{\mathsf{m}}$.

5.2.2.1 The Abstract Semantics for Intervals

The abstract semantics for Intervals needs two auxiliary functions. First, an abstract non-relational semantics for arithmetic expressions $[\![\mathsf{a}]\!]_i^{\curvearrowright} \in \mathsf{Mem}^\imath \to \mathrm{Intv}$ (Figure 5.2), used for approximating the concrete semantics of assignments. Second, an abstract semantics for boolean expressions $[\![\mathsf{b}]\!]_i^{\curvearrowright} \in \mathsf{Mem}^\imath \to \mathsf{Mem}^\imath$, used for approximating the set $\{\mathsf{m} \in X \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\}$, for a given $X \subseteq \mathsf{Mem}$. Defining a precise semantics for boolean expressions could be quite complex, so we settle for a very simple definition. Furthermore, we assume that all negations $\neg$ have been removed using DeMorgan's laws and usual arithmetic laws: $\neg(\mathsf{b}_1 \vee \mathsf{b}_2) \equiv (\neg\mathsf{b}_1) \wedge (\neg\mathsf{b}_2)$, $\neg(\mathsf{a}_1 \leq \mathsf{a}_2) \equiv (\mathsf{a}_2 \leq \mathsf{a}_1)$, etc. This allows us to give the semantics only for the base cases $\mathsf{a}_1 \bowtie \mathsf{a}_2$, where $\bowtie \in \{=, \neq, <, \leq\}$. We can note that the set $\{\mathsf{m} \in X \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\}$ is always a subset of the set X. This means that the identity function is always sound, namely $[\![\mathsf{b}]\!]_i^{\curvearrowright}\mathsf{m}^\imath \triangleq \mathsf{m}^\imath$ is a safe, but coarse, choice. We resort to the identity only in some cases, in order to decrease imprecision. In particular, we define:

$$
\begin{array}{l}
\text{let } \mathsf{m}^\imath(\mathsf{x}) = [a, b], \mathsf{m}^\imath(\mathsf{y}) = [c, d] \text{ in} \\
\quad [\![\mathsf{x} \leq \mathsf{n}]\!]_i^{\curvearrowright} \triangleq \left(\!\!\left| \; a \leq n \; ? \; \mathsf{m}^\imath[\mathsf{x} \hookleftarrow [a, \min\{b, n\}]] \; : \; \mathsf{m}^\imath_\circ \; \right|\!\!\right) \\
\quad [\![\mathsf{x} \leq \mathsf{y}]\!]_i^{\curvearrowright} \triangleq \left(\!\!\left| \; a \leq d \; ? \; \mathsf{m}^\imath[\mathsf{x} \hookleftarrow [a, \min\{b, d\}] \; \mathsf{y} \hookleftarrow [\max\{a, c\}, d]] \; : \; \mathsf{m}^\imath_\circ \; \right|\!\!\right)
\end{array}
$$

For all other cases of the form $\mathsf{a}_1 \bowtie \mathsf{a}_2$ we approximate with the identity function. Finally, for the following cases we can define easily a better abstraction than identity:

$$
\begin{array}{lll}
[\![\mathsf{tt}]\!]_i^{\curvearrowright}\mathsf{m}^\imath \triangleq \mathsf{m}^\imath & [\![\mathsf{ff}]\!]_i^{\curvearrowright}\mathsf{m}^\imath \triangleq \mathsf{m}^\imath_\circ & [\![(\mathsf{b})]\!]_i^{\curvearrowright}\mathsf{m}^\imath \triangleq [\![\mathsf{b}]\!]_i^{\curvearrowright}\mathsf{m}^\imath \\
[\![\mathsf{b}_1 \wedge \mathsf{b}_2]\!]_i^{\curvearrowright}\mathsf{m}^\imath \triangleq [\![\mathsf{b}_1]\!]_i^{\curvearrowright}\mathsf{m}^\imath \mathbin{\dot{\sqcap}^\imath} [\![\mathsf{b}_2]\!]_i^{\curvearrowright}\mathsf{m}^\imath & & [\![\mathsf{b}_1 \vee \mathsf{b}_2]\!]_i^{\curvearrowright}\mathsf{m}^\imath \triangleq [\![\mathsf{b}_1]\!]_i^{\curvearrowright}\mathsf{m}^\imath \mathbin{\dot{\sqcup}^\imath} [\![\mathsf{b}_2]\!]_i^{\curvearrowright}\mathsf{m}^\imath
\end{array}
$$

Remark. To define a precise abstract non-relational semantics for boolean expressions, a common solution is to use a backward abstract semantics for arithmetic expressions ${}^{\mathcal{B}}[\![\mathsf{a}]\!]_i^{\curvearrowright} \in \mathsf{Mem}^\imath \to (\mathrm{Intv} \to \mathsf{Mem}^\imath)$. The meaning of ${}^{\mathcal{B}}[\![\mathsf{a}]\!]_i^{\curvearrowright}(\mathsf{m}^\imath)([a, b]) = \mathsf{n}^\imath$ is that $\mathsf{n}^\imath$ is the most abstract memory less than, or equal to, $\mathsf{m}^\imath$ such that $[\![\mathsf{a}]\!]_i^{\curvearrowright}\mathsf{n}^\imath = [a, b]$.

Now we are ready to define the abstract post-conditions semantics $[\![\mathsf{P}]\!]_i^{\curvearrowright}$ for intervals. It is derived by calculus from $[\![\mathsf{P}]\!]^{\curvearrowright}$, meaning that the soundness proof (in Appendix A) is constructive: the derivation of the proof gives us the definition of the abstract semantics. The semantics can be computed directly on program syntax by the semantics operator $[\![\mathsf{P}]\!]_i^{\curvearrowright} \in \mathsf{Mem}^\imath \to \mathsf{Mem}^\imath$, defined for every $\mathsf{P} \in \mathsf{Imp}$ as follow:

$$
\begin{array}{l}
[\![\mathsf{P}]\!]_i^{\curvearrowright}\mathsf{m}^\imath_\circ \triangleq \mathsf{m}^\imath_\circ \qquad [\![{}^{\mathsf{i}}\mathsf{skip}^{\mathsf{f}}]\!]_i^{\curvearrowright}\mathsf{m}^\imath \triangleq \mathsf{m}^\imath \qquad [\![{}^{\mathsf{i}}\mathsf{x} := \mathsf{a}^{\mathsf{f}}]\!]_i^{\curvearrowright}\mathsf{m}^\imath \triangleq \mathsf{m}^\imath[\mathsf{x} \hookleftarrow [\![\mathsf{a}]\!]_i^{\curvearrowright}\mathsf{m}^\imath] \\
[\![{}^{\mathsf{i}}\mathsf{c}_1{}^{\mathsf{l}} \,.\, {}^{\mathsf{k}}\mathsf{c}_2{}^{\mathsf{f}}]\!]_i^{\curvearrowright}\mathsf{m}^\imath \triangleq [\![{}^{\mathsf{k}}\mathsf{c}_2{}^{\mathsf{f}}]\!]_i^{\curvearrowright} \circ [\![{}^{\mathsf{i}}\mathsf{c}_1{}^{\mathsf{l}}]\!]_i^{\curvearrowright}\mathsf{m}^\imath \\
[\![{}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\ \mathsf{P}_1\ \}\ \mathsf{else}\ \{\ \mathsf{P}_2\ \}^{\mathsf{f}}]\!]_i^{\curvearrowright}\mathsf{m}^\imath \triangleq [\![\mathsf{P}_1]\!]_i^{\curvearrowright} \circ [\![\mathsf{b}]\!]_i^{\curvearrowright}\mathsf{m}^\imath \mathbin{\dot{\sqcup}^\imath} [\![\mathsf{P}_2]\!]_i^{\curvearrowright} \circ [\![\neg\mathsf{b}]\!]_i^{\curvearrowright}\mathsf{m}^\imath \\
[\![{}^{\mathsf{i}}\mathsf{while}\ {}^{\mathsf{l}}\mathsf{b}\ \{\ \mathsf{P}\ \}^{\mathsf{f}}]\!]^{\curvearrowright}\mathsf{m}^\imath \triangleq [\![\neg\mathsf{b}]\!]_i^{\curvearrowright}\left(\mathrm{lfp}^{\dot{\sqsubseteq}^\imath}_{\mathsf{m}^\imath_\circ} \mathsf{F}_i^{\curvearrowright}\right) \text{ where: } \mathsf{F}_i^{\curvearrowright}(\mathsf{n}^\imath) \triangleq \mathsf{m}^\imath \mathbin{\dot{\sqcup}^\imath} [\![\mathsf{P}]\!]_i^{\curvearrowright}[\![\mathsf{b}]\!]_i^{\curvearrowright}\mathsf{n}^\imath
\end{array}
$$

Arithmetic expressions: $[\![\mathsf{a}]\!]^{\curvearrowright}_{\imath} \in \mathsf{Mem}^{\imath} \to \mathrm{Intv}$

$$[\![\mathsf{n}]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \triangleq [n,n] \qquad [\![\mathsf{x}]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \triangleq \mathsf{m}^{\imath}(\mathsf{x})$$
$$[\![\mathsf{a}_1 + \mathsf{a}_2]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \triangleq [\![\mathsf{a}_1]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} +^{\imath} [\![\mathsf{a}_2]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \qquad [\![\mathsf{a}_1 - \mathsf{a}_2]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \triangleq [\![\mathsf{a}_1]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} -^{\imath} [\![\mathsf{a}_2]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath}$$
$$[\![\mathsf{a}_1 * \mathsf{a}_2]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \triangleq [\![\mathsf{a}_1]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} *^{\imath} [\![\mathsf{a}_2]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \qquad [\![(\mathsf{a})]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \triangleq [\![\mathsf{a}]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath}$$

where abstract arithmetic operators $\oplus^{\imath} \in \mathrm{Intv} \times \mathrm{Intv} \to \mathrm{Intv}$, with $\oplus \in \{+,-,*\}$, are:

$$[a,b] +^{\imath} [c,d] \triangleq [a+c, b+d] \qquad [a,b] -^{\imath} [c,d] \triangleq [a-d, b-c]$$
$$[a,b] *^{\imath} [c,d] \triangleq [\min\{ac,ad,bc,bd\}, \max\{ac,ad,bc,bd\}]$$

Figure 5.2: Abstract Intervals semantics for arithmetic expressions.

The operator $\mathsf{F}^{\curvearrowright}_{\imath}$ is monotone on the CPO $\langle \mathsf{Mem}^{\imath}, \dot{\sqsubseteq}^{\imath}, \dot{\sqcup}^{\imath}, \mathsf{m}^{\imath}_{\circ} \rangle$, so its least fixpoint $\dot{\bigsqcup}^{\imath}_{n<\omega} \mathsf{F}^{\curvearrowright n}_{\imath}(\mathsf{m}^{\imath}_{\circ})$ exists. Furthermore, we have that the abstract semantics is correct.

Theorem 16. *The abstract post-conditions semantics for Intervals is a sound approximation of the post-conditions semantics, namely:*

$$[\![\mathsf{P}]\!]^{\curvearrowright}\gamma^{\mathsf{m}}_{\imath}(\mathsf{m}^{\imath}) \subseteq \gamma^{\mathsf{m}}_{\imath}[\![\mathsf{P}]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \quad \textit{or, equivalently,} \quad \alpha^{\mathsf{m}}_{\imath}[\![\mathsf{P}]\!]^{\curvearrowright}X \mathrel{\dot{\sqsubseteq}^{\imath}} [\![\mathsf{P}]\!]^{\curvearrowright}_{\imath}\alpha^{\mathsf{m}}_{\imath}(X)$$

for every $\mathsf{P} \in \mathsf{Imp}$, $\mathsf{m}^{\imath} \in \mathsf{Mem}^{\imath}$ *and* $X \in \wp(\Sigma)$.

Proof (*Sketch*). We need to prove that $[\![\mathsf{P}]\!]^{\curvearrowright}_{\imath}$ approximates the best correct approximation of $[\![\mathsf{P}]\!]^{\curvearrowright}$, namely that for every $\mathsf{m}^{\imath} \in \mathsf{Mem}^{\imath}$ we have $\alpha^{\mathsf{m}}_{\imath}[\![\mathsf{P}]\!]^{\curvearrowright}\gamma^{\mathsf{m}}_{\imath}(\mathsf{m}^{\imath}) \mathrel{\dot{\sqsubseteq}^{\imath}} [\![\mathsf{P}]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath}$. The proof is by structural induction on P and it relies on the soundness of the abstract semantics for arithmetic and boolean expressions. The first requires $\{n \mid \exists \mathsf{m} \in \gamma^{\mathsf{m}}_{\imath}(\mathsf{m}^{\imath}) \,.\, \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \subseteq \gamma_{\imath}[\![\mathsf{a}]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath}$, while the second requires $\{\mathsf{m} \mid \mathsf{m} \in \gamma^{\mathsf{m}}_{\imath}(\mathsf{m}^{\imath}) \wedge \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \subseteq \gamma_{\imath}[\![\mathsf{b}]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath}$. As an example, we show the derivation for the base case of the assignment statement.

$$\begin{aligned}
&\alpha^{\mathsf{m}}_{\imath}[\![{}^{\mathtt{i}}\mathsf{x} := \mathsf{a} {}^{\mathtt{f}}]\!]^{\curvearrowright}\gamma^{\mathsf{m}}_{\imath}(\mathsf{m}^{\imath}) \\
&= \quad \| \text{ definition of } [\![\cdot]\!]^{\curvearrowright} \\
&\alpha^{\mathsf{m}}_{\imath}(\{\mathsf{m}[\mathsf{x} \hookleftarrow n] \mid \mathsf{m} \in \gamma^{\mathsf{m}}_{\imath}(\mathsf{m}^{\imath}) \wedge \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\}) \\
&= \quad \|_{\alpha^{\mathsf{m}}_{\imath} = \dot{\alpha}_{\imath} \circ \alpha_{nr}} \text{ and definition of } \alpha_{nr} \\
&\dot{\alpha}_{\imath} \circ (\lambda \mathsf{y} \,.\, \{\mathsf{m}(\mathsf{y}) \mid \mathsf{m} \in \{\mathsf{m}[\mathsf{x} \hookleftarrow n] \mid \mathsf{m} \in \gamma^{\mathsf{m}}_{\imath}(\mathsf{m}^{\imath}) \wedge \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\}\}) \\
&= \\
&\dot{\alpha}_{\imath} \circ (\lambda \mathsf{y} \,.\, (\!|\, \mathsf{y} = \mathsf{x} \mathbin{?} \{n \mid \exists \mathsf{m} \in \gamma^{\mathsf{m}}_{\imath}(\mathsf{m}^{\imath}) \,.\, \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mathbin{:} \{\mathsf{m}(\mathsf{y}) \mid \mathsf{m} \in \gamma^{\mathsf{m}}_{\imath}(\mathsf{m}^{\imath})\} \,|\!)) \\
&\dot{\sqsubseteq}^{\imath} \quad \| \text{ soundness of } [\![\mathsf{a}]\!]^{\curvearrowright}_{\imath} \\
&\dot{\alpha}_{\imath} \circ (\lambda \mathsf{y} \,.\, (\!|\, \mathsf{y} = \mathsf{x} \mathbin{?} \gamma_{\imath}[\![\mathsf{a}]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \mathbin{:} \{\mathsf{m}(\mathsf{y}) \mid \mathsf{m} \in \gamma^{\mathsf{m}}_{\imath}(\mathsf{m}^{\imath})\} \,|\!)) \\
&= \quad \| \text{ definition of } \dot{\alpha}_{\imath} \\
&\lambda \mathsf{y} \,.\, (\!|\, \mathsf{y} = \mathsf{x} \mathbin{?} \alpha_{\imath}\gamma_{\imath}[\![\mathsf{a}]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \mathbin{:} \alpha_{\imath}(\{\mathsf{m}(\mathsf{y}) \mid \mathsf{m} \in \gamma^{\mathsf{m}}_{\imath}(\mathsf{m}^{\imath})\}) \,|\!) \\
&= \quad \| \text{ definition of } \gamma^{\mathsf{m}}_{\imath} \\
&\lambda \mathsf{y} \,.\, (\!|\, \mathsf{y} = \mathsf{x} \mathbin{?} \alpha_{\imath}\gamma_{\imath}[\![\mathsf{a}]\!]^{\curvearrowright}_{\imath}\mathsf{m}^{\imath} \mathbin{:} \alpha_{\imath}\gamma_{\imath}(\mathsf{m}^{\imath}(\mathsf{y})) \,|\!)
\end{aligned}$$

$$
\begin{array}{ll}
\dot{\sqsubseteq}^{\imath} & \| \text{ reductivity of } \alpha_\imath\gamma_\imath \\
\lambda y \,.\, (\!|\, y = x \,?\, [\![a]\!]^{\curvearrowright}_{\imath} m^{\imath} \,:\, m^{\imath}(y) \,|\!) & \\
= & \\
m^{\imath}[x \leftarrow [\![a]\!]^{\curvearrowright}_{\imath} m^{\imath}] & \\
= & \| \text{ definition of } [\![\cdot]\!]^{\curvearrowright}_{\imath} \\
[\![{}^{i}x := a^{f}]\!]^{\curvearrowright}_{\imath} m^{\imath} &
\end{array}
$$

□

It is easy to note that for every program $P = {}^{i}c^{f}$ we have that $\alpha^{m}_{\imath}(\{m \mid \langle f, m\rangle \in \tau^{\mathcal{R}}\}) \dot{\sqsubseteq}^{\imath} [\![P]\!]^{\curvearrowright}_{\imath}\alpha^{m}_{\imath}(\{m \mid \langle i, m\rangle \in \tau^{\mathcal{R}}\})$. Basically this semantics, given an abstract pre-condition (on memories) outputs a correct approximation of the abstract strongest post-condition (on memories).

Similarly to the concrete case, the abstract post-conditions semantics can be used to compute $\tau^{\imath}_{\ell} \in \mathsf{Lab} \to \mathsf{Mem}^{\imath}$, which is a sound approximation of $\tau^{\mathcal{R}}_{\ell}$. Indeed, if a program $P = {}^{i}c^{f}$ is not a sub-program (of another program), then $\tau^{\imath}_{\ell}(i) = \alpha^{m}_{\imath}(\mathsf{Mem}) = m^{\imath}_{\bullet}$, by definition. If a program $P = {}^{i}c^{f}$ is contained in a bigger program, we have the following cases. If the containing program is ${}^{l}c_1\,{}^{k}.P$, then $\tau^{\imath}_{\ell}(i) = \tau^{\imath}_{\ell}(k)$. If the containing program is ${}^{l}\mathsf{if}\ b\ \mathsf{then}\ \{P\}\ \mathsf{else}\ \{P_2\}^{k}$, then $\tau^{\imath}_{\ell}(i) = [\![b]\!]^{\curvearrowright}_{\imath}\tau^{\imath}_{\ell}(l)$. If the containing program is ${}^{l}\mathsf{if}\ b\ \mathsf{then}\ \{P_1\}\ \mathsf{else}\ \{P\}^{k}$, then $\tau^{\imath}_{\ell}(i) = [\![\neg b]\!]^{\curvearrowright}_{\imath}\tau^{\imath}_{\ell}(l)$. Finally, if the containing program is ${}^{l}\mathsf{while}\ {}^{t}b\ \{P\}^{k}$, then $\tau^{\imath}_{\ell}(i) = [\![b]\!]^{\curvearrowright}_{\imath}\tau^{\imath}_{\ell}(t)$ where $\tau^{\imath}_{\ell}(t) = \mathsf{lfp}^{\dot{\sqsubseteq}^{\imath}}_{m^{\imath}_{\circ}} F^{\curvearrowright}_{\imath}$ and $F^{\curvearrowright}_{\imath}(n^{\imath}) \triangleq \tau^{\imath}_{\ell}(l) \dot{\sqcup}^{\imath} [\![P]\!]^{\curvearrowright}_{\imath}[\![b]\!]^{\curvearrowright}_{\imath}n^{\imath}$. In any case, $\tau^{\imath}_{\ell}(f) = [\![P]\!]^{\curvearrowright}_{\imath}\tau^{\imath}_{\ell}(i)$.

Theorem 17. *For each control point* $k \in \mathsf{Lab}$, *the abstract state semantics* $\tau^{\imath}_{\ell}$ *is sound w.r.t. the state semantics* $\tau^{\mathcal{R}}_{\ell}$, *namely:*

$$\tau^{\mathcal{R}}_{\ell}(k) \subseteq \gamma^{m}_{\imath} \circ \tau^{\imath}_{\ell}(k) \quad \textit{or, equivalently} \quad \alpha^{m}_{\imath} \circ \tau^{\mathcal{R}}_{\ell}(k) \dot{\sqsubseteq}^{\imath} \tau^{\imath}_{\ell}(k)$$

Proof. The proof follows trivially from the soundness of the abstract post-conditions semantics (Theorem 16) and from the definition of $\tau^{\mathcal{R}}_{\ell}$ and $\tau^{\imath}_{\ell}$. □

5.2.2.2 The Verification

With a sound abstract semantics we can approximate the verification check $\tau^{\mathcal{R}} \subseteq \mathtt{inv}^{[n,m]}_{x}$ in the abstract domain of Intervals. By isomorphism, we have that the latter check is equivalent to the following: $\alpha_{\ell}(\tau^{\mathcal{R}}) = \tau^{\mathcal{R}}_{\ell} \dot{\subseteq} \alpha_{\ell}(\mathtt{inv}^{[n,m]}_{x})$. This means that the property is satisfied if and only if for every $k \in \mathsf{Lab}$ we have that $\tau^{\mathcal{R}}_{\ell}(k) \subseteq \{m \in \mathsf{Mem} \mid n \leq m(x) \leq m\}$. An under-approximation of $\{m \in \mathsf{Mem} \mid n \leq m(x) \leq m\}$ in $\mathsf{Mem}^{\imath}$ is the abstract memory $p^{\imath} \triangleq \lambda y \,.\, (\!|\, y = x \,?\, [n, m] \,:\, m^{\imath}_{\bullet} \,|\!)$ (indeed, $\alpha^{m}_{\imath}(\{m \in \mathsf{Mem} \mid n \leq m(x) \leq m\})$ is exactly $p^{\imath}$). Hence we can verify the property in the abstract, as stated by the following proposition.

Proposition 2. *If for every* $k \in \mathsf{Lab}$ *we have that* $\tau^{\imath}_{\ell}(k) \dot{\sqsubseteq}^{\imath} p^{\imath}$, *then* $\tau^{\mathcal{R}} \subseteq \mathtt{inv}^{[n,m]}_{x}$ *holds.*

Proof. We have that $\tau^{\mathcal{R}} \subseteq \mathtt{inv}_{\mathsf{x}}^{[n,m]}$ if and only if for every $\mathtt{k} \in \mathsf{Lab}$ it holds $\tau^{\mathcal{R}}_{\ell}(\mathtt{k}) \subseteq \{\mathsf{m} \in \mathsf{Mem} \mid n \leq \mathsf{m}(\mathsf{x}) \leq m\}$. Then the proof is given by the following implications.

$$
\begin{array}{l}
\tau^{\imath}_{\ell}(\mathtt{k}) \mathrel{\dot{\sqsubseteq}^{\imath}} \mathsf{p}^{\imath} \\
\Downarrow \quad \| \gamma^{\mathsf{m}}_{\imath}(\mathsf{p}^{\imath}) \subseteq \{\mathsf{m} \in \mathsf{Mem} \mid n \leq \mathsf{m}(\mathsf{x}) \leq m\} \text{ and monotonicity of } \gamma^{\mathsf{m}}_{\imath} \\
\gamma^{\mathsf{m}}_{\imath}(\tau^{\imath}_{\ell}(\mathtt{k})) \subseteq \gamma^{\mathsf{m}}_{\imath}(\mathsf{p}^{\imath}) \subseteq \{\mathsf{m} \in \mathsf{Mem} \mid n \leq \mathsf{m}(\mathsf{x}) \leq m\} \\
\Downarrow \quad \| \text{ soundness of } \tau^{\imath}_{\ell} \text{ (Theorem 17)} \\
\tau^{\mathcal{R}}_{\ell}(\mathtt{k}) \subseteq \gamma^{\mathsf{m}}_{\imath}(\tau^{\imath}_{\ell}(\mathtt{k})) \subseteq \gamma^{\mathsf{m}}_{\imath}(\mathsf{p}^{\imath}) \subseteq \{\mathsf{m} \in \mathsf{Mem} \mid n \leq \mathsf{m}(\mathsf{x}) \leq m\}
\end{array}
$$

□

As expected we lose completeness, indeed if $\forall\, \mathtt{k} \in \mathsf{Lab}\,.\, \tau^{\imath}_{\ell}(\mathtt{k}) \mathrel{\dot{\sqsubseteq}^{\imath}} \mathsf{p}^{\imath}$ holds then we have that the program satisfies $\mathtt{inv}_{\mathsf{x}}^{[n,m]}$, but if it does not hold we cannot say anything about the verification.

Remark. Note that the proposed abstract semantics does not guarantee termination, since Intervals is not ACC. To enforce termination we need to insert a widening operator into the semantics of conditional commands.

6 Hyper Program Analysis

The previous chapter was meant to introduce program verification by abstract interpretation, using trace properties as an example. Now we deal with the general case, where specifications are modeled as hyperproperties. Again, our systems are computer programs and, given a specification formalized as a hyperproperty, we want to check whether a program satisfies the specification or not.

6.1 Hyper Static Analysis

Similarly to what was done in the previous chapter, we need to fix some key concepts. As it happens for trace properties, we have to choose the *standard semantics*, which is the most precise representation of the behavior of a program. We use as standard semantics the transition system $\langle \Sigma^{\mathsf{P}}, \Upsilon^{\mathsf{P}}, \Omega^{\mathsf{P}}, \tau_{\mathsf{P}} \rangle$ associated to the program P (in turn generated by the SOS of Imp). Again, this is not a mandatory choice. Second, we have the *collecting semantics*, which is the most precise specification that the program satisfies. Since, in this chapter, we are interested in hyperproperties, the collecting semantics should be a set of sets of execution denotations, so it is different from the base semantic $\mathcal{S}^{\mathsf{P}}_{\text{base}}$ of the P (i.e. the interpretation of P in the execution denotations domain). Indeed, we cannot use the semantics in the hierarchy as they are, we need some extra work in order to define the collecting semantics.

Third, we need the *abstract collecting semantics*, which is an approximation of the collecting semantics, describing only the computable information we can model about a program behaviors. Again, the collecting semantics is, in general, not computable, hence we seek a (decidable) approximation, usually losing completeness.

6.1.1 The Hyperproperties Verification Issue

Setting the execution denotations domain $\mathbb{Den}$, then programs representations, i.e. the base semantics, and hyperproperties lie in different domains, the first are in $\wp(\mathbb{Den})$, whilst the latter are in $\wp(\wp(\mathbb{Den}))$. In fact, a hyperproperty is modeled as the set of all programs (representations) satisfying it. Suppose that $\mathcal{S}^{\mathsf{P}}_{\text{base}}$ is an element of the hierarchy of 4.1, corresponding to the execution denotations domain $\mathbb{Den}$. Then the collecting semantics is $\mathcal{S}^{\mathsf{P}}_{\text{coll}} \triangleq \{\mathcal{S}^{\mathsf{P}}_{\text{base}}\}$, which is indeed the strongest program hyperproperty of P. The set of all possible hyperproperties is $\mathtt{GEN}^{\mathtt{H}} \triangleq \wp(\wp(\mathbb{Den}))$. Then we have that P satisfies a hyperproperty $\mathfrak{Hp} \in \mathtt{GEN}^{\mathtt{H}}$, as usual written $\mathsf{P} \models \mathfrak{Hp}$, if and only if $\mathcal{S}^{\mathsf{P}}_{\text{coll}} \subseteq \mathfrak{P}$ or, equivalently, if and only if $\mathcal{S}^{\mathsf{P}}_{\text{base}} \in \mathfrak{Hp}$.

Remark. This is exactly the definition of system interpretations, system specifications and systems strongest specification, as introduced in Chapter 3. Indeed, hyperproperties are (mathematical) properties of programs, in the more general sense.

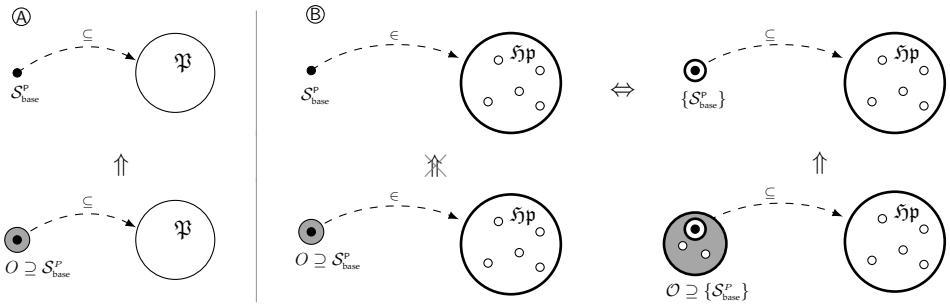


Figure 6.1: Over-approximation of trace properties Ⓐ and hyperproperties Ⓑ

An example of a (generic) hyperproperty for $\mathbb{Den} = \Sigma^{\infty}$ is *generalized non-interference* $\texttt{GNI} \triangleq \{X \subseteq \mathbb{Den} \mid \forall \bar{\sigma}, \bar{\sigma}' \in X \, \exists \bar{\sigma}^i \in X \, . \, (\bar{\sigma}^i_{\vdash} =_{\mathsf{H}} \bar{\sigma}_{\vdash} \wedge \bar{\sigma}^i \approx_{\mathsf{L}} \bar{\sigma}')\}$ [Clarkson and Schneider, 2010], stating that, for each pair $\bar{\sigma}, \bar{\sigma}'$ of executions there exists an interleaving one $\bar{\sigma}^i$ which agrees with $\bar{\sigma}$ on private variables (H) in input ($\vdash$) and with $\bar{\sigma}'$ on public variables (L)[1]. The program in Example 7 (Chapter 5) does not satisfy GNI, since $\{\tau^{\infty}\} \not\subseteq \texttt{GNI}$.

In the classic static analysis framework, since $\mathcal{S}^{\mathsf{P}}_{\text{coll}} = \mathcal{S}^{\mathsf{P}}_{\text{base}}$, we can compute a sound over-approximation $O \supseteq \mathcal{S}^{\mathsf{P}}_{\text{base}}$ of the base semantics allowing sound verification of trace properties (Figure 6.1, part Ⓐ). This is obtained by means of an abstraction of the concrete domain $\texttt{TRC}^{\mathsf{P}}$, where the abstract (base) semantics plays the role of the over-approximation. Let P be a program, $\texttt{TRC}^{\mathsf{P}}_{\sharp}$ an abstract domain of $\texttt{TRC}^{\mathsf{P}}$, $\mathfrak{P} \in \texttt{TRC}^{\mathsf{P}}$ a trace property and $\mathcal{S}^{\mathsf{P}}_{\sharp}$ an abstract interpretation of $\mathcal{S}^{\mathsf{P}}_{\text{base}}$ in $\texttt{TRC}^{\mathsf{P}}_{\sharp}$, i.e. $\mathcal{S}^{\mathsf{P}}_{\text{base}} \subseteq \gamma(\mathcal{S}^{\mathsf{P}}_{\sharp})$, then:

$$\langle \texttt{TRC}^{\mathsf{P}}, \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \texttt{TRC}^{\mathsf{P}}_{\sharp}, \subseteq^{\sharp} \rangle \;\text{ and }\; \gamma(\mathcal{S}^{\mathsf{P}}_{\sharp}) \subseteq \mathfrak{P} \;\text{ implies }\; \mathsf{P} \models \mathfrak{P}$$

Recall that, by under-approximation we can improve decidability of the confutation of a trace property, since if $U \subseteq \mathcal{S}^{\mathsf{P}}_{\text{base}}$ and $U \not\subseteq \mathfrak{P}$ then we have that $\mathcal{S}^{\mathsf{P}}_{\text{base}} \not\models \mathfrak{P}$. At this point, we can note, as expected, that trace hyperproperties can be verified in the classic analysis framework based on abstract interpretation:

$$\langle \texttt{TRC}^{\mathsf{P}}, \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \texttt{TRC}^{\mathsf{P}}_{\sharp}, \subseteq^{\sharp} \rangle \;\text{ and }\; \gamma(\mathcal{S}^{\mathsf{P}}_{\sharp}) \subseteq \bigcup \mathfrak{tHp} \;\text{ implies }\; \mathsf{P} \models \mathfrak{tHp}$$

Hence, we can still use classic methods based on over-approximation for verifying trace hyperproperties. Moreover, when dealing with confutation of specifications, also in this case we can use under-approximations in the usual way, since if we have $U \subseteq \mathcal{S}^{\mathsf{P}}_{\text{base}}$ and $U \not\subseteq \bigcup \mathfrak{tHp}$ then still we can derive that $\mathsf{P} \not\models \mathfrak{tHp}$.

Unfortunately, when we do not have restrictions on hyperproperties, the base semantics, in general, does not provide enough information for approximating verification, since $O \supseteq \mathcal{S}^{\mathsf{P}}_{\text{base}} \wedge O \in \mathfrak{Hp} \;\not\Rightarrow\; \mathcal{S}^{\mathsf{P}}_{\text{base}} \in \mathfrak{Hp}$ (Figure 6.1, part Ⓑ on the left). Over-approximations do not work properly because we are approximating on the wrong domain or, better, we are approximating the wrong object. Indeed, if we move towards $\texttt{GEN}^{\mathtt{H}}$, then $\mathcal{O} \supseteq \{\mathcal{S}^{\mathsf{P}}_{\text{base}}\} \wedge \mathcal{O} \subseteq \mathfrak{Hp} \;\Rightarrow\; \{\mathcal{S}^{\mathsf{P}}_{\text{base}}\} \subseteq \mathfrak{Hp}$, i.e. $\mathcal{S}^{\mathsf{P}}_{\text{base}} \in \mathfrak{Hp}$ (Figure 6.1, part Ⓑ on the right). The problem is due to the fact that the specification is defined on the domain $\texttt{GEN}^{\mathtt{H}}$, different from the domain

[1] Here $=_{\mathsf{H}}$ is an equivalence on states while $\approx_{\mathsf{L}}$ is on traces.

$\mathtt{TRC}^{\mathsf{P}}$, where the base semantics is computed. The solution is clearly to approximate the collecting semantics. In this way, we can exploit the abstract interpretation framework even for approximating hyperproperties verification. Hence, our goal is to define a program P semantics on the hyperlevel, similar to what has been done in [Assaf et al., 2017], i.e. we define the *hypersemantics* $\mathcal{S}^{\mathsf{P}}_{\text{coll}}$ such that $\mathcal{S}^{\mathsf{P}}_{\text{coll}} = \{\mathcal{S}^{\mathsf{P}}_{\text{base}}\}$.

An over-approximation of $\mathcal{S}^{\mathsf{P}}_{\text{coll}}$ clearly leads to a sound verification mechanism for hyperproperties. In fact, let P be a program, $\mathtt{GEN}^{\mathtt{H}}_{\sharp}$ be an abstract domain of $\mathtt{GEN}^{\mathtt{H}}$, $\mathfrak{Hp} \in \mathtt{GEN}^{\mathtt{H}}$ be a hyperproperty and $\mathcal{S}^{\mathsf{P}}_{\sharp}$ be an abstract interpretation of $\mathcal{S}^{\mathsf{P}}_{\text{coll}}$ in $\mathtt{GEN}^{\mathtt{H}}_{\sharp}$, i.e. $\mathcal{S}^{\mathsf{P}}_{\text{coll}} \subseteq \gamma^{H}(\mathcal{S}^{\mathsf{P}}_{\sharp})$, then:

$$\langle \mathtt{GEN}^{\mathtt{H}}, \subseteq \rangle \underset{\alpha^{H}}{\overset{\gamma^{H}}{\leftrightarrows}} \langle \mathtt{GEN}^{\mathtt{H}}_{\sharp}, \sqsubseteq^{\sharp} \rangle \text{ and } \gamma^{H}(\mathcal{S}^{\mathsf{P}}_{\sharp}) \subseteq \mathfrak{Hp} \quad \text{imply } \mathsf{P} \models \mathfrak{Hp}$$

Hence, we build a hypersemantics of the program, and then we can over-approximate it in some abstraction of the domain $\mathtt{GEN}^{\mathtt{H}}$. This is depicted in Figure 6.2, where in Ⓐ we have the classic case and in Ⓑ the hyper case.

Now we can use abstract interpretation in order to compute sound over-approximations of the collecting semantics. The problem here is that $\{\mathcal{S}^{\mathsf{P}}_{\text{base}}\}$ is *hard* to define in such a way that it could fit in the abstract interpretation framework, namely in a constructive way (as defined in Chapter 2). Classic verification methods for trace properties rely on the fact that base and collecting semantics coincide. Hence, the fact that we are able, by definition, to compute the base semantics implies that we are able to compute the collecting semantics as well. This does not hold for hyperproperties verification. In this case, there is a further layer of complexity, other than the one of making the computation feasible (i.e. the verification check decidable). So the first problem is to define the concrete semantics, which will be abstracted in the approximation phase. Indeed, we can follow different approaches.

The first one, which is the more intuitive, but also the more complicated to follow, is to try and apply the abstract interpretation "as it is". This means to find a way to compute $\mathcal{S}^{\mathsf{P}}_{\text{coll}}$, defining this latter on a suitable computational domain. Then we can abstract it as shown in Chapter 2, in order to make the verification check feasible.

Another approach is to define a hypersemantics $\mathcal{H}^{\mathsf{P}}$, computing at the level of sets of sets which *is not* the collecting semantics. In this approach we can follow two paths. The first is general, and it requires that the hypersemantics is correct, namely that it is an approximation of the collecting semantics. Formally, $\mathcal{S}^{\mathsf{P}}_{\text{coll}} \subseteq \mathcal{H}^{\mathsf{P}}$. Ideally, we should adapt the base semantics, lifting its semantic operator to sets of sets. The second is specifications-centric, and it requires that the hypersemantics is *equisatisfying* to the collecting semantics, w.r.t. a given set of specifications. Given a set of hyperproperties $\{\mathfrak{Hp}_i\}_{i \in \Delta} \subseteq \mathtt{GEN}^{\mathtt{H}}$, we say that $\mathcal{H}^{\mathsf{P}}$

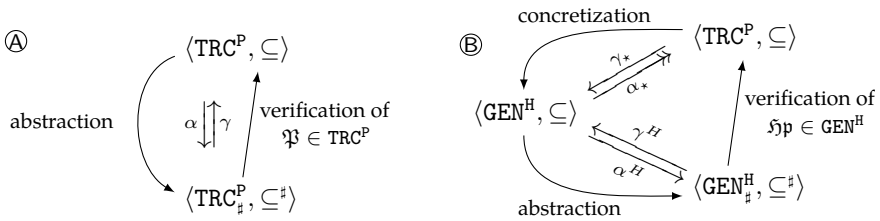


Figure 6.2: Verification (abstract interpretation) of properties Ⓐ and hyperproperties Ⓑ

is $\{\mathfrak{Hp}_i\}_{i\in\Delta}$ equisatisfying to $\mathcal{S}^{\mathsf{P}}_{\text{coll}}$ when: $\forall i \in \Delta \,.\, \mathcal{S}^{\mathsf{P}}_{\text{coll}} \subseteq \mathfrak{Hp}_i \Leftrightarrow \mathcal{H}^{\mathsf{P}} \subseteq \mathfrak{Hp}_i$. Then, again, we can abstract the hypersemantics as shown in Chapter 2, in order to make the verification check feasible.

Finally, as a last resort, we can use the base semantics, setting again $\mathcal{S}^{\mathsf{P}}_{\text{coll}} = \mathcal{S}^{\mathsf{P}}_{\text{base}}$ as for trace properties, and verify a *stronger* trace property. Given a hyperproperty $\mathfrak{Hp}$, a stronger trace property is $\mathfrak{P} \in \mathtt{TRC}^{\mathsf{P}}$ such $\wp(\mathfrak{P}) \subseteq \mathfrak{Hp}$, namely the trace hyperproperty corresponding to $\mathfrak{P}$ implies $\mathfrak{Hp}$. Clearly this method works only for the verification of subset-closed hyperproperties.

In this chapter we reason about these approaches, in order to distill pros and cons of each one. In Subsection 6.1.2 we inspect the case of a stronger trace property and in Subsection 6.1.3 the case of an equisatisfying hyperproperty. Finally, in Subsection 6.1.4, we deal with correct hypersemantics. Note that, $\mathcal{S}^{\mathsf{P}}_{\text{coll}}$ is a correct hypersemantics, so we present the case of computing exactly the collecting semantics as a particular case of correct hypersemantics.

6.1.2 Stronger Trace Property

As sketched just before, we wonder whether we can use classic verification methods for, at least, a subset of hyperproperties. Recall that a trace hyperproperty $\mathfrak{thp} \in \mathtt{TRC}^{\mathsf{H}}$ (see Chapter 4.2) is such that $\wp(\bigcup \mathfrak{thp}) = \mathfrak{thp}$. The verification of trace hyperproperties is simplified as follows:

$$\mathsf{P} \models \mathfrak{thp} \in \mathtt{TRC}^{\mathsf{H}} \;\Leftrightarrow\; \{\{\bar{\sigma}\} \mid \bar{\sigma} \in \mathcal{S}^{\mathsf{P}}_{\text{base}}\} \subseteq \mathfrak{thp} \;\Leftrightarrow\; \forall \bar{\sigma} \in \mathcal{S}^{\mathsf{P}}_{\text{base}} \,.\, \{\bar{\sigma}\} \in \mathfrak{thp}$$

This means that, exactly as it happens for trace properties, we can check this kind of hyperproperties on single executions: if we find at least one execution not satisfying the hyperproperty, then the program does not satisfy it. The hyperproperties we can verify, precisely, with the base semantics are all and only the trace hyperproperties, as stated by the following proposition.

Proposition 3. *For every hyperproperty* $\mathfrak{Hp}$*:*

$$\mathfrak{Hp} \in \mathtt{TRC}^{\mathsf{H}} \;\Leftrightarrow\; \exists \mathfrak{P} \in \mathtt{TRC}^{\mathsf{P}} \,\forall \mathsf{P} \,.\, (\mathsf{P} \models \mathfrak{P} \Leftrightarrow \mathsf{P} \models \mathfrak{Hp})$$

Proof. Trivial, since trace properties $\mathtt{TRC}^{\mathsf{P}}$ and trace hyperproperties $\mathtt{TRC}^{\mathsf{H}}$ are isomorphic (see Chapter 4.2). □

For instance, $\mathtt{Even}^{\mathsf{x}}_{\mathcal{H}} \triangleq \wp(\mathtt{Even}^{\mathsf{x}})$ is the trace hyperproperty equivalent to trace property $\mathtt{Even}^{\mathsf{x}}$ of Chapter 5. We can further generalize this restriction, allowing us to preserve the possibility of verifying hyperproperties on base semantics at least for confutation. It should be clear that, in the general case, we have to compute the whole semantics $\mathcal{S}^{\mathsf{P}}_{\text{base}}$ in order to confute the hyperproperty. However, it is worth noting that there is a particular kind of hyperproperties that generalizes hypersafety and whose verification test can be simplified. These are subset-closed hyperproperties (introduced in Section 4.2). Recall that a subset-closed hyperproperty $\mathfrak{chp} \in \mathtt{SSC}^{\mathsf{H}}$ is such that if $X \in \mathfrak{chp}$ then $\forall Y \subseteq X \,.\, Y \in \mathfrak{chp}$.

All trace hyperproperties are subset-closed but not vice-versa (one example is observational determinism [Zdancewic and Myers, 2003], which is subset-closed but not a trace hyperproperty). In particular, a subset-closed hyperproperty $\mathfrak{chp}$ is also a trace hyperproperty if, in addition, it holds: $\mathcal{X} \subseteq \mathfrak{chp} \Rightarrow \bigcup \mathcal{X} \in \mathfrak{chp}$. It turns out that lots of interesting

hyperproperties are subset-closed, e.g. all hypersafety and some hyperliveness. In this case, validation becomes:

$$\mathsf{P} \models \mathfrak{chp} \in \mathtt{SSC}^{\mathtt{H}} \;\Leftrightarrow\; \wp(\mathcal{S}^{\mathsf{P}}_{\text{base}}) \subseteq \mathfrak{chp} \;\Leftrightarrow\; \forall X \subseteq \mathcal{S}^{\mathsf{P}}_{\text{base}} \,.\, X \in \mathfrak{chp}$$

As observed in Section 4.2, $\wp(\mathcal{S}^{\mathsf{P}}_{\text{base}})$ is the strongest subset-closed hyperproperty of P, which is an approximation of the collecting semantics, through the abstraction $\rho_{\text{s}} = \lambda\mathcal{X} \,.\, \{Y \subseteq X \mid X \in \mathcal{X}\}$ of Chapter 4.2. It is clear that this does not change the validation of $\mathfrak{chp}$, but it may in general simplify the confutation, since we do not need the whole semantics $\mathcal{S}^{\mathsf{P}}_{\text{base}}$: it is sufficient to find a $X \subseteq \mathcal{S}^{\mathsf{P}}_{\text{base}}$ such that $X \notin \mathfrak{chp}$ in order to imply $\mathcal{S}^{\mathsf{P}}_{\text{coll}} \not\subseteq \mathfrak{chp}$. A subset-closed hyperproperty for $\mathbb{Den} = \Sigma^{\infty}$ which is not a trace hyperproperty is *termination insensitive non-interference* $\mathtt{TINI} \triangleq \{X \subseteq \mathbb{Den} \mid \forall \bar{\sigma}, \bar{\sigma}' \in X \,.\, \bar{\sigma}_{\vdash} =_{\mathsf{L}} \bar{\sigma}'_{\vdash} \Rightarrow (\bar{\sigma}_{\dashv} =_{\circlearrowleft} \vee \bar{\sigma}'_{\dashv} =_{\circlearrowleft} \vee \bar{\sigma}_{\dashv} =_{\mathsf{L}} \bar{\sigma}'_{\dashv})\}$ [Clarkson and Schneider, 2010], stating that, each pair of executions agreeing on public variables (L) in input ($\vdash$), must terminate agreeing on public variables in output ($\dashv$). The program in Example 7, with typing $\Gamma(\mathsf{x}) = \mathsf{L}, \Gamma(\mathsf{y}) = \mathsf{H}$, satisfies TINI since all terminating traces provide the same value for x, i.e. $\tau^{\infty} \in \mathtt{TINI}$. In [Clarkson and Schneider, 2010], the authors proved that TINI is 2-hypersafety, hence it is subset-closed, and, conversely, they proved that GNI is not subset-closed.

Finally, recall that subset-closed hyperproperties can be seen as unions of trace hyperproperties (Section 4.2), namely any $\mathfrak{chp} \in \mathtt{SSC}^{\mathtt{H}}$ can be characterized as $\mathfrak{chp} = \bigcup_{i \in \Delta} \mathfrak{thp}_i$. This implies that, in order to validate $\mathfrak{chp}$ on the base semantics it is sufficient to validate just one of these $\mathfrak{thp}_i$. In fact, if $\mathsf{P} \models \mathfrak{thp}_i$, i.e. $\mathcal{S}^{\mathsf{P}}_{\text{base}} \in \mathfrak{thp}_i$, then $\mathcal{S}^{\mathsf{P}}_{\text{base}} \in \mathfrak{chp}$ and hence $\mathsf{P} \models \mathfrak{chp}$.

6.1.3 Equisatisfying Hyperproperty

Since, as we will se in a moment, the collecting semantics is hard to define in a constructive way, we may use an equisatisfying hypersemantics, easier to define. This latter is specific to the hyperproperties we want to verify, hence we lose generality. Given a set of hyperproperties $\{\mathfrak{hp}_i\}_{i \in \Delta} \subseteq \mathtt{GEN}^{\mathtt{H}}$, we say that $\mathcal{H}^{\mathsf{P}}$ is $\{\mathfrak{hp}_i\}_{i \in \Delta}$ equisatisfying to $\mathcal{S}^{\mathsf{P}}_{\text{coll}}$ when: $\forall i \in \Delta \,.\, \mathcal{S}^{\mathsf{P}}_{\text{coll}} \subseteq \mathfrak{hp}_i \Leftrightarrow \mathcal{H}^{\mathsf{P}} \subseteq \mathfrak{hp}_i$. The intuition here is that defining a hypersemantics which works for every hyperproperty is hard. Instead, defining a hypersemantics driven by the particular hyperproperties we want to verify is easier, clearly losing the capability to verify other hyperproperties.

Remark. Note that an equisatisfying hypersemantics is a slightly different concept compared to a collecting semantics. Both are complete verification methods for a particular set of specifications but the second implies that the set of specifications is an abstraction of $\mathtt{GEN}^{\mathtt{H}}$ while the first does not.

As the limit case, we have that the set of specifications is a singleton, meaning that the hypersemantics can be used only for verifying one hyperproperty. In this latter case, we fix the hyperproperty we want to verify to a given $\mathfrak{hp} \in \mathtt{GEN}^{\mathtt{H}}$. Suppose to have a hypersemantics $\mathcal{H}^{\mathsf{P}}$ which is $\{\mathfrak{hp}\}$ equisatisfying, namely, such that:

$$\mathcal{H}^{\mathsf{P}} \subseteq \mathfrak{hp} \Leftrightarrow \mathcal{S}^{\mathsf{P}}_{\text{coll}} \subseteq \mathfrak{hp} \tag{6.1}$$

The hypersemantics must be related to the base semantics in some way, so suppose we have an abstraction function $\alpha_{\mathcal{H}} \in \mathtt{TRC}^{\mathsf{P}} \to \mathtt{GEN}^{\mathtt{H}}$, such that $\mathcal{H}^{\mathsf{P}} = \alpha_{\mathcal{H}}(\mathcal{S}^{\mathsf{P}}_{\text{base}})$. Suppose

now that the base semantics is computed by means of the computational fixpoint definition $\langle F, \mathbb{O}, \bot \rangle$, where $\mathbb{O} = \langle \mathtt{TRC}^{\mathsf{P}}, \sqsubseteq, \sqcup \rangle$, i.e. $\mathcal{S}^{\mathsf{P}}_{\text{base}} = \mathsf{lfp}^{\sqsubseteq}_{\bot} F$ and the hypersemantics by means of $\langle F_{\mathcal{H}}, \mathbb{O}_{\mathcal{H}}, \bot^{\mathcal{H}} \rangle$, where $\mathbb{O}_{\mathcal{H}} = \langle \mathtt{GEN}^{\mathtt{H}}, \sqsubseteq^{\mathcal{H}}, \sqcup^{\mathcal{H}} \rangle$, i.e. $\mathcal{H}^{\mathsf{P}} = \mathsf{lfp}^{\sqsubseteq^{\mathcal{H}}}_{\bot^{\mathcal{H}}} F_{\mathcal{H}}$. Suppose that $\alpha_{\mathcal{H}}$, F and $F_{\mathcal{H}}$ satisfy the condition of the Kleenian Fixpoint Transfer theorem 6. Then we have $\mathcal{H}^{\mathsf{P}} = \mathsf{lfp}^{\sqsubseteq^{\mathcal{H}}}_{\bot^{\mathcal{H}}} F_{\mathcal{H}} = \alpha_{\mathcal{H}}(\mathsf{lfp}^{\sqsubseteq}_{\bot} F) = \alpha_{\mathcal{H}}(\mathcal{S}^{\mathsf{P}}_{\text{base}})$. The abstraction $\alpha_{\mathcal{H}}$ is justified by the fact that the hypersematics definition is guided by $\alpha_{\mathcal{H}}$. All information for the verification process is contained in $\mathcal{S}^{\mathsf{P}}_{\text{base}}$, the problem is how this information is represented in the base semantics. The abstraction highlights the relations between traces needed for the verification of $\mathfrak{Hp}$. This could not be done approximating a semantics at the level of sets, but could be done approximating a semantics at the level of sets of sets. An example of this approach could be found in [Urban and Müller, 2018], where $\alpha_{\mathcal{H}}$ is defined by partitioning the execution denotations domain $\mathbb{D}\mathtt{en}$. Instead, in Chapter 7 we will define a hypersemantics equisatisfying w.r.t. particular kinds of subset-closed hyperproperties, simplifying the verification process.

Remark. Here, the abstraction function is on the computational domains, *not* on the approximation domain. Indeed, the only correctness criterion (on the approximation order) needed is the one of Equation 6.1.

Then, as usual, we can define an abstract hypersemantics $\mathcal{H}^{\mathsf{P}}_{\sharp}$, approximating $\mathcal{H}^{\mathsf{P}}$, which is effectively computable. This is done applying the abstract interpretation framework where the concrete semantics is $\mathcal{H}^{\mathsf{P}}$. Indeed, a sound over-approximation of this latter could be safely used for verifying $\mathfrak{Hp}$, namely:

$$\gamma(\mathcal{H}^{\mathsf{P}}_{\sharp}) \subseteq \mathfrak{Hp} \Rightarrow \mathcal{S}^{\mathsf{P}}_{\text{coll}} \subseteq \mathfrak{Hp}$$

Example 8. In [Urban and Müller, 2018], the authors define the hyperproperty called *input data usage* expressing the fact that the outcome of a program does not depend on part of its input data. For a program P we can identify two sets of variables O_{P} and I_{P} identifying input and output data, respectively. Given a trace $\bar{\sigma} \in \Sigma^{\vec{\infty}}$ we denote, as usual, with $\bar{\sigma}_{\vdash}$ its initial state and with $\bar{\sigma}_{\dashv}$ its final state. If $\bar{\sigma}$ is infinite then $\bar{\sigma}_{\dashv} = \circlearrowleft$. The input variables at the initial states store the values of a program input data and the output variables at the final states store the values of a program outcome. Following [Urban and Müller, 2018], we denote with $\sigma(i)$ the value of the input data stored in the input variable i in the state σ. Similarly, we denote with $\sigma(o)$ the value of the outcome stored in the output variable o in the state σ. The relation $\not\approx_i \subseteq \Sigma \times \Sigma$ denotes the fact that two states disagree on the value of the input variable i but agree on the values of all others variables. An input variable $i \in I_{\mathsf{P}}$ is *unused* w.r.t. a program with maximal trace semantics $\tau^{\vec{\infty}}$ when:

$$\textsc{unused}_i(\tau^{\vec{\infty}}) \triangleq \left(\begin{array}{l} \forall \bar{\sigma} \in \tau^{\vec{\infty}}, n \in \mathbb{Z} \,.\, \bar{\sigma}_{\vdash}(i) \neq n \Rightarrow \\ \exists \bar{\sigma}' \in \tau^{\vec{\infty}} \,.\, \bar{\sigma}'_{\vdash} \not\approx_i \sigma_{\vdash} \wedge \bar{\sigma}'_{\vdash}(i) = n \wedge \bar{\sigma}_{\dashv} = \bar{\sigma}'_{\dashv} \end{array} \right) \tag{6.2}$$

This basically means that the outcome of the program does not depend on the initial value of the input variable i. The input data usage hyperproperty $\mathcal{N}$ can be formally defined as:

$$\mathtt{IDU} \triangleq \{\tau^{\vec{\infty}} \subseteq \Sigma^{\vec{\infty}} \mid \forall i \in I_{\mathsf{P}} \,.\, \textsc{unused}_i(\tau^{\vec{\infty}})\}$$

IDU expresses the fact that the outcome of a program does not depend on any input data.

In practice, weaker forms of input data usage hyperproperties could be useful, namely restricting the check to subsets $J \subseteq I_{\mathsf{P}}$ of input variables, namely $\texttt{IDU}_J \triangleq \{\tau^{\vec{\infty}} \in \Sigma^{\vec{\infty}} \mid \forall i \in J \,.\, \textsc{unused}_i(\tau^{\vec{\infty}})\}$.

In order to verify this hyperproperty with abstract interpretation, in [Urban and Müller, 2018] the authors define a hypersemantics, called *outcome semantics*, serving as concrete semantics. They derive this semantics abstracting the maximal trace semantics by partitioning. Given a partition $Q \in \wp(\wp(\Sigma^{\vec{\infty}}))$ of programs traces, the abstraction $\alpha_{\mathcal{Q}} \in \wp(\Sigma^{\vec{\infty}}) \to \wp(\wp(\Sigma^{\vec{\infty}}))$ is defined as $\alpha_{\mathcal{Q}}(X) \triangleq \{X \cap Y \mid Y \in \mathcal{Q}\}$. In particular, they define the *outcome partition* $\mathcal{O}$, parametric in a set of output variables $o_1, o_2, \ldots o_m$, as:

$$\mathcal{O} \triangleq \left\{ \{\bar{\sigma} \in \Sigma^{\vec{+}} \mid \forall k \in [0,m] \,.\, \bar{\sigma}_{\dashv}(o_k) = v_k\} \;\middle|\; v_1, v_2, \ldots v_k \in \mathbb{Z} \right\} \cup \{\Sigma^{\vec{\omega}}\}$$

The partition contains all sets of finite traces that agree on the values of the output variables in their outcome, and all infinite traces. Then, the outcome abstraction is:

$$\alpha_{\bullet} \triangleq \lambda X \,.\, \left\{ \{\bar{\sigma} \in X^{\vec{+}} \mid \forall k \in [0,m] \,.\, \bar{\sigma}_{\dashv}(o_k) = v_k\} \;\middle|\; v_1, v_2, \ldots v_k \in \mathbb{Z} \right\} \cup \{X^{\vec{\omega}}\}$$

Finally, the outcome semantics is obtained as abstraction of the maximal trace semantics: $\tau^{\bullet} = \alpha_{\bullet}(\tau^{\vec{\infty}}) \in \wp(\wp(\Sigma^{\vec{\infty}}))$. The authors show also how to compute this semantics in a constructive way, with a suitable computational fixpoint definition (see [Urban and Müller, 2018] for the details). This latter semantics computes at the level of sets of sets and it is proven to be complete w.r.t. the input data usage hyperproperty: $\mathsf{P} \models \texttt{IDU}_J \Leftrightarrow \tau^{\bullet} \subseteq \texttt{IDU}_J$.

Hence, the outcome semantics is an example of an equisatisfying hypersemantics w.r.t. $\{\texttt{IDU}_J\}$: it could be used to verify (precisely) IDU but it cannot be used for other hyperproperties.

6.1.4 Correct Hypersemantics

A correct hypersemantics[2] $\mathcal{H}^{\mathsf{P}}$ is such that $\{\mathcal{S}^{\mathsf{P}}_{\text{base}}\} \subseteq \mathcal{H}^{\mathsf{P}} \subseteq \wp(\mathcal{S}^{\mathsf{P}}_{\text{base}})$, meaning that it can be used for hyperproperties verification, but it is not guaranteed to be complete. Indeed, if $\{\mathcal{S}^{\mathsf{P}}_{\text{base}}\} \subsetneq \mathcal{H}^{\mathsf{P}}$ we lose completeness in the verification of generic hyperproperties. Still, we have completeness for subset-closed hyperproperties, since $\mathcal{H}^{\mathsf{P}} \subseteq \wp(\mathcal{S}^{\mathsf{P}}_{\text{base}})$. In the following we will show how to compute a correct hypersemantics $\mathcal{H}^{\mathsf{P}} = \{\mathcal{S}^{\mathsf{P}}_{\text{base}}\}$ or $\mathcal{H}^{\mathsf{P}} = \wp(\mathcal{S}^{\mathsf{P}}_{\text{base}})$. We will see an example of a correct hypersemantics different from these two in Chapter 7.

Remark. When $\mathcal{H}^{\mathsf{P}} = \{\mathcal{S}^{\mathsf{P}}_{\text{base}}\}$, it means that we are computing exactly the collecting semantics of P in $\texttt{GEN}^{\mathtt{H}}$. Analogously, when $\mathcal{H}^{\mathsf{P}} = \wp(\mathcal{S}^{\mathsf{P}}_{\text{base}})$, it means that we are computing the collecting semantics of P in $\texttt{SSC}^{\mathtt{H}}$.

6.1.4.1 Subset-closed and Generic Hypersemantics

Given a program P, we denote with $\mathcal{H}^{\mathsf{P}}_{\mathsf{G}}$ its generic (correct) hypersemantics, i.e. $\mathcal{H}^{\mathsf{P}}_{\mathsf{G}} \triangleq \mathcal{S}^{\mathsf{P}}_{\text{coll}} = \{\mathcal{S}^{\mathsf{P}}_{\text{base}}\}$, which is its strongest generic hyperproperty. Analogously, we denote with $\mathcal{H}^{\mathsf{P}}_{\mathsf{S}}$ its subset-closed (correct) hypersemantics, i.e. $\mathcal{H}^{\mathsf{P}}_{\mathsf{S}} \triangleq \rho_{\mathsf{S}}(\mathcal{S}^{\mathsf{P}}_{\text{coll}}) = \wp(\mathcal{S}^{\mathsf{P}}_{\text{base}})$[3], which is its strongest subset-closed hyperproperty.

[2] An example of this kind of hypersemantics was first spelled out in [Assaf et al., 2017].

[3] Recall that $\rho_{\mathsf{S}} = \lambda\mathcal{X} \,.\, \{Y \subseteq X \mid X \in \mathcal{X}\}$ is the upper closure operator defined in Subsection 4.2.3.

It is worth nothing that, $\mathcal{H}^{\mathsf{P}}_{\mathtt{s}}$ and $\mathcal{H}^{\mathsf{P}}_{\mathtt{c}}$ do not give us more information on the behavior of P than $\mathcal{S}^{\mathsf{P}}_{\text{base}}$, being isomorphic to this latter. Namely these hypersemantics do not provide different observables of the program, but only new verification methods for hyperproperties. In particular, over-approximations of hypersemantics, defined on more expressive semantic levels, provide verification methods for generic and subset-closed hyperproperties. We cannot verify precisely, or sometimes verify at all, these hyperproperties with base semantics.

Using hypersemantics we are basically computing the base semantics at the hyperlevel: our aim is to emulate the base semantics computation at the level of sets of sets. In this case we have to *transfer* the fixpoint computation from the abstract domain of base semantics, to the concrete domain of hypersemantics. We can follow two ways: we can just *lift* the operator to sets, or we can use the *best complete concretization* of the base semantic operator. Basically, we want to find a monotone operator $F_{\mathcal{H}} \in \wp(\wp(\mathbb{D}\mathtt{en})) \to \wp(\wp(\mathbb{D}\mathtt{en}))$, such that $\mathcal{H}^{\mathsf{P}} = \mathsf{lfp}\, F_{\mathcal{H}}$, built on top of the operator F used to compute the base semantics. In order to achieve this goal we need some preliminary results.

Again on Fixpoint Transfer. When the semantics is constructive, we can derive an approximate semantics by abstraction of the concrete one. The Kleenian Fixpoint Approximation (Theorem 4) requires abstraction soundness, i.e. $\alpha \circ F \sqsubseteq^{\sharp} F^{\sharp} \circ \alpha$, guaranteeing fixpoint approximation, i.e. $\alpha(\mathsf{lfp}^{\sqsubseteq}_{\perp} F) \sqsubseteq^{\sharp} \mathsf{lfp}^{\sqsubseteq^{\sharp}}_{\perp^{\sharp}} F^{\sharp}$. The Kleenian Fixpoint Transfer (Theorem 6) requires completeness, i.e. $\alpha \circ F = F^{\sharp} \circ \alpha$, guaranteeing the fixpoint transfer from the concrete domain to the abstract domain, i.e. $\alpha(\mathsf{lfp}^{\sqsubseteq}_{\perp} F) = \mathsf{lfp}^{\sqsubseteq^{\sharp}}_{\perp^{\sharp}} F^{\sharp}$.

Suppose now we are interested in transferring the fixpoint computation from the abstract level to the concrete one. This is what we want in hyperproperties verification: we want to transfer a fixpoint semantics, we are able to compute, from the abstract domain $\mathtt{TRC}^{\mathsf{P}}$ to the concrete domain $\mathtt{GEN}^{\mathsf{H}}$. Unfortunately, the completeness requirement observed in the abstract (backward), i.e. $\alpha \circ F = F^{\sharp} \circ \alpha$, is not the same as checking completeness in the concrete (forward), i.e. $F \circ \gamma = \gamma \circ F^{\sharp}$. In order to transfer fixpoints from abstract to concrete we need precisely the latter direction. In this case, we provide the forward version of the Kleenian Fixpoint Transfer theorem.

Let $\langle F, \mathbb{O}, \perp \rangle$, with $\mathbb{O} = \langle \mathcal{O}, \sqsubseteq, \sqcup \rangle$, and $\langle F^{\sharp}, \mathbb{O}^{\sharp}, \perp^{\sharp} \rangle$, with $\mathbb{O}^{\sharp} = \langle \mathcal{O}^{\sharp}, \sqsubseteq^{\sharp}, \sqcup^{\sharp} \rangle$, be concrete and abstract computational fixpoint definitions.

Theorem 18 (Forward Kleenian Fixpoint Tranfser)**.** *Assume that the strict Scott-continuous function $\gamma \in \mathcal{O}^{\sharp} \to \mathcal{O}$ satisfies the* commutation condition *$F \circ \gamma = \gamma \circ F^{\sharp}$. Then we have $\mathsf{lfp}^{\sqsubseteq}_{\perp} F = \gamma(\mathsf{lfp}^{\sqsubseteq^{\sharp}}_{\perp^{\sharp}} F^{\sharp})$.*

Proof. See the extended version (Theorem 29) in Appendix A. □

As happens for the classic Kleenian Fixpoint Transfer theorem, the requirement that γ must be continuous is sometimes to strong. Indeed, it is sufficient that γ preserves the limit of increasing iterates, not necessarily every directed set/chain.

Example 9. Suppose to have, for an arbitrary set S, two computational fixpoint definitions $\langle F, \mathbb{O}, \{\varnothing\} \rangle$ and $\langle F^{\sharp}, \mathbb{O}^{\sharp}, \varnothing \rangle$, with $\mathbb{O} = \langle \wp(\wp(S)), \subseteq, \cup \rangle$ and $\mathbb{O}^{\sharp} = \langle \wp(S), \subseteq, \cup \rangle$. Then let $\gamma \triangleq \lambda X \,.\, \wp(X)$. Clearly, γ is not Scott-continuous, since $\wp(X \cup Y) \neq \wp(X) \cup \wp(Y)$, in general. Suppose that the iterates of $F^{\sharp}$ are: $F^{\sharp 0}(\varnothing) = \varnothing$, $F^{\sharp}(\varnothing) = X$, $F^{\sharp}(X) = X \cup Y$. In this case,

they form an increasing (limited) chain and hence $F^{\sharp 0}(\varnothing) \cup F^{\sharp 1}(\varnothing) \cup F^{\sharp 2}(\varnothing) = X \cup Y$. Now, $\gamma(F^{\sharp 0}(\varnothing)) = \{\varnothing\}, \gamma(F^{\sharp 1}(\varnothing)) = \wp(X), \gamma(F^{\sharp 2}(\varnothing)) = \wp(X \cup Y)$ form an increasing (limited) chain as well, since $X \subseteq X \cup Y$ trivially implies $\wp(X) \subseteq \wp(X \cup Y)$. Let $F^0(\{\varnothing\}) \triangleq \{\varnothing\}$, $F^1(\{\varnothing\}) \triangleq \wp(X)$, $F^2(\wp(X)) \triangleq \wp(X \cup Y)$. The commutation condition holds and hence we have:

$$\bigcup_{n<3} F^n(\{\varnothing\}) = \mathsf{lfp}^{\subseteq}_{\{\varnothing\}} F = \wp(X \cup Y) = \gamma(\mathsf{lfp}^{\subseteq}_{\varnothing} F^\sharp) = \gamma(\bigcup_{n<3} F^{\sharp n}(\varnothing))$$

In a Galois connection-based abstract interpretation, it is well known that the Kleenian Fixpoint Approximation theorem trivially hold when $F^\sharp$ is the best correct approximation of F, i.e. $F^\sharp = \alpha \circ F \circ \gamma$. Hence, we look for a similar characterization in the dual case. In particular, we look for a systematic way to retrieve a concrete operator which best represents a given abstract operator. Exploiting the "duality principle" of abstract interpretation [Cousot and Cousot, 1992] we can obtain the best correct concretization as $F \triangleq \gamma \circ F^\sharp \circ \alpha$. Then we still trivially have that $\gamma(\mathsf{lfp}^{\sqsupseteq^\sharp}_{\bot^\sharp} F^\sharp) \sqsupseteq \mathsf{lfp}^{\sqsupseteq}_{\bot} F$ and $\mathsf{lfp}^{\sqsupseteq^\sharp}_{\bot^\sharp} F^\sharp \sqsupseteq^\sharp \alpha(\mathsf{lfp}^{\sqsupseteq}_{\bot} F)$. Moreover, in a Galois insertion settings, it is always possible to derive a complete (backward and forward) concretization, called *best complete concretisation*, of a given abstract semantics.

Theorem 19 (Best Complete Concretization)**.** *Let $\langle \mathcal{O}, \sqsubseteq \rangle$ and $\langle \mathcal{O}^\sharp, \sqsubseteq^\sharp \rangle$ be two POSET such that $\langle \mathcal{O}, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{O}^\sharp, \sqsubseteq^\sharp \rangle$. Let $F^\sharp \in \mathcal{O}^\sharp \to \mathcal{O}^\sharp$, monotone, and $F^{\mathrm{bcc}} \triangleq \gamma \circ F^\sharp \circ \alpha$. Then $\mathcal{O}^\sharp$ is both backward and forward complete for F^{bcc}.*

Proof. Due to the fact that the two domains are linked by a Galois insertion, it holds that $\alpha \circ \gamma = \mathsf{id}$ on $\mathcal{O}^\sharp$. Hence we have:

- $\alpha \circ F^{\mathrm{bcc}} = \alpha \circ \gamma \circ F^\sharp \circ \alpha = F^\sharp \circ \alpha$
- $F^{\mathrm{bcc}} \circ \gamma = \gamma \circ F^\sharp \circ \alpha \circ \gamma = \gamma \circ F^\sharp$

□

Note that $F^\sharp$ is the bca of F^{bcc} in $\mathcal{O}^\sharp$: $F^{\mathrm{bcc}^{\mathrm{bca}}} = \alpha \circ F^{\mathrm{bcc}} \circ \gamma = \alpha \circ \gamma \circ F^\sharp \circ \alpha \circ \gamma = F^\sharp$. Hence, given an abstract function $F^\sharp$ it is possible to derive a concrete function F, for which $F^\sharp$ is an approximation, such that $\alpha(\mathsf{lfp}^{\sqsubseteq}_{\bot} F) = \mathsf{lfp}^{\sqsubseteq^\sharp}_{\bot^\sharp} F^\sharp$ and $\mathsf{lfp}^{\sqsubseteq}_{\bot} F = \gamma(\mathsf{lfp}^{\sqsubseteq^\sharp}_{\bot^\sharp} F^\sharp)$, assuming γ Scott-continuous.

Remark. In a Galois connection setting, γ is always co-continuous but not necessarily continuous. Nevertheless, it is sufficient that γ preserves the least upper bound for the iterates, as pointed out in Example 9.

Another result we need concerns the possibility to have a canonical concretization of a given abstract domain, inducing a Galois insertion.

Proposition 4 (Downward Embedding Abstraction)**.** *Let $\langle \mathcal{O}, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$ be a complete lattice. Let $\alpha_\vee \in \wp(\mathcal{O}) \to \mathcal{O}$ and $\gamma_\downarrow \in \mathcal{O} \to \wp(\mathcal{O})$ be defined as $\alpha_\vee(X) \triangleq \bigsqcup X$ and $\gamma_\downarrow(o) \triangleq \{o' \mid o' \sqsubseteq o\}$. Now let $\wp^\downarrow(\mathcal{O}) \triangleq \{X \subseteq \mathcal{O} \mid o \in X \Rightarrow \gamma_\downarrow(o) \subseteq X\}$ be the set of downward-closed subsets of $\mathcal{O}$. It is known that $\langle \wp^\downarrow(\mathcal{O}), \subseteq, \cup, \cap, \mathcal{O}, \{\bot\} \rangle$ is a complete lattice. Then we have the following Galois insertion:*

$$\langle \wp^\downarrow(\mathcal{O}), \subseteq \rangle \underset{\alpha_\vee}{\overset{\gamma_\downarrow}{\leftrightarrows}} \langle \mathcal{O}, \sqsubseteq \rangle$$

Proof. Trivial, since $(\wp(\mathcal{O}), \alpha_\vee, \gamma_\downarrow, \mathcal{O})$ is a Galois insertion. □

An example of this kind of abstraction is given by the adjoint functions $\langle \alpha_\cup, \gamma_\wp \rangle$ of Chapter 4.2 between $\langle \mathtt{SSC}^{\mathtt{H}}, \subseteq \rangle$ and $\langle \mathtt{TRC}^{\mathtt{P}}, \subseteq \rangle$.

Computing Correct Hypersemantics. Now we are ready to define the hypersemantics. First, we consider the generic hypersemantics $\mathcal{H}^{\mathtt{P}}_{\mathtt{G}}$. We assume that the base semantics is constructively defined over $\langle F, \mathbb{O}, \bot \rangle$, where $\mathbb{O} = \langle \mathtt{TRC}^{\mathtt{P}}, \sqsubseteq, \sqcup \rangle$. These latter may or may not be equal to the approximation domain for trace properties $\langle \mathtt{TRC}^{\mathtt{P}}, \subseteq \rangle$. In order to define the operator on sets of sets, we can try to use the classic lift to sets, namely we can define $F_{\mathtt{G}}$ as the direct image of F. Formally: $F_{\mathtt{G}} \triangleq \lambda \mathcal{X} \,.\, \{F(X) \mid X \in \mathcal{X}\}$. The problem now is to find a computational order. Trivially, the lift does not guarantee monotonicity for $\langle \mathtt{GEN}^{\mathtt{H}}, \subseteq \rangle$. To overcome the problem we have some options. As a simple solution, we can compute $\mathcal{H}^{\mathtt{P}}_{\mathtt{G}}$ in the computational domain $\langle \wp^{\bullet}(\mathtt{TRC}^{\mathtt{P}}), \boxdot, \boxsqcup, \{\bot\} \rangle$, where $\wp^{\bullet}(\mathtt{TRC}^{\mathtt{P}}) \triangleq \{\mathcal{X} \subseteq \mathtt{TRC}^{\mathtt{P}} \mid |\mathcal{X}| = 1\}$, $\{C\} \boxdot \{C'\} \triangleq (C \sqsubseteq C')$ and $\{C\} \boxsqcup \{C'\} \triangleq \{C \sqcup C'\}$. The domain is a complete lattice and it easy to note that the iterates of $F_{\mathtt{G}}$ form an increasing $\boxdot$-chain: for every n, $F^n(\bot) \sqsubseteq F^{n+1}(\bot)$ implies that $F^n_{\mathtt{G}}(\{\bot\}) = \{F^n(\bot)\} \boxdot \{F^{n+1}(\bot)\} = F^{n+1}_{\mathtt{G}}(\{\bot\})$. Indeed $\mathcal{H}^{\mathtt{P}}_{\mathtt{G}}$ is the $\boxdot$-least fixpoint of $F_{\mathtt{G}}$ greater than $\{\bot\}$.

This solution allows us to compute the generic hypersemantics but breaks the link with hyperproperties. We aim at defining a computational domain whose carrier set is $\mathtt{GEN}^{\mathtt{H}}$ and hence useful for specifications approximation, i.e. a computational order not restricted to singletons of $\wp(\mathtt{TRC}^{\mathtt{P}})$ only, but defined for arbitrary elements in $\mathtt{GEN}^{\mathtt{H}}$. A possible way to follow is to weaken again the algebraic properties of the computational domain. As pointed out in Chapter 2, abstract interpretation could be defined over preorders, instead of partial orders. Consider the classic *Hoare ordering* $\sqsubseteq^{\mathtt{H}}$, used for the definition of the powerdomains in the denotational semantics theory [Plotkin, 1976]. This latter is defined as:

$$\mathcal{X} \sqsubseteq^{\mathtt{H}} \mathcal{Y} \triangleq (\forall C \in \mathcal{X} \, \exists C' \in \mathcal{Y} \,.\, C \sqsubseteq C')$$

Although $\sqsubseteq^{\mathtt{H}}$ is just a preorder, it is a partial order for the iterates of $F_{\mathtt{G}}$ starting from $\{\bot\}$, indeed it coincides with $\boxdot$ for them. We can hence use it as a computational (pre)order for computing $\mathcal{H}^{\mathtt{P}}_{\mathtt{G}}$, similarly to what is done in [Cousot and Cousot, 1993] in the case of strictness analysis.

Remark. We could use the *Smyth ordering* $\sqsubseteq^{\mathtt{S}}$ or the *Egli-Milner ordering* $\sqsubseteq^{\mathtt{E}}$, in place of the Hoare ordering. They are defined as

$$\mathcal{X} \sqsubseteq^{\mathtt{S}} \mathcal{Y} \triangleq (\forall C' \in \mathcal{Y} \, \exists C \in \mathcal{X} \,.\, C \sqsubseteq C') \quad \text{and} \quad \mathcal{X} \sqsubseteq^{\mathtt{E}} \mathcal{Y} \triangleq (\mathcal{X} \sqsubseteq^{\mathtt{H}} \mathcal{Y} \wedge \mathcal{X} \sqsubseteq^{\mathtt{S}} \mathcal{Y})$$

For our purposes, all these orderings are equivalent, in the sense that in all cases we have that they are partial orders for the iterates of $F_{\mathtt{G}}$.

The (partial) least upper bound operator $\boxplus$ is simply defined as the component-wise lift of $\sqcup$, namely $\boxplus_{i \in \Delta} \mathcal{X}_i \triangleq \{\bigsqcup_{i \in \Delta} X_i \mid \forall i \in \Delta \,.\, X_i \in \mathcal{X}_i\}$. For every $\sqsubseteq^{\mathtt{H}}$-chain it computes exactly the least upper bound. Hence we have that $\langle F_{\mathtt{G}}, \mathbb{O}^{\mathtt{G}}, \{\bot\} \rangle$, where $\mathbb{O}^{\mathtt{G}} \triangleq \langle \mathtt{GEN}^{\mathtt{H}}, \sqsubseteq^{\mathtt{H}}, \boxplus \rangle$ is a computational fixpoint definition. Indeed we have:

$$\mathcal{H}^{\mathtt{P}}_{\mathtt{G}} = \mathsf{lfp}^{\sqsubseteq^{\mathtt{H}}}_{\{\bot\}} F_{\mathtt{G}} = \Big\boxplus^{\mathtt{H}}_{n<\omega} F^n_{\mathtt{G}}(\{\bot\})$$

Now, we consider the case of the subset-closed hypersemantics $\mathcal{H}^{\mathtt{P}}_{\mathtt{S}}$. Suppose that the base semantics is constructively defined over $\langle F, \mathbb{O}, \varnothing \rangle$, where $\mathbb{O} = \langle \mathtt{TRC}^{\mathtt{P}}, \subseteq, \cup \rangle$ coincides with the approximation order. Applying Proposition 4, we have the Galois insertion

$$\langle \mathtt{SSC}^{\mathtt{H}}, \subseteq \rangle \underset{\alpha_{\cup}}{\overset{\gamma_{\wp}}{\leftrightarrows}} \langle \mathtt{TRC}^{\mathtt{P}}, \subseteq \rangle$$
$$\alpha_{\cup} = \alpha_{\vee}(\mathcal{X}) = \bigcup \mathcal{X} \text{ and } \gamma_{\wp} = \gamma_{\downarrow}(X) = \{Y \mid Y \subseteq X\}$$

due to the fact that $\mathtt{SSC}^{\mathtt{H}} = \wp^{\downarrow}(\mathtt{TRC}^{\mathtt{P}})$. Now consider the following computational fixpoint definition $\langle F_{\mathtt{s}}, \mathbb{O}^{\mathtt{s}}, \{\varnothing\}\rangle$, where $F_{\mathtt{s}} = \gamma_{\wp} \circ F \circ \alpha_{\cup}$ and $\mathbb{O}^{\mathtt{s}} \triangleq \langle \mathtt{SSC}^{\mathtt{H}}, \subseteq, \cup\rangle$. The concretization $\gamma_{\wp}$ is not continuous, but it preserves the join for the iterates of F (see Example 9). Hence, we can still apply Theorem 19 considering $F_{\mathtt{s}}$ as the best complete concretization of F. The commutation condition holds by definition, so we have that $\gamma_{\wp}(\mathsf{lfp}^{\subseteq}_{\varnothing} F) = \mathsf{lfp}^{\subseteq}_{\{\varnothing\}} F_{\mathtt{s}}$, namely:

$$\mathsf{lfp}^{\subseteq}_{\{\varnothing\}} F_{\mathtt{s}} = \gamma_{\wp}(\mathsf{lfp}^{\subseteq}_{\varnothing} F) = \gamma_{\wp}(\mathcal{S}^{\mathtt{P}}_{\mathtt{base}}) = \wp(\mathcal{S}^{\mathtt{P}}_{\mathtt{base}}) = \mathcal{H}^{\mathtt{P}}_{\mathtt{s}}$$

Indeed $F_{\mathtt{s}}$ is $\subseteq$-monotone and $F^0_{\mathtt{s}}(\{\varnothing\}) = \{\varnothing\} \subseteq F^1_{\mathtt{s}}(\{\varnothing\}) = \wp(F(\varnothing)) \subseteq F^2_{\mathtt{s}}(\{\varnothing\}) = \wp(F^2(\varnothing)) \subseteq \ldots F^n_{\mathtt{s}}(\{\varnothing\}) = \wp(F^n(\varnothing))$ since, for every n, $F^n(\varnothing) \subseteq F^{n+1}(\varnothing)$. The operator $F_{\mathtt{s}}$ reaches a fixpoint at the same ordinal as F, namely:

$$\mathcal{H}^{\mathtt{P}}_{\mathtt{s}} = \mathsf{lfp}^{\subseteq}_{\{\varnothing\}} F_{\mathtt{s}} = \bigcup_{n<\omega} F^n_{\mathtt{s}}(\{\varnothing\})$$

When the computational domain of the base semantics is different from the approximation order, things are more complicated. Suppose that the base semantics is constructively defined over $\langle F, \mathbb{O}, \bot\rangle$, where $\mathbb{O} = \langle \mathtt{TRC}^{\mathtt{P}}, \sqsubseteq, \sqcup\rangle$. In this case, applying Proposition 4, we obtain $\gamma_{\downarrow}(\mathcal{S}^{\mathtt{P}}_{\mathtt{base}}) = \{X \mid X \sqsubseteq \mathcal{S}^{\mathtt{P}}_{\mathtt{base}}\}$ which is different from $\wp(\mathcal{S}^{\mathtt{P}}_{\mathtt{base}})$. Hence, also in this case, we have to weaken the algebraic properties of the computational domain, similarly to what we have done for the generic hypersemantics. The solution consists in setting the computational fixpoint definition to $\langle F_{\mathtt{s}}, \mathbb{O}^{\mathtt{s}}, \{\bot\}\rangle$, where $F_{\mathtt{s}} = \gamma_{\wp} \circ F \circ \alpha_{\cup}$ and $\mathbb{O}^{\mathtt{s}} \triangleq \langle \mathtt{GEN}^{\mathtt{H}}, \uplus, \Cup\rangle$. The partial least upper bound operator is given by the best correct approximation of $\sqcup$, i.e.

$$\Cup\{\mathcal{X}_i\}_{i\in\Delta} \triangleq \gamma_{\wp}(\bigsqcup\{\alpha_{\cup}(\mathcal{X}_i)\}_{i\in\Delta}) = \wp(\bigsqcup\{\bigcup \mathcal{X}_i\}_{i\in\Delta})$$

The operator $F_{\mathtt{s}}$ is $\uplus$-monotone (and increasing) and its iterates from $\{\bot\}$ stabilize at the same ordinal of F, namely

$$\mathcal{H}^{\mathtt{P}}_{\mathtt{s}} = \mathsf{lfp}^{\uplus}_{\{\bot\}} F_{\mathtt{s}} = \Cup_{n<\omega} F^n_{\mathtt{s}}(\{\bot\})$$

Remark. In this case, we had to use $\mathtt{GEN}^{\mathtt{H}}$, instead of $\mathtt{SSC}^{\mathtt{H}}$, as the carrier set for the computational fixpoint definition since the set $\{\bot\}$, i.e. the starting point of the iterates, is not subset-closed, hence $\{\bot\} \notin \mathtt{SSC}^{\mathtt{H}}$.

A Three Dimensions Hierarchy of Semantics. Up to now, we simply reasoned on single semantics. Finally, we can show that the whole hierarchy of base semantics can be translated at the hyperlevel, preserving all the abstraction relations between semantics. In the classic hierarchy, $\tau^{\vec{\infty}}$, $\tau^{\vec{+}}$ and $\tau^{\vec{\omega}}$ (and hence all their relational abstractions) are *backward* semantics in the sense that they are suffix-closed [Cousot and Cousot, 2001]. This means that they represent systems executions with complete traces and all their suffixes. Instead, the semantics $\tau^{\vec{\propto}}$ (and its abstraction) is *forward* in the sense that it is prefix-closed [Cousot and Cousot, 2012]. This means that it represents systems executions with all partial computations starting from initial states (i.e. trace prefixes).

All the semantics in the classic hierarchy are abstractions of $\tau^{\vec{\infty}}$. Analogously, every generic hypersemantics is an abstraction of $\tau^{\vec{\infty}}_{\mathtt{g}}$ and every subset-closed hypersemantics is an abstraction of $\tau^{\vec{\infty}}_{\mathtt{s}}$.

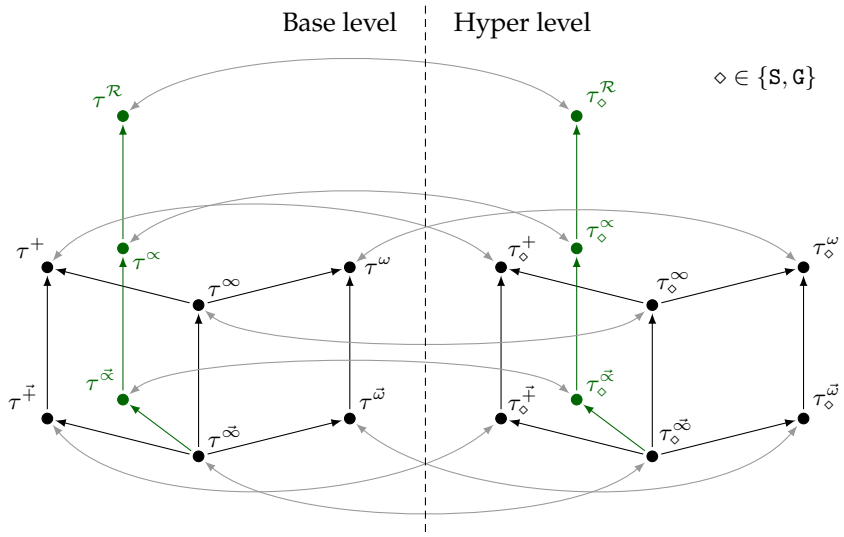


Figure 6.3: A part of the classic hierarchy of semantics with its hyper counterpart.

Proposition 5. *Let* $\circledast \in \{\vec{\infty}, \vec{+}, \vec{\omega}, \infty, +, \omega, \vec{\propto}, \propto, \mathcal{R}\}$*, let* α *be such that* $\tau^{\circledast} = \alpha(\tau^{\vec{\infty}})$ *in the classic hierarchy of semantics, and let* $\hat{\alpha}$ *its direct image, then:*

$$\tau^{\circledast}_{\mathtt{G}} = \hat{\alpha}(\tau^{\vec{\infty}}_{\mathtt{G}}) \text{ and } \tau^{\circledast}_{\mathtt{S}} = \hat{\alpha}(\tau^{\vec{\infty}}_{\mathtt{S}})$$

Proof.

$$\hat{\alpha}(\tau^{\vec{\infty}}_{\mathtt{G}}) = \alpha(\{\tau^{\vec{\infty}}\}) = \{\alpha(\tau^{\vec{\infty}})\} = \{\tau^{\circledast}\} = \tau^{\circledast}_{\mathtt{G}}$$

$$\hat{\alpha}(\tau^{\vec{\infty}}_{\mathtt{S}}) = \hat{\alpha}(\wp(\tau^{\vec{\infty}})) = \{\alpha(X) \mid X \subseteq \tau^{\vec{\infty}}\} = \{X \mid X \subseteq \alpha(\tau^{\vec{\infty}})\} = \wp(\tau^{\circledast}) = \tau^{\circledast}_{\mathtt{S}}$$

□

So, lifting to sets the abstraction function used to go from a semantics to another semantics, in the classic hierarchy, results in an abstraction between the respective hypersemantics at the hyperlevel. Furthermore, generic and subset-closed hypersemantics are isomorphic to the base ones.

Proposition 6. *Let* $\circledast \in \{\vec{\infty}, \vec{+}, \vec{\omega}, \infty, +, \omega, \vec{\propto}, \propto, \mathcal{R}\}$*,* $\iota^{\widehat{\mathtt{S}\mathtt{T}}} \triangleq \alpha_{\cup}$*,* $\iota^{\widehat{\mathtt{T}\mathtt{S}}} \triangleq \gamma_{\wp}$*,* $\iota^{\widehat{\mathtt{G}\mathtt{S}}} \triangleq \rho_{\mathtt{S}}$ *and* $\iota^{\widehat{\mathtt{S}\mathtt{G}}} \triangleq \lambda\mathcal{X} \,.\, \{\bigcup \mathcal{X}\}$*, then:*

$$\tau^{\circledast} = \iota^{\widehat{\mathtt{S}\mathtt{T}}}(\tau^{\circledast}_{\mathtt{S}}) = \iota^{\widehat{\mathtt{S}\mathtt{T}}} \circ \iota^{\widehat{\mathtt{G}\mathtt{S}}}(\tau^{\circledast}_{\mathtt{G}}) \text{ and } \tau^{\circledast}_{\mathtt{G}} = \iota^{\widehat{\mathtt{S}\mathtt{G}}}(\tau^{\circledast}_{\mathtt{S}}) = \iota^{\widehat{\mathtt{S}\mathtt{G}}} \circ \iota^{\widehat{\mathtt{T}\mathtt{S}}}(\tau^{\circledast})$$

The isomorphism is between single semantics, not between the domains: $\mathtt{SSC}^{\mathtt{H}}$ is strictly more expressive than $\mathtt{TRC}^{\mathtt{P}}$ and $\mathtt{GEN}^{\mathtt{H}}$ is strictly more expressive than $\mathtt{SSC}^{\mathtt{H}}$. Even if the hypersemantics are isomorphic to base ones, they allows us, with the same information, to gain expressiveness in verification. Propositions 5, 6 justify Figure 6.3, where an arrow between semantics means that there is an abstraction relation, while a double arrow means that the semantics are isomorphic. On the left we have the base level, corresponding to the (extended) classic hierarchy, and on the right the hyper level, corresponding to hypersemantics.

6.1.4.2 Post/Pre Hypersemantics

In the previous sections we have seen two intuitive ways for extending base semantics to sets of sets. Clearly, there are a lot of other possibilities and, in this section, we see how to define hypersemantics useful for partial verification. In particular, we define the post and pre hypersemantics for the more concrete cases where $\mathbb{D}\mathtt{en} = \Sigma^{\vec{\infty}}$, for backward semantics, and $\mathbb{D}\mathtt{en} = \Sigma^{\vec{\alpha}}$, for forward semantics. The *Post hypersemantics* $\tau^{\vec{\infty}}_{\mathtt{post}}$ is defined as:

$$\tau^{\vec{\infty}}_{\mathtt{post}} \triangleq \left\{ \left\{ \bigcup_{n>0} \tau^{\vec{n}}_X \cup \tau^{\vec{\omega}} \right\} \;\middle|\; X \subseteq \Omega \right\} \qquad \text{where } \tau^{\vec{n}}_X \triangleq \{\sigma \in \tau^{\vec{n}} \mid \sigma_{n-1} \in X\}$$

The *Pre hypersemantics* $\tau^{\vec{\alpha}}_{\mathtt{pre}}$ is defined as:

$$\tau^{\vec{\alpha}}_{\mathtt{pre}} \triangleq \left\{ \left\{ \bigcup_{n>0} \tau^{\dot{\vec{n}}}_X \right\} \;\middle|\; X \subseteq \Upsilon \wedge X \neq \varnothing \right\} \qquad \text{where } \tau^{\dot{\vec{n}}}_X \triangleq \{\sigma \in \tau^{\dot{\vec{n}}} \mid \sigma_0 \in X\}$$

The first collects the sets of terminating computations partitioned by all possible sets of final states, plus the infinite computations of course. This is a backward semantics and intuitively says which initial states we need to take in order to reach some given final states. The second does the opposite, namely it collects the sets of partial (finite) computations partitioned by all the possible sets of initial states. This is a forward semantics and intuitively says which partial computations we obtain starting from some given initial states.

Example 10. As example, consider the transition system with:

- $\Sigma = \{a, b, c, d, e\}$;
- $\tau = \{\langle a, b\rangle, \langle a, c\rangle, \langle b, d\rangle, \langle c, c\rangle, \langle e, b\rangle, \langle e, e\rangle\}$;
- $\Upsilon = \{a, e\}$;
- $\Omega = \{d\}$.

Then the maximal trace semantics and the partial trace semantics are

$$\tau^{\vec{\infty}} = \{d, bd, abd\} \cup \{e^n bd \mid n \geq 1\} \cup \{c^\omega, ac^\omega, e^\omega\}$$
$$\tau^{\vec{\alpha}} = \{a, ab, abd\} \cup \{ac^n \mid n \geq 1\} \cup \{e^n \mid n \geq 1\} \cup \{e^n b \mid n \geq 1\} \cup \{e^n bd \mid n \geq 1\}$$

The hyper versions are

$$\tau^{\vec{\infty}}_{\mathtt{post}} = \left\{\tau^{\vec{\infty}}, \{c^\omega, ac^\omega, e^\omega\}\right\}$$
$$\tau^{\vec{\alpha}}_{\mathtt{pre}} = \left\{\tau^{\vec{\alpha}}, \{a, ab, abd\} \cup \{ac^n \mid n \geq 1\}, \{e^n \mid n \geq 1\} \cup \{e^n b \mid n \geq 1\} \cup \{e^n bd \mid n \geq 1\}\right\}$$

being $\wp(\Omega) = \left\{\{d\}, \varnothing\right\}$ and $\wp(\Upsilon) \setminus \{\varnothing\} = \left\{\{a, e\}, \{a\}, \{e\}\right\}$.

These hypersemantics can be used for *partially verifying hyperproperties*, since they provide the semantics parametrically on the subsets of blocking/initial states. Suppose that, instead of checking *whether* a program fulfills a hyperproperty $\mathfrak{Hp}$, we want to check *when* a program fulfills it. The problem boils down to analyze the intersection $\tau^{\vec{\infty}}_{\mathtt{post}} \cap \mathfrak{Hp}$ [or $\tau^{\vec{\alpha}}_{\mathtt{pre}} \cap \mathfrak{Hp}$]. If the intersection is $\varnothing$ then the answer is "never", if the answer is $\tau^{\vec{\infty}}_{\mathtt{post}}$ [or $\tau^{\vec{\alpha}}_{\mathtt{pre}}$] then $\mathsf{P} \models \mathfrak{Hp}$,

otherwise we have that for particular final states [initial states] the system satisfies the hyperproperty. Hence we have a form of *partial satisfiability*. This is useful, for example when we want to know under what conditions we can still use an unsafe system.

Computing Post/Pre Hypersemantics. As for the other semantics, also pre and post hypersemantics are constructive, hence we give now their computational fixpoint definitions. For the post hypersemantics we have $\langle F^{\vec{\infty}}_{\mathtt{post}}, \mathbb{O}, \{\bot^{\vec{\infty}}\}\rangle$, where $\mathbb{O} \triangleq \langle \mathtt{GEN}^{\mathtt{H}}, \underline{\boxplus}, \tilde{\sqcup}\rangle$ and[4]

$$F^{\vec{\infty}}_{\mathtt{post}} \triangleq \lambda\mathcal{X}\,.\,\{X \cup \Sigma^{\vec{\omega}} \mid X \subseteq \Omega\}\,\tilde{\sqcup}\,\{X \sqcup \tau^{\dot{\vec{2}}} \frown X \mid X \in \mathcal{X}\}$$

The partial least upper bound operator is defined, for every non-empty $\mathcal{X}, \mathcal{Y} \in \mathtt{GEN}^{\mathtt{H}}$, as:

$$\mathcal{X}\,\tilde{\sqcup}\,\mathcal{Y} \triangleq \begin{array}{l}\{X \sqcup Y \mid X \in \mathcal{X} \wedge Y \in \mathcal{Y} \wedge (X \sqsubseteq Y \vee Y \sqsubseteq X)\} \cup \\ \{X \in \mathcal{X} \mid \forall Y \in \mathcal{Y}\,.\,X \not\sqsubseteq Y \wedge Y \not\sqsubseteq X\} \cup \{Y \in \mathcal{Y} \mid \forall X \in \mathcal{X}\,.\,Y \not\sqsubseteq X \wedge X \not\sqsubseteq Y\}\end{array}$$

It basically makes the union of the elements of $\mathcal{X}$ and $\mathcal{Y}$ which are in relation $\sqsubseteq$, and adds all other elements of both sets, as they are. For the pre hypersemantics we have $\langle F^{\vec{\propto}}_{\mathtt{pre}}, \mathbb{O}, \{\varnothing\}\rangle$, where $\mathbb{O} \triangleq \langle \mathtt{GEN}^{\mathtt{H}}, \underline{\boxplus}, \tilde{\cup}\rangle$ and

$$F^{\vec{\propto}}_{\mathtt{pre}} \triangleq \lambda\mathcal{X}\,.\,\{X \subseteq \Upsilon \mid X \neq \varnothing\}\,\tilde{\cup}\,\{X \cup X \frown \tau^{\dot{\vec{2}}} \mid X \in \mathcal{X}\}$$

The partial least upper bound operator is defined, for every non-empty $\mathcal{X}, \mathcal{Y} \in \mathtt{GEN}^{\mathtt{H}}$, as:

$$\mathcal{X}\,\tilde{\cup}\,\mathcal{Y} \triangleq \begin{array}{l}\{X \cup Y \mid X \in \mathcal{X} \wedge Y \in \mathcal{Y} \wedge (X \subseteq Y \vee Y \subseteq X)\} \cup \\ \{X \in \mathcal{X} \mid \forall Y \in \mathcal{Y}\,.\,X \not\subseteq Y \wedge Y \not\subseteq X\} \cup \{Y \in \mathcal{Y} \mid \forall X \in \mathcal{X}\,.\,Y \not\subseteq X \wedge X \not\subseteq Y\}\end{array}$$

It follows the same intuition as the operator for the post hypersemantics. For the iterates of $F^{\vec{\infty}}_{\mathtt{post}}$ and $F^{\vec{\propto}}_{\mathtt{pre}}$ we have that $\underline{\boxplus}$ and $\underline{\boxplus}$ are partial orders and the iterates are increasing. Furthermore, both operators stabilize in at most a countable number of steps, indeed:

$$\tau^{\vec{\infty}}_{\mathtt{post}} = \mathsf{lfp}^{\underline{\boxplus}}_{\{\bot^{\vec{\infty}}\}} F^{\vec{\propto}}_{\mathtt{post}} = \tilde{\bigsqcup}_{n<\omega} {F^{\vec{\infty}}_{\mathtt{post}}}^{n}(\{\bot^{\vec{\infty}}\}) \quad \text{and} \quad \tau^{\vec{\propto}}_{\mathtt{pre}} = \mathsf{lfp}^{\underline{\boxplus}}_{\{\varnothing\}} F^{\vec{\propto}}_{\mathtt{pre}} = \tilde{\bigcup}_{n<\omega} {F^{\vec{\propto}}_{\mathtt{pre}}}^{n}(\{\varnothing\})$$

Analyzing Analyses. Pre and Post hypersemantics do not only allow us to provide weaker forms of satisfiability, but they provide a promising methodology allowing us to lift static analyses (for hyperproperties) directly at the hyper level. We believe that this approach could provide a deep insight and useful formal tools also for tackling the problem of *analyzing analyzers*, aiming at systematically analyzing static analyses [Giacobazzi, Logozzo, and Ranzato, 2015; Cousot, Giacobazzi, and Ranzato, 2019].

A static analysis for invariants can be seen as the characterization, potentially approximated, of the set of reachable states $\tau^{\mathcal{R}}$ from the initial states in Υ, which provides a, potentially approximated, invariant of the program. Very often, we are interested in a restricted invariant, namely on the reachable states $\tau^{\mathcal{R}}|_I$ originated from a subset $I \subseteq \Upsilon$ of initial states. The most common static analyzers compute this information by a system of equations associating each control point with a set of memories (the invariant), as

[4] Here $\sqcup$ is the least upper bound operator used to compute the maximal trace semantics τ^{∞}.

we have seen in Chapter 5. This means that they compute the state semantics in its form $\tau^{\mathcal{R}}_{\ell} \in \mathsf{Lab} \to \wp(\mathsf{Mem})$, or $\tau^{\mathcal{R}}_{\ell}|_I$ if it is restricted to initial states in I.

We can observe that the semantics of an (abstract) interpreter of a program P is an abstraction of the hypersemantics of P. We have already seen that $\tau^{\mathcal{R}}$ is an abstraction of $\tau^{\vec{\propto}}$, through the function $\alpha^{\mathcal{R}} \circ \alpha^{\propto}$ (see Section 4.1.2). Then, by isomorphism, $\tau^{\mathcal{R}}_{\ell}$ is also an abstraction of $\tau^{\vec{\propto}}$, through the function $\alpha_\star \triangleq \alpha_\ell \circ \alpha^{\mathcal{R}} \circ \alpha^{\propto}$. Analogously, the semantics of an (abstract) interpreter, associating with each possible subset I of initial states, the corresponding reachable states $\tau^{\mathcal{R}}_{\ell}|_I$, is an abstraction of $\tau^{\vec{\propto}}_{\mathtt{pre}}$. As usual, we obtain *abstract* invariants in the abstract domain $\mathcal{A}$ exploiting a Galois connection $\langle \wp(\mathsf{Mem}), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$, extended to labels, namely:

$$\langle \mathsf{Lab} \to \wp(\mathsf{Mem}), \dot{\subseteq}\rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle \mathsf{Lab} \to \mathcal{A}, \dot{\preccurlyeq}\rangle$$

Proposition 7. *The semantics of the abstract interpreter, applied on a given program* P*, computing abstract invariants in* $\mathcal{A}$*, is* $\hat{\dot{\alpha}} \circ \hat{\alpha_\star}(\tau^{\vec{\propto}}_{\mathtt{pre}})$*, i.e. it is an abstraction of the Pre hypersemantics* $\tau^{\vec{\propto}}_{\mathtt{pre}}$ *of* P*.*

We recall that $\hat{\cdot}$ indicates the direct image lift, namely $\hat{\dot{\alpha}} \circ \hat{\alpha_\star} = \lambda \mathcal{X} \,.\, \{\dot{\alpha} \circ \alpha_\star(X) \mid X \in \mathcal{X}\}$. The meaning of the proposition is that if we specialize, in the sense of partial evaluation, a static analyzer on a given program P then the semantics of the specialized program is an abstraction of the Pre hypersemantics of P. Hence, analyzing $\tau^{\vec{\propto}}_{\mathtt{pre}}$ we can, indirectly, obtain information about the analyzer.

6.2 Hyper Abstract Domains

Once we have lifted the base semantics, obtaining a constructive, possibly approximated, version $\mathcal{H}^{\mathsf{P}}$of the collecting semantics $\mathcal{S}^{\mathsf{P}}_{\text{coll}}$, in order to perform verification we need to compute this latter on an abstract domain, namely we have to compute the abstract collecting semantics. As already observed, in the classic framework of abstract interpretation we can compute a sound over-approximation $O \supseteq \mathcal{S}^{\mathsf{P}}_{\text{base}}$ of a program base semantics allowing sound verification of trace properties. This is obtained by means of an abstraction of the concrete domain, where the abstract semantics plays the role of the over-approximation.

Similarly, an over-approximation $\mathcal{O} \supseteq \mathcal{H}^{\mathsf{P}}$ leads to a sound verification mechanism for hyperproperties. Let $\mathtt{GEN}^{\mathtt{H}}_{\sharp}$ (with partial order $\sqsubseteq^{\sharp}$) be an abstract domain of $\mathtt{GEN}^{\mathtt{H}}$, forming the Galois connection $(\langle \mathtt{GEN}^{\mathtt{H}}, \subseteq\rangle, \alpha, \gamma, \langle \mathtt{GEN}^{\mathtt{H}}_{\sharp}, \sqsubseteq^{\sharp}\rangle)$. Let $\mathfrak{Hp} \in \mathtt{GEN}^{\mathtt{H}}$ be a hyperproperty and $\mathcal{H}^{\mathsf{P}}_{\sharp}$ be an abstract interpretation of $\mathcal{H}^{\mathsf{P}}$ on $\mathtt{GEN}^{\mathtt{H}}_{\sharp}$, i.e. $\mathcal{H}^{\mathsf{P}} \subseteq \gamma(\mathcal{H}^{\mathsf{P}}_{\sharp})$, then $\gamma(\mathcal{H}^{\mathsf{P}}_{\sharp}) \subseteq \mathfrak{Hp}$ implies $\mathsf{P} \models \mathfrak{Hp}$. Hence, at this point, we wonder how we can define abstract domains at the hyperlevel, i.e. on sets of sets, in order to approximate hypersemantics, i.e. semantics lifted to the hyperlevel.

6.2.1 The Compositional Nature of Hyper Abstract Domains

A hyper abstract domain, or *hyperdomain*, can be decomposed basically into two parts: an inner abstraction and an outer abstraction. Note that we are not talking about a generic abstract domain on sets of sets: our focus is on the verification of hyperproperties, hence we need domains, on sets of sets, which *represent* information concerning programs, whose base semantics is on sets. Let us consider Non-Interference (NI) introduced in Chapter 3 as

running example, in order to provide the intuition beyond these concepts. Non-Interference requires that, for each set of two computations agreeing on low inputs, the low output is constant.

The *inner abstraction* approximates sets of denotations in $\mathbb{D}\mathtt{en}$, namely it says which information about program executions should be observed. In NI (supposing to have only two variables, one H and one L), for each set of computations we are interested in the constant analysis on the low variable, i.e. each set of computations (starting from states agreeing on the low variable) should be contained in a set of the form $\mathtt{C}_l \triangleq \{\bar{\sigma}\sigma \mid \sigma_{|\mathsf{H}} \in \mathbb{Z} \wedge \sigma_{|\mathsf{L}} = l\}$, where $l \in \mathbb{Z}$ is a fixed integer number.

The *outer abstraction* approximates sets of sets of denotations, namely it says which information about programs base semantics is interesting, in other words, which is the desired invariant among all the sets of computations collected. In the example, we require that all the possible resulting sets are constant in the initial or final value of low variable, hence they are a set in $\wp(\{\mathtt{C}_l \mid l \in \mathbb{Z}\})$.

It should be clear that, the outer abstraction is defined at the hyperlevel and therefore in order to compose it with the inner one, defined at the base level $\wp(\mathbb{D}\mathtt{en})$, we need to lift the inner abstraction to $\wp(\wp(\mathbb{D}\mathtt{en}))$. In this case, the *lifting function* just leverages the domain at level of sets of sets. In the case of hyperdomains, lifting a domain does not introduce computability problems, hence we can always use the additive lift. Formally, suppose the inner abstraction in $\mathcal{A}$ is given by the Galois connection

$$\langle \wp(\mathbb{D}\mathtt{en}), \subseteq \rangle \underset{\alpha_{\mathfrak{i}}}{\overset{\gamma_{\mathfrak{i}}}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq \rangle$$

The *lifting transformer* $\mathcal{L}$ is an operator lifting functions on sets to functions on sets of sets, namely $\mathcal{L} \triangleq \lambda f \,.\, \lambda \mathcal{X} \,.\, \{f(X) \mid X \in \mathcal{X}\}$. Let us consider the dual transformer $\mathcal{G}$ defined as $\mathcal{G} \triangleq \lambda f \,.\, \lambda Y \,.\, \{X \mid f(X) \in Y\}$. Due to Elementwise Set Abstraction (Section 2.3), we have that $\mathcal{L}(\alpha_{\mathfrak{i}})$ and $\mathcal{G}(\alpha_{\mathfrak{i}})$ form a Galois connection, in particular we have

$$\langle \wp(\wp(\mathbb{D}\mathtt{en})), \subseteq \rangle \underset{\mathcal{L}(\alpha_{\mathfrak{i}})}{\overset{\mathcal{G}(\alpha_{\mathfrak{i}})}{\leftrightarrows}} \langle \wp(\mathcal{A}), \subseteq \rangle$$

We obtained so far, starting form the inner abstraction defined on the base level and applying the additive lift, the hyperdomain where we can define the outer abstraction. In other words, the outer abstraction is a further abstraction of $\wp(\mathcal{A})$ given by some Galois connection

$$\langle \wp(\mathcal{A}), \subseteq \rangle \underset{\alpha_{\mathfrak{o}}}{\overset{\gamma_{\mathfrak{o}}}{\leftrightarrows}} \langle \mathcal{A}_{\mathfrak{h}}, \preccurlyeq_{\mathfrak{h}} \rangle$$

This outer abstraction captures the information that must be invariant among all the collected sets of executions (abstracted in $\mathcal{A}$), looking, by construction, for invariants among elements of $\mathcal{A}$. Finally, by composition, we have that

$$\langle \wp(\wp(\mathbb{D}\mathtt{en})), \subseteq \rangle \underset{\alpha_{\mathfrak{o}} \circ \mathcal{L}(\alpha_{\mathfrak{i}})}{\overset{\mathcal{G}(\alpha_{\mathfrak{i}}) \circ \gamma_{\mathfrak{o}}}{\leftrightarrows}} \langle \mathcal{A}_{\mathfrak{h}}, \preccurlyeq_{\mathfrak{h}} \rangle$$

In Figure 6.4 we have a graphical representation of how a hyperdomain works. Note that, it is not mandatory, for the inner abstraction in $\mathcal{A}$, to form a Galois connection. Indeed, in order to apply the lifting transformer, the abstraction function $\alpha_{\mathfrak{i}}$ may also fail additivity (it may even fail monotonicity). This implies that we can build and hyperdomain starting from

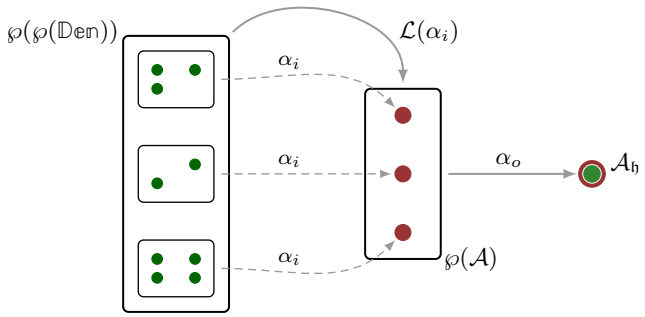


Figure 6.4: Three layers abstraction.

a base analysis formalized with just an abstraction function or a concretization function. In the first case we have just to apply the lifting method explained above. If we have a concretization-based abstract interpretation, we have to use the lifting function $\mathcal{L}$ on the concretization instead of the abstraction. Suppose to have a concretization $\gamma \in \mathcal{A} \to \wp(\mathbb{D}\mathtt{en})$, then the function $\mathcal{L}(\gamma) \triangleq \lambda X \,.\, \{\gamma(a) \mid a \in X\}$ is co-additive. Indeed:

$$\begin{aligned}\mathcal{L}(\gamma)(\textstyle\bigcap_{i\in\Delta} X_i) &= \{\gamma(a) \mid a \in \textstyle\bigcap_{i\in\Delta} X_i\} = \\ &= \{\gamma(a) \mid \textstyle\bigwedge_{i\in\Delta} a \in X_i\} = \\ &= \textstyle\bigcap_{i\in\Delta}\{\gamma(a) \mid a \in X_i\} = \textstyle\bigcap_{i\in\Delta} \mathcal{L}(\gamma)(X_i)\end{aligned}$$

Then we have that its adjoint $\mathcal{L}(\gamma)^+$ exists and hence we have the Galois connection

$$\langle \wp(\wp(\mathbb{D}\mathtt{en})), \subseteq\rangle \underset{\mathcal{L}(\gamma)^+}{\overset{\mathcal{L}(\gamma)}{\leftrightarrows}} \langle \wp(\mathcal{A}), \subseteq\rangle$$

Example 11. The abstract domains defined in [Assaf et al., 2017] are instances of the pattern proposed here. For example, the cardinality abstraction $\mathtt{crdval} \in \wp(\mathbb{Z}) \to [0, \infty]$ (which is not continuous) corresponds to our inner abstraction, while $\alpha_{\max} \in \wp([0, \infty]) \to [0, \infty]$ computing the least upper bound, i.e. $\alpha_{\max}(X) \triangleq \max X$, is the outer abstraction. The resulting abstraction is obtained by lifting the inner one and composing it with the outer one, i.e. $\alpha_{\mathtt{crdval}} \in \wp(\wp(\mathbb{Z})) \to [0, \infty]$ coincides with $\alpha_{\max} \circ \mathcal{L}(\mathtt{crdval})$, which is the process we have generalized above.

In the following, we give some constructions useful to define hyperdomains, starting from initial known abstractions on sets.

6.2.1.1 Dealing with Constants Propagation

Suppose to define a hyperanalysis on the concrete domain $\wp(\wp(\mathbb{Z}))$, and to be interested in hyper constant propagation. Hence, the goal is to verify that all the sets of computations provide constant results. This corresponds intuitively to an inner abstraction which is the classic constant propagation (lifted as shown before), and an outer abstraction retrieving

information about the constant analysis at base level. The classic domain of constants $\mathtt{C} \triangleq \mathbb{Z} \cup \{\bot, \top\}$ is defined by the Galois insertion $(\langle \wp(\mathbb{Z}), \subseteq \rangle, \alpha_{\mathtt{c}}, \gamma_{\mathtt{c}}, \langle \mathtt{C}, \preceq \rangle)$ where $\mathtt{c}_1 \preceq \mathtt{c}_2 \triangleq (\mathtt{c}_1 = \bot \vee \mathtt{c}_1 = \mathtt{c}_2 \vee \mathtt{c}_2 = \top)$ and

$$\alpha_{\mathtt{c}} \triangleq \lambda X \,.\, \begin{cases} \bot & \text{if } X = \varnothing \\ n & \text{if } X = \{n\} \\ \top & \text{otherwise} \end{cases} \quad \gamma_{\mathtt{c}} \triangleq \lambda \mathtt{c} \,.\, \begin{cases} \varnothing & \text{if } \mathtt{c} = \bot \\ \{n\} & \text{if } \mathtt{c} = n \\ \mathbb{Z} & \text{otherwise} \end{cases}$$

In order to get an abstract domain on sets of sets we rely on the lifting transformer, obtaining the following Galois insertion

$$\langle \wp(\wp(\mathbb{Z})), \subseteq \rangle \underset{\mathcal{L}(\alpha_c)}{\overset{\mathcal{G}(\alpha_c)}{\leftrightarrows}} \langle \wp(\mathtt{C}), \subseteq \rangle$$

Now, we can naively apply the same construction used for constant propagation on sets again, namely on sets of sets, obtaining the domain $\{\{\mathtt{c}\} \mid \mathtt{c} \in \mathtt{C}\} \cup \{\varnothing, \mathtt{C}\}$, ordered by set inclusion. Clearly this domain does not give us useful information about a program. Indeed, to look for constant invariants at the hyperlevel, namely in the outer abstraction, means to check that all the collected sets of values are constants. Hence, we need to retrieve information about what there is inside the analysis at base level. This is obtained by using the Galois insertion

$$\langle \wp(\mathtt{C}), \subseteq \rangle \underset{\alpha_{\mathtt{c}}}{\overset{\gamma_{\mathtt{c}}}{\leftrightarrows}} \langle \wp(\mathbb{Z}) \cup \{\mathtt{C}\}, \subseteq \rangle \text{ where } \alpha_{\mathtt{c}}(X) \triangleq \begin{cases} X & \text{if } X \subseteq \mathbb{Z} \\ \mathtt{C} & \text{otherwise} \end{cases} \quad \gamma_{\mathtt{c}} \triangleq \mathsf{id}$$

Obtaining, by composition, the insertion

$$\langle \wp(\wp(\mathbb{Z})), \subseteq \rangle \underset{\alpha_{\mathtt{c}} \circ \mathcal{L}(\alpha_c)}{\overset{\mathcal{G}(\alpha_c) \circ \gamma_{\mathtt{c}}}{\leftrightarrows}} \langle \wp(\mathbb{Z}) \cup \{\mathtt{C}\}, \subseteq \rangle) \tag{6.3}$$

In this example, we have an outer abstraction that simply checks whether all the collected sets of computations satisfy the constant property for numerical variables, namely all the sets of computations produce constant values. We can generalize the same idea to any inner abstraction, namely we can build an outer abstraction checking whether all the collected sets of computations constantly satisfy an abstract property, fixed by the inner abstraction. We call this hyperdomain *hyper* (*abstract*) *constant propagation* of an inner abstraction.

Hyper (Abstract) Constant Propagation. Consider a lattice $\langle \mathcal{A}, \preccurlyeq, \curlyvee, \curlywedge, \top_{\mathcal{A}}, \bot_{\mathcal{A}} \rangle$, forming the Galois connection $(\langle \wp(\mathcal{C}), \subseteq \rangle, \alpha, \gamma, \langle \mathcal{A}, \preccurlyeq \rangle)$. The set of *atoms* $\mathrm{Atm}^{\mathcal{A}}$ of $\mathcal{A}$ is the set of its elements covering the bottom, i.e. $\mathrm{Atm}^{\mathcal{A}} \triangleq \{a \in \mathcal{A} \mid \forall a' \in \mathcal{A} \,.\, a' \preccurlyeq a \Rightarrow (a' = \bot^{\mathcal{A}} \vee a' = a)\}$. We assume that the concrete domain is a powerset since this is the challenging case. If the domain is not a powerset then we can just apply the constant propagation construction to the element of the domain instead of numbers.

In order to build hyper abstract constant propagation we require that the set of atoms forms a partition, by means of α, of $\mathcal{C}$, meaning that for each element $c \in \mathcal{C}$ we have that $\alpha(\{c\}) \in \mathrm{Atm}^{\mathcal{A}}$. This constraint is fulfilled by domains which are *partitioning* [Hunt and Mastroeni, 2005]. In [Hunt and Mastroeni, 2005] the authors prove that any abstract domain

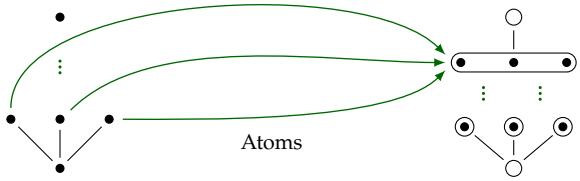


Figure 6.5: Hyper (abstract) constants domain construction.

can be made partitioning. This is actually true for domains forming a Galois insertion with $\wp(\mathcal{C})$. This is not a concern, since we can always reduce a connection to an injection. This basically means that we can always apply the following construction.

As an example, consider the abstract domain $\mathsf{Sgn} \triangleq \{\varnothing, \mathbb{Z}_{<0}, \{0\}, \mathbb{Z}_{>0}, \mathbb{Z}_{\geq 0}, \mathbb{Z}\} \subseteq \wp(\mathbb{Z})$, where $\mathbb{Z}_{<0} \triangleq \{n \in \mathbb{Z} \mid n < 0\}$ and the others are similarly defined. The set of its atoms is $\mathrm{Atm}^{\mathsf{Sgn}} = \{\mathbb{Z}_{<0}, \{0\}, \mathbb{Z}_{>0}\}$, which is a partition of $\mathbb{Z}$. In order to perform hyper constants on $\mathcal{A}$ we consider the set of its atoms because these latter precisely identify the properties of concrete values observed in $\mathcal{A}$ (in Sgn the sign of any value). The idea is to check whether these abstract values remain constant during computations. For instance, we aim at checking whether all the computations starting from inputs with the same sign, keep constant the value sign during execution. If we want to perform constants on $\wp(\mathsf{Sgn})$, we cannot allow a set to be abstracted in $\mathbb{Z}_{\geq 0}$ because if $\mathsf{Sgn}(X) = \mathbb{Z}_{\geq 0}$ then the values in X do not have the same property, i.e. they are not constant, w.r.t. Sgn. At this point, we can define the hyper (abstract) constant domain for $\mathcal{A}$ as $\mathcal{A}_{\mathfrak{hc}} \triangleq \wp(\mathrm{Atm}^{\mathcal{A}}) \cup \{\mathcal{A}\}$, forming the following Galois insertion:

$$\langle \wp(\mathcal{A}), \subseteq \rangle \underset{\alpha_{\mathfrak{hc}}}{\overset{\gamma_{\mathfrak{hc}}}{\leftrightarrows}} \langle \mathcal{A}_{\mathfrak{hc}}, \subseteq \rangle \text{ where } \alpha_{\mathfrak{hc}}(X) \triangleq \begin{cases} X & \text{if } X \subseteq \mathrm{Atm}^{\mathcal{A}} \\ \mathcal{A} & \text{otherwise} \end{cases} \quad \gamma_{\mathfrak{hc}} \triangleq \mathsf{id}$$

Then, applying the lifting transformer and composing, we have

$$\langle \wp(\wp(\mathcal{C})), \subseteq \rangle \underset{\mathcal{L}(\alpha)}{\overset{\mathcal{G}(\alpha)}{\leftrightarrows}} \langle \wp(\mathcal{A}), \subseteq \rangle \quad \langle \wp(\wp(\mathcal{C})), \subseteq \rangle \underset{\alpha_{\mathfrak{hc}} \circ \mathcal{L}(\alpha)}{\overset{\mathcal{G}(\alpha) \circ \gamma_{\mathfrak{hc}}}{\leftrightarrows}} \langle \mathcal{A}_{\mathfrak{hc}}, \subseteq \rangle \tag{6.4}$$

Back to the example where $\mathcal{C} = \mathbb{Z}$ and $\mathcal{A} = \mathsf{Sgn}$, we have that $\mathsf{Sgn}_{\mathfrak{hc}} \triangleq \wp(\{\varnothing, \mathbb{Z}_{<0}, \{0\}, \mathbb{Z}_{>0}\}) \cup \{\mathsf{Sgn}\}$ is the hyperdomain, abstraction of $\wp(\wp(\mathbb{Z}))$, for hyper (abstract) Sgn-constants. In Figure 6.5 we have a graphical representation of the hyperlevel constants construction.

6.2.1.2 Dealing with Intervals.

Suppose now to be interested in a hyper intervals analysis. The classic abstract domain of Intervals, as introduced in Section 5.2.2 is defined over numerical values, but the intervals construction can be easily generalized [Cousot and Cousot, 1979b]. Given a complete lattice $\langle \mathcal{C}, \preccurlyeq, \curlyvee, \curlywedge, \top, \bot \rangle$, we can define the $\mathcal{C}$-Intervals domain as follows.

Definition 28 ($\mathcal{C}$-Interval Domain)**.** The domain of $\mathcal{C}$-Intervals has a carrier set:

$$\mathrm{Intv}^{\mathcal{C}} \triangleq \{[a, b] \mid a \in \mathcal{C} \setminus \{\top\} \wedge b \in \mathcal{C} \setminus \{\bot\} \wedge a \preccurlyeq b\} \cup \{[\top, \bot]\}$$

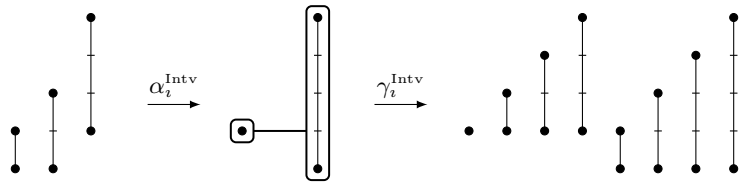


Figure 6.6: Abstraction and concretization of an Interval of Intervals.

Then the relation $\sqsubseteq_c^i \subseteq \text{Intv}^c \times \text{Intv}^c$, defined as $[a,b] \sqsubseteq_c^i [c,d] \triangleq (c \preccurlyeq a \wedge b \preccurlyeq d)$, is a partial order. The domain has arbitrary least upper bounds and greatest lower bounds, defined as:

$$
\begin{aligned}
&{}^c\curlyvee^i\{[a_i,b_i]\}_{i\in\Delta} \triangleq [\curlywedge\{a_i\}_{i\in\Delta}, \curlyvee\{a_i\}_{i\in\Delta}] \\
&{}_c\curlywedge_i\{[a_i,b_i]\}_{i\in\Delta} \triangleq \begin{cases} [\top,\bot] & \text{if } \curlyvee\{a_i\}_{i\in\Delta} \preccurlyeq \curlywedge\{a_i\}_{i\in\Delta} \\ \curlyvee\{a_i\}_{i\in\Delta}, \curlywedge\{a_i\}_{i\in\Delta} & \text{otherwise} \end{cases}
\end{aligned}
$$

The domain has a minimum $[\top,\bot]$, that we denote $\bot_c^i$, and a maximum $[\bot,\top]$, that we denote $\top_c^i$. The domain is, indeed, a complete lattice.

The corresponding Galois connection between (the powerset of) the concrete domain $\mathcal{C}$ and its intervals domain is

$$\langle \wp(\mathcal{C}), \subseteq\rangle \underset{\alpha_i^{\mathcal{C}}}{\overset{\gamma_i^{\mathcal{C}}}{\leftrightarrows}} \langle \text{Intv}^c, \sqsubseteq_c^i\rangle \text{ where } \alpha_i^c(X) \triangleq [\curlywedge X, \curlyvee X] \text{ and } \gamma_i^c([a,b]) \triangleq \{c \in \mathcal{C} \mid a \preccurlyeq c \preccurlyeq b\}$$

An instance of this pattern is the classic domain of Intervals over integers, where the initial domain is the complete lattice of Integers. This latter has a carrier set $\mathbb{Z}\cup\{-\infty,+\infty\}$, $\max$ as a supremum, $\min$ as a infimum, $-\infty$ as a minimum and $+\infty$ as a maximum.

We can use the intervals construction for an inner abstraction when we aim at characterizing invariants of intervals of computations. In this case we use the lift $\mathcal{L}$ and then we compose it with an outer abstraction determining the desired invariants. But, we can use this construction also for an outer abstraction by defining it on an domain $\mathcal{A}$ already obtained by an inner abstraction. In this case we characterize interval invariants of an inner abstract domain, abstraction of $\wp(\mathcal{A})$. For instance, if the inner abstraction is `Pos`, then we would characterize the sign properties of interval bounds.

In Figure 6.6 we have a graphical representation of the interval abstraction applied to the interval domain itself. A set of intervals is abstracted into a higher-order interval. The concretization is the set of all intervals comprised, w.r.t. the partial order of the classic Interval domain, between the left and the right interval bounds.

Example: Intervals-Parity Analysis. Consider the classic domain of Intervals on integers. Now, we can lift it at hyper level, obtaining the following Galois connection:

$$\langle \wp(\wp(\mathbb{Z})), \subseteq, \cap, \cup, \wp(\mathbb{Z}), \varnothing\rangle \underset{\mathcal{L}(\alpha_i)}{\overset{\mathcal{G}(\alpha_i)}{\leftrightarrows}} \langle \wp(\text{Intv}), \subseteq, \cup, \cap, \text{Intv}, \varnothing\rangle$$

From a set of intervals we can forget the exact relation between the bounds, instead we can collect together the information of the left bounds and the right bounds. In order to get the Intervals-Parity domain we need an intermediate abstraction. A set of intervals can be seen as a set of pairs, hence we can exploit the abstraction from sets of pairs to pairs of sets (a slightly modified version of the Componentwise Abstraction introduced in Section 2.3). This is formalized by the following Galois connection, where $\mathbb{Z}_\infty \triangleq \mathbb{Z} \cup \{-\infty, +\infty\}$:

$$\langle \wp(\mathrm{Intv}), \subseteq, \cup, \cap, \mathrm{Intv}, \varnothing \rangle \underset{\alpha_\imath^\times}{\overset{\gamma_\imath^\times}{\leftrightarrows}} \langle \wp(\mathbb{Z}_\infty)^2, \subseteq^2, \cup^2, \cap^2, \langle \mathbb{Z}_\infty, \mathbb{Z}_\infty \rangle, \langle \varnothing, \varnothing \rangle \rangle$$

$$\alpha^\times(X) \triangleq \langle \{a \mid [a,b] \in X\}, \{b \mid [a,b] \in X\} \rangle$$

$$\gamma_\times(\langle X, Y \rangle) \triangleq \begin{cases} \{[a,b] \mid a \in X, b \in Y, a \leq b\} \cup \{\bot^\imath\} & \text{if } +\infty \in X \wedge -\infty \in Y \\ \{[a,b] \mid a \in X, b \in Y, a \leq b\} & \text{otherwise} \end{cases}$$

The intervals-parity analysis retrieves the parity of the left and right bounds of a set of intervals. The Parity domain of $\mathbb{Z}_\infty$ has as carrier set $\mathtt{Par} \triangleq \{\bot, \mathrm{Even}, \mathrm{Odd}, \top\}$. The relation $\leqslant \subseteq \mathtt{Par} \times \mathtt{Par}$, defined as $\leqslant \triangleq \{\langle \bot, \mathtt{p} \rangle \mid \mathtt{p} \in \mathtt{Par}\} \cup \{\langle \mathtt{p}, \top \rangle \mid \mathtt{p} \in \mathtt{Par}\}$, is a partial order. The domain has binary infimum and supremum, trivially defined. The minimum is $\bot$ and the maximum is $\top$. Then we have the following Galois connection:

$$\langle \wp(\mathbb{Z}_\infty), \subseteq, \cup, \cap, \mathbb{Z}_\infty, \varnothing \rangle \underset{\alpha_{\mathtt{Par}}}{\overset{\gamma_{\mathtt{Par}}}{\leftrightarrows}} \langle \mathtt{Par}, \leqslant, \veebar, \barwedge, \top, \bot \rangle$$

$$\alpha_{\mathtt{Par}}(X) \triangleq \begin{cases} \bot & \text{if } X = \varnothing \\ \mathrm{Even} & \text{if } X \subseteq 2\mathbb{Z} \\ \mathrm{Odd} & \text{if } X \subseteq 2\mathbb{Z}{+}1 \\ \top & \text{otherwise} \end{cases} \qquad \gamma_{\mathtt{Par}}(\mathtt{p}) \triangleq \begin{cases} \varnothing & \text{if } \mathtt{p} = \bot \\ 2\mathbb{Z} & \text{if } \mathtt{p} = \mathrm{Even} \\ 2\mathbb{Z}{+}1 & \text{if } \mathtt{p} = \mathrm{Odd} \\ \mathbb{Z}^\infty & \text{otherwise} \end{cases}$$

Now, we can apply this abstraction componentwise to $\wp(\mathbb{Z}_\infty)$, obtaining the following Galois connection:

$$\langle \wp(\mathbb{Z}_\infty)^2, \subseteq^2, \cup^2, \cap^2, \langle \mathbb{Z}_\infty, \mathbb{Z}_\infty \rangle, \langle \varnothing, \varnothing \rangle \rangle \underset{\alpha^2_{\mathtt{Par}}}{\overset{\gamma^2_{\mathtt{Par}}}{\leftrightarrows}} \langle \mathtt{Par}^2, \leqslant^2, \veebar^2, \barwedge^2, \langle \top, \top \rangle, \langle \bot, \bot \rangle \rangle$$

$$\alpha^2_{\mathtt{Par}}(\langle X, Y \rangle) \triangleq \langle \alpha_{\mathtt{Par}}(X), \alpha_{\mathtt{Par}}(Y) \rangle \qquad \gamma^2_{\mathtt{Par}}(\langle \mathtt{p}_1, \mathtt{p}_2 \rangle) \triangleq \langle \gamma_{\mathtt{Par}}(\mathtt{p}_1), \gamma_{\mathtt{Par}}(\mathtt{p}_2) \rangle$$

Due to the fact that Galois connections are closed by composition we finally have the abstraction from the concrete domain and the domain for the Intervals-Parity analysis:

$$\langle \wp(\wp(\mathbb{Z})), \subseteq, \cup, \cap, \wp(\mathbb{Z}), \varnothing \rangle \underset{\alpha^\imath_{\mathrm{Par}}}{\overset{\gamma^\imath_{\mathrm{Par}}}{\leftrightarrows}} \langle \mathtt{Par}^2, \leqslant^2, \veebar^2, \barwedge^2, \langle \top, \top \rangle, \langle \bot, \bot \rangle \rangle$$

$$\alpha^\imath_{\mathrm{Par}} \triangleq \alpha^2_{\mathtt{Par}} \circ \alpha^\times \circ \mathcal{L}(\alpha_\imath) \qquad \gamma^\imath_{\mathrm{Par}} \triangleq \mathcal{G}(\alpha_\imath) \circ \gamma^\times \circ \gamma^2_{\mathtt{Par}}$$

Example 12. The set of sets of integers $\mathcal{X} = \{\{1,2,3\}, \{1,4\}\}$ is abstracted as follows:

$$\alpha^\imath_{\mathrm{Par}}(\mathcal{X}) = \alpha^2_{\mathtt{Par}} \circ \alpha^\times(\{[1,3],[1,4]\}) = \alpha^2_{\mathtt{Par}}(\langle \{1\}, \{3,4\} \rangle) = \langle \mathrm{Odd}, \top \rangle$$

In this case, we have that the left bounds are all even, instead the right bounds do not agree on parity.

Note that, also this hyperdomain follows the pattern described in Subsection 6.2.1. The inner abstraction is $\alpha_\imath$, then we have the additive lift $\mathcal{L}$ and, finally, the outer abstraction is $\alpha^2_{\texttt{Par}} \circ \alpha^\times$.

Example: Bound-Stability Analysis. The Bound-Stability analysis is very similar to the Intervals-Parity analysis, but it checks if the bounds of a set of intervals are stable (i.e. have constant values) instead of checking their parity. The Constants domain of $\mathbb{Z}_\infty$ has as carrier set $\texttt{Cst} \triangleq \{n \mid n \in \mathbb{Z}\} \cup \{\bot, \top\}$. The relation $\leqslant \subseteq \texttt{Cst} \times \texttt{Cst}$, defined as $\leqslant \triangleq \{\langle \bot, \texttt{c}\rangle \mid \texttt{c} \in \texttt{Cst}\} \cup \{\langle \texttt{c}, \top\rangle \mid \texttt{c} \in \texttt{Cst}\}$, is a partial order. The domain has binary infimum and supremum, trivially defined. The minimum is $\bot$ and the maximum is $\top$. Then we have the following Galois connection:

$$\langle \wp(\mathbb{Z}_\infty), \subseteq, \cup, \cap, \mathbb{Z}_\infty, \varnothing\rangle \underset{\alpha_{\texttt{Cst}}}{\overset{\gamma_{\texttt{Cst}}}{\leftrightarrows}} \langle \texttt{Cst}, \leqslant, \veebar, \barwedge, \top, \bot\rangle$$

$$\alpha_{\texttt{Cst}}(X) \triangleq \begin{cases} \bot & \text{if } X = \varnothing \\ n & \text{if } X = \{n\} \\ \top & \text{otherwise} \end{cases} \qquad \gamma_{\texttt{Cst}}(\texttt{c}) \triangleq \begin{cases} \varnothing & \text{if } \texttt{c} = \bot \\ \{n\} & \text{if } \texttt{c} = n \\ \mathbb{Z}^\infty & \text{otherwise} \end{cases}$$

Now, we can apply this abstraction componentwise to $\wp(\mathbb{Z}_\infty)$, obtaining the following Galois connection:

$$\langle \wp(\mathbb{Z}_\infty)^2, \subseteq^2, \cup^2, \cap^2, \langle \mathbb{Z}_\infty, \mathbb{Z}_\infty\rangle, \langle \varnothing, \varnothing\rangle\rangle \underset{\alpha^2_{\texttt{Cst}}}{\overset{\gamma^2_{\texttt{Cst}}}{\leftrightarrows}} \langle \texttt{Cst}^2, \leqslant^2, \veebar^2, \barwedge^2, \langle \top, \top\rangle, \langle \bot, \bot\rangle\rangle$$

$$\alpha^2_{\texttt{Cst}}(\langle X, Y\rangle) \triangleq \langle \alpha_{\texttt{Cst}}(X), \alpha_{\texttt{Cst}}(Y)\rangle \qquad \gamma^2_{\texttt{Cst}}(\langle \texttt{c}_1, \texttt{c}_2\rangle) \triangleq \langle \gamma_{\texttt{Cst}}(\texttt{c}_1), \gamma_{\texttt{Cst}}(\texttt{c}_2)\rangle$$

Due to the fact that Galois connections are closed by composition we finally have the abstraction from the concrete domain and the domain for the Bound-Stability analysis:

$$\langle \wp(\wp(\mathbb{Z})), \subseteq, \cup, \cap, \wp(\mathbb{Z}), \varnothing\rangle \underset{\alpha^\imath_{\texttt{Cst}}}{\overset{\gamma^\imath_{\texttt{Cst}}}{\leftrightarrows}} \langle \texttt{Cst}^2, \leqslant^2, \veebar^2, \barwedge^2, \langle \top, \top\rangle, \langle \bot, \bot\rangle\rangle$$

$$\alpha^\imath_{\texttt{Cst}} \triangleq \alpha^2_{\texttt{Cst}} \circ \alpha^\times \circ \mathcal{L}(\alpha_\imath) \qquad \gamma^\imath_{\texttt{Cst}} \triangleq \mathcal{G}(\alpha_\imath) \circ \gamma^\times \circ \gamma^2_{\texttt{Cst}}$$

Example 13. The set of sets of integers $\mathcal{X} = \{\{1, 2, 3\}, \{1, 4\}\}$ is abstracted as follows:

$$\alpha^\imath_{\texttt{Cst}}(\mathcal{X}) = \alpha^2_{\texttt{Cst}} \circ \alpha^\times(\{[1, 3], [1, 4]\}) = \alpha^2_{\texttt{Cst}}(\langle\{1\}, \{3, 4\}\rangle) = \langle 1, \top\rangle$$

In this case, we have that the left bound is stable, i.e. it is always 1, whilst the right bound it is not.

Also this hyperdomain follows the pattern describe in Subsection 6.2.1. The inner abstraction is $\alpha_\imath$, then we have the additive lift $\mathcal{L}$ and, finally, the outer abstraction is $\alpha^2_{\texttt{Cst}} \circ \alpha^\times$.

Again on Analyzing Analyses. The design patterns introduced in this section are not useful for hyperproperties verification only. Indeed they can also be used in the context of the analysis of analyses, as introduced at the end of Section 6.1. For instance, suppose to have an analyzer for the Intervals domain, that we call *first level analysis*. Then, collecting the Intervals subsequently generated by this static analyzer, we can infer whether the Intervals bounds are stable (indeed, we approximate this information), applying the Bound-Stability

analysis. We call this latter *second level analysis* and it builds on top of the semantics of the first order analyzer. If a bound is not stable we can apply a widening or other optimizations, in order to speed-up the (first level) analysis. As noted at the end of Section 6.1, the analysis with the Bound-Stability domain is applied on the partial evaluation of the first level analyzer on the target program.

7

Bounded Subset-Closed Hyperproperties

Subset-closed hyperproperties, introduced in Chapter 4.2, are particular hyperproperties containing only downward closed elements. In the previous chapter we have seen that these specifications are simpler to verify, indeed they are in between, for what concerns verification difficulty, generic hyperproperties and trace hyperproperties. Now we consider particular subset-closed hyperproperties, for which the verification is simplified, still not as simple as the one for trace properties.

7.1 Bounded Subset-Closed Hyperproperties

Recall that the set of subset-closed hyperproperties $\mathtt{SSC}^{\mathtt{H}}$, with typical elements $\mathfrak{chp} \in \mathtt{SSC}^{\mathtt{H}}$, is defined as:

$$\mathtt{SSC}^{\mathtt{H}} \triangleq \left\{ \mathfrak{hp} \in \mathtt{GEN}^{\mathtt{H}} \;\middle|\; X \in \mathfrak{hp} \Rightarrow (\forall Y \subseteq X \,.\, Y \in \mathfrak{hp}) \right\}$$

where $\mathtt{GEN}^{\mathtt{H}}$ is the set of all generic hyperproperties over the execution denotations domain $\mathbb{Den}$, namely $\mathtt{GEN}^{\mathtt{H}} \triangleq \wp(\wp(\mathbb{Den}))$. The strongest subset-closed hyperproperty for a program P is $\wp(\mathcal{S}^{\mathsf{P}}_{\mathsf{base}})$, meaning that these hyperproperties can be refuted by means of a single subset of the base semantics, which is a witness of refutation.

Now, we define a stronger notion of subset-closed hyperproperty, allowing us to further restrict the set of possible refuting witnesses.

Definition 29 (k-Bounded Subset-Closed Hyperproperties)**.** Given an ordinal $k < \omega$, the set of *k-bounded subset-closed hyperproperties* is:

$$\mathtt{SSC}^{\mathtt{H}}_{k} \triangleq \{ \mathfrak{chp} \in \mathtt{SSC}^{\mathtt{H}} \mid X \notin \mathfrak{chp} \Leftrightarrow (\exists T_k \subseteq X \,.\, (|T_k| \leq k \wedge T_k \notin \mathfrak{chp})) \}$$

The set T_k is the *witness of refutation*, namely a set of traces of cardinality at most k violating the specification. In other words, in a k-bounded subset-closed hyperproperty, every set of traces not satisfying the hyperproperty has a refuting witness with at most k traces. This means that, in order to refute the hyperproperty, we need to exhibit a counterexample consisting in at most k traces. Formally, suppose $\mathfrak{chp} \in \mathtt{SSC}^{\mathtt{H}}_{k}$, if we find $\{\bar{\sigma}^1, \bar{\sigma}^2, \ldots \bar{\sigma}^k\} \subseteq X$ such that $\{\bar{\sigma}^1, \bar{\sigma}^2, \ldots \bar{\sigma}^k\} \notin \mathfrak{chp}$, then we imply that $X \notin \mathfrak{chp}$. Hence $X \models \mathfrak{chp}$ if and only if $\{\{\bar{\sigma}^1, \bar{\sigma}^2, \ldots \bar{\sigma}^k\} \mid \bar{\sigma}^1, \bar{\sigma}^2, \ldots \bar{\sigma}^k \in X\} \subseteq \mathfrak{chp}$. The if and only if here is crucial: in order to refute the hyperproperty we can show just a counterexample set of cardinality k, namely $\exists \{\bar{\sigma}^1, \bar{\sigma}^2, \ldots \bar{\sigma}^k\} \subseteq X \,.\, \{\bar{\sigma}^1, \bar{\sigma}^2, \ldots \bar{\sigma}^k\} \notin \mathfrak{chp}$ implies $X \not\models \mathfrak{chp}$.

A subset-closed hyperproperty is *unbounded* when $k = \omega$, meaning that the witnesses of refutation could be infinite. It is clear that, the union of all the k-bounded and unbounded subset-closed hyperproperties is precisely the set of all the subset-closed hyperproperties.

Proposition 8. *It holds that* $\mathtt{SSC}^{\mathtt{H}} = \bigcup_{k<\omega+1} \mathtt{SSC}^{\mathtt{H}}_k$.

For every $\mathfrak{chp} \in \mathtt{SSC}^{\mathtt{H}}$ we can define a refuting set $R_{\mathfrak{chp}}$, namely a set of sets of traces representing the witnesses for refuting the specification. These sets are similar to the prefixes representing the "bad thing" in safety (hyper)properties. It is possible to define different refuting sets for a given subset-closed hyperproperty, since when a set satisfies $X \notin \mathfrak{chp}$ then we have that $X \cup Y \notin \mathfrak{chp}$, by subset-closure. A $\mathtt{SSC}^{\mathtt{H}}$ hyperproperty $\mathfrak{chp}$ is violated if and only if the given set of traces is a superset of an element in $R_{\mathfrak{chp}}$. So $\mathfrak{chp} \in \mathtt{SSC}^{\mathtt{H}}$ can be characterized as:

$$\forall X \in \wp(\mathbb{Den}) \,.\, (\exists T_r \in R_{\mathfrak{chp}} \,.\, T_r \subseteq X) \Leftrightarrow X \notin \mathfrak{chp} \tag{7.1}$$

If, in addition, $\mathfrak{chp} \in \mathtt{SSC}^{\mathtt{H}}_k$, with $k < \omega$, then we can define the *minimal* refuting set $R^{\min}_{\mathfrak{chp}}$ (i.e. the one containing the sets with minimal cardinality) characterizing the hyperproperty. The set is minimal in the sense that $\forall T_r, T'_r \in R^{\min}_{\mathfrak{chp}} \,.\, T'_r \not\subseteq T_r \vee |T'_r| \geq |T_r|$ (namely, $T'_r \subseteq T_r$ implies $T'_r = T_r$). This means that for every set violating the hyperproperty, $R^{\min}_{\mathfrak{chp}}$ contains only its minimal representative. In particular, every element in $R^{\min}_{\mathfrak{chp}}$ has cardinality k.

Example 14. Let $\mathsf{Mem} = \mathsf{Var} \to \mathbb{Z}$ and $\mathbb{Den} = \mathsf{Mem} \times \mathsf{Mem}$. Non-Interference, parametric on a security variables typing $\Gamma \in \mathsf{Var} \to \{\mathsf{L}, \mathsf{H}\}$, is:

$$\mathsf{NI} \triangleq \{X \in \wp(\mathbb{Den}) \mid \forall \bar{\sigma}, \bar{\sigma}' \in X \,.\, (\bar{\sigma}_{\vdash} =_{\mathsf{L}} \bar{\sigma}'_{\vdash} \Rightarrow \bar{\sigma}_{\dashv} =_{\mathsf{L}} \bar{\sigma}'_{\dashv})\}$$

where $\bar{\sigma}_{\vdash}$ and $\bar{\sigma}_{\dashv}$ are the projections on the first and last element of the pair $\bar{\sigma}$, respectively. The equivalence $=_{\mathsf{L}}$ holds for memories agreeing on the values of public (L) variables. NI is in $\mathtt{SSC}^{\mathtt{H}}_2$, namely $X \models \mathsf{NI}$ if and only if $\{\{\bar{\sigma}, \bar{\sigma}'\} \mid \bar{\sigma}, \bar{\sigma}' \in X\} \subseteq \mathsf{NI}$. Hence, if we find a pair of interfering executions, i.e. $\{\bar{\sigma}, \bar{\sigma}'\} \notin \mathsf{NI}$, then we proved that $X \not\models \mathsf{NI}$. Indeed, the minimal refuting set for Non-Interference is:

$$R^{\min}_{\mathsf{NI}} \triangleq \left\{\{\bar{\sigma}, \bar{\sigma}'\} \in \wp(\mathbb{Den}) \;\middle|\; \bar{\sigma}_{\vdash} =_{\mathsf{L}} \bar{\sigma}'_{\vdash} \wedge \bar{\sigma}_{\dashv} \neq_{\mathsf{L}} \bar{\sigma}'_{\dashv}\right\}$$

Remark. By substituting $\subseteq$ with the prefix-set relation $\leqslant$[1] in (7.1) we obtain the minimal refuting set for an k-hypersafety.

In the rest of the work, when we say bounded hyperproperty, we mean a k-bounded subset-closed hyperproperty, for some $k < \omega$. It is worth noting, that we can restate the satisfiability for bounded hyperproperties.

Proposition 9. *A program* P*, with base semantics* $\mathcal{S}^{\mathsf{P}}_{\mathsf{base}}$ *satisfies a* k*-bounded subset-closed hyperproperty* $\mathfrak{chp}$ *if and only if* $\{X \subseteq \mathcal{S}^{\mathsf{P}}_{\mathsf{base}} \mid |X| = k\} \subseteq \mathfrak{chp}$.

We can then approximate $\{X \subseteq \mathcal{S}^{\mathsf{P}}_{\mathsf{base}} \mid |X| = k\}$ in order to verify the hyperproperty. The set $\{X \subseteq \mathcal{S}^{\mathsf{P}}_{\mathsf{base}} \mid |X| = k\}$ is not a correct hypersemantics, since it does not contain $\mathcal{S}^{\mathsf{P}}_{\mathsf{base}}$. Nevertheless, it is equisatisfying to the collecting semantics $\{\mathcal{S}^{\mathsf{P}}_{\mathsf{base}}\}$ w.r.t. $\mathtt{SSC}^{\mathtt{H}}_k$. We will see how to exploit this fact in a moment.

Note that a k-bounded subset-closed hyperproperty with $k = \omega$ is not simpler to verify than an arbitrary subset-closed hyperproperty. Indeed, simplifications occur, as always, when from infinite objects we move to finite ones.

[1] Here $X \leqslant Y$ if and only if for every $\bar{\sigma} \in X$ exists $\bar{\sigma}' \in Y$ such that $\bar{\sigma}$ is a prefix of $\bar{\sigma}'$ (see Section 4.2).

7.1.1 Expressiveness

In the context of trace properties, a particular kind of properties are the *safety* ones [Alpern and Schneider, 1985], expressing the fact that "nothing bad happens". These properties are interesting because they depend only on the history of single executions, meaning that safety properties are dynamically monitorable [Alpern and Schneider, 1985]. Similarly, *safety hyperproperties* (or hypersafety) are the lift to sets of safety properties. This means that, for each set of executions that is not in a safety hyperproperty, there exists a finite prefix-set of finite executions (the "bad thing") which cannot be extended to satisfy the property. Dually, liveness (trace) properties express the fact that "something good eventually happens", namely the systems satisfying a liveness property are those that, eventually, exhibit a good behavior. Again, *liveness hyperproperties* (or hyperliveness) are the lift to sets of liveness properties. This means that a set of finite traces can be extended to a set of infinite traces satisfying the property. An interesting aspect of the safety/liveness dichotomy is that every trace property can be expressed as the intersection of a safety and a liveness one. This also holds for hyperproperties, i.e. every hyperproperty can be expressed as the intersection of a hypersafety and a hyperliveness one, as we explained in Section 4.2.

Another particular class of hyperproperties are the ones formed by the *k-safety hyperproperties* (or k-hypersafety). They are safety hyperproperties in which the "bad thing" never involves more than k executions [Clarkson and Schneider, 2010]. This means that it is possible to check the violation of a k-hypersafety just observing a set of k executions (note that 1-hypersafety are exactly safety properties). This is important for verification, in fact, it is possible to reduce the verification of a k-hypersafety on a system S to the verification of a safety on the self-composed system S^k [Clarkson and Schneider, 2010].

It turns out that all hypersafety are subset-closed [Clarkson and Schneider, 2010]. Also some hyperliveness are subset-closed, in fact every trace hyperproperty is subset-closed and hence every liveness property, which is an hyperliveness, is in $\mathtt{SSC}^{\mathtt{H}}$. Every k-hypersafety is k-bounded and every liveness is a 1-bounded subset-closed hyperproperty. But there are other hyperliveness which are bounded, as we can see in the next example.

Example 15. Suppose now that executions denotations are infinite sequences of states, namely $\mathbb{D}\mathtt{en} = \mathsf{Mem}^{\omega}$. Suppose also that the systems of interest can receive requests and can provide responses to these requests. We denote with the predicate $\mathrm{Req}(\mathfrak{d}, i)$ the fact that a system, in the execution $\mathfrak{d}$, has received a request at time i, namely in the state $\mathfrak{d}_i$. Analogously, we denote with the predicate $\mathrm{Resp}(\mathfrak{d}, i, j)$ the fact that the system has provided a response at time j to the request received at time i. Then we can define a policy saying that if the executions of a system receive a request at time i then they have to provide a response at time j, meaning that if they receive a request at the same time then they have to respond at the same time. Formally:

$$\mathtt{SyncR} \triangleq \left\{ X \subseteq \mathsf{Mem}^{\omega} \;\middle|\; \forall \mathfrak{d}, \mathfrak{d}' \in X \, \forall i \in \mathbb{N} \,.\; \begin{array}{l} (\mathrm{Req}(\mathfrak{d}, i) \wedge \mathrm{Req}(\mathfrak{d}', i)) \Rightarrow \\ \exists j \in \mathbb{N} \,.\, (\mathrm{Resp}(\mathfrak{d}, i, j) \wedge \mathrm{Resp}(\mathfrak{d}', i, j)) \end{array} \right\}$$

It is easy to note that SyncR is subset-closed but it is not an hypersafety. Indeed it is an hyperliveness, but it is also a bounded subset-closed hyperproperty. In particular, it is in $\mathtt{SSC}^{\mathtt{H}}_2$: in order to refute it, it is sufficient to look for sets of (infinite) sequences with cardinality 2.

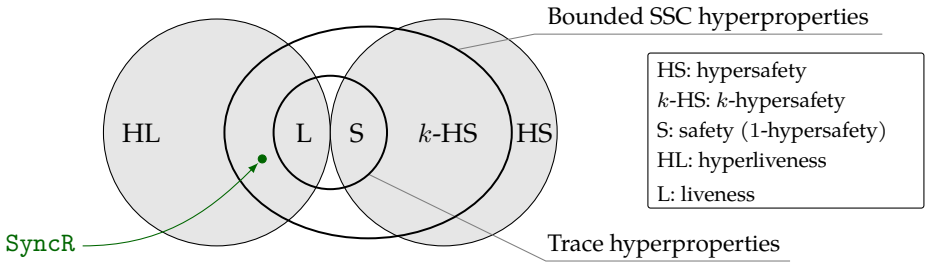


Figure 7.1: Bounded subset-closed hyperproperties.

Example 15 proves that there are hyperproperties which are not k-hypersafety but are k-bounded subset-closed (other than the trivial liveness properties). In Fig. 7.1 we have a graphical representation of how we can classify hyperproperties, w.r.t. the safety/liveness dichotomy and subset-closure.

Remark. Bounded hyperproperties give a characterization of some useful hyperproperties. Non-Interference is widely characterized (most of the times) as a 2-hypersafety, but it is not the case for the classic definition of NI, as defined in Example 14. Indeed, when execution denotations are just (I/O) pairs, we cannot introduce the concept of history of the computation and hence of "safety": it does not make sense to define prefixes of pairs (they are just the first elements or the whole pair). Nevertheless, we can define the subset-closure, and say that NI is 2-bounded.

7.2 Verification of Bounded Subset-Closed Hyperproperties

We have introduced so far k-bounded subset-closed hyperproperties, namely hyperproperties $\mathfrak{chp}$ such that, in order to prove that $X \notin \mathfrak{chp}$ we just need to exhibit a counterexample $T_k \subseteq X$ consisting in k elements. Formally, $X \in \mathfrak{chp} \Rightarrow \forall Y \subseteq X \,.\, Y \in \mathfrak{chp}$ and $X \notin \mathfrak{hp} \Leftrightarrow (\exists T_k \subseteq X \,.\, (|T_k| \leq k \wedge T_k \notin \mathfrak{chp}))$. In this section, for simplicity of exposition, we denote with $\mathcal{S}$ the base semantics of a given system. For instance, with $\mathcal{S}$ we could mean the base semantics $\mathcal{S}^{\mathsf{P}}_{\text{base}}$ of a given program P, for a given execution denotations domain $\mathbb{Den}$. In this setting, a system, with base semantics $\mathcal{S} \in \wp(\mathbb{Den})$, satisfies a k-bounded subset-closed hyperproperty $\mathfrak{chp}$ if and only if $\mathcal{S}^{|k} \triangleq \{X \subseteq \mathcal{S} \mid |X| = k\} \subseteq \mathfrak{chp}$ (Proposition 9). In the following we explain how it is possible to simplify the verification process even further, for particular bounded hyperproperties. Hence we have that $\mathcal{S}^{|k}$ is an equisatisfying hypersemantics w.r.t. $\mathtt{SSC}^{\mathtt{H}}_k$: $\{\mathcal{S}\} \not\subseteq \mathcal{S}^{|k}$, but for every $\mathfrak{chp} \in \mathtt{SSC}^{\mathtt{H}}_k$ we have that $\mathsf{P} \models \mathfrak{chp}$ if and only if $\mathcal{S}^{|k} \subseteq \mathfrak{chp}$.

7.2.1 Partitions-Driven Verification

We can observe that for particular bounded hyperproperties the verification can be split into several parts, simplifying the process. For a given bounded hyperproperty $\mathfrak{chp} \in \mathtt{SSC}^{\mathtt{H}}_k$, let $\mathfrak{chp}^{|k}$ be the set $\{X \in \mathfrak{chp} \mid |X| = k\}$. Then we have that the semantics $\mathcal{S}$ satisfies $\mathfrak{chp}$ if and only if $\mathcal{S}^{|k} \subseteq \mathfrak{chp}^{|k}$. A *finite* partition $\mathbb{P}$ of $\mathfrak{chp}^{|k}$ is a non-empty subset of $\wp(\mathfrak{chp}^{|k})$ such that:

$$|\mathbb{P}| \in \mathbb{N} \qquad \text{and} \qquad \bigcup \mathbb{P} = \mathfrak{chp}^{|k} \qquad \text{and} \qquad \forall \mathcal{X}, \mathcal{Y} \in \mathbb{P} \,.\, \mathcal{X} \cap \mathcal{Y} = \varnothing$$

Suppose to have a family of *partitioning functions* $\{f_i\}_{i \in \Delta}$ for $\mathfrak{chp}^{|k}$, indexed, by an arbitrary finite set Δ, namely $f_i \in \wp(\wp(\mathbb{Den})) \to \wp(\wp(\mathbb{Den}))$, for every $i \in \Delta$, and $\{f_i(\mathfrak{chp}^{|k}) \mid i \in \Delta\}$ is a finite partition[2] of $\mathfrak{chp}^{|k}$. Note that the functions f_i can be freely applied to $\mathcal{S}^{|k}$, but applying $\{f_i\}_{i \in \Delta}$ to $\mathcal{S}^{|k}$, in general, does not result in a partition of $\mathcal{S}^{|k}$. Then we can divide the verification process w.r.t. the partitions generated by $\{f_i\}_{i \in \Delta}$, as stated by the following definition.

Definition 30 (Partitionable Hyperproperty)**.** Given a k-bounded subset-closed hyperproperty $\mathfrak{chp}$, we say that $\mathfrak{chp}$ is *partitionable* if there exists a family, indexed by a finite set Δ, of partitioning functions $\{f_i\}_{i \in \Delta}$ such that for every $\mathcal{S} \in \wp(\mathbb{Den})$ we have that:

$$\mathcal{S}^{|k} \subseteq \mathfrak{chp}^{|k} \Leftrightarrow \forall i \in \Delta \,.\, f_i(\mathcal{S}^{|k}) \subseteq f_i(\mathfrak{chp}^{|k})$$

The definition says that for some bounded hyperproperties we can make the verification on a set of simpler hyperproperties, as we can see in the next example.

Example 16. Non-Interference, as introduced in Example 14, is a partitionable 2-bounded hyperproperty. Given the set $\{\mathsf{L}, \neg\mathsf{L}\}$, the family of partitioning functions is composed by:

$$f_{\mathsf{L}} \triangleq \lambda \mathcal{X} \,.\, \left\{ \{\langle \mathsf{m}_1, \mathsf{m}_1' \rangle, \langle \mathsf{m}_2, \mathsf{m}_2' \rangle\} \in \mathcal{X} \;\middle|\; \mathsf{m}_1 \overset{\mathsf{L}}{=} \mathsf{m}_2 \right\}$$
$$f_{\neg\mathsf{L}} \triangleq \lambda \mathcal{X} \,.\, \left\{ \{\langle \mathsf{m}_1, \mathsf{m}_1' \rangle, \langle \mathsf{m}_2, \mathsf{m}_2' \rangle\} \in \mathcal{X} \;\middle|\; \mathsf{m}_1 \overset{\mathsf{L}}{\neq} \mathsf{m}_2 \right\}$$

We have that $\{f_{\mathsf{L}}(\mathsf{NI}^{|2}), f_{\neg\mathsf{L}}(\mathsf{NI}^{|2})\}$ is clearly a partition of $\mathsf{NI}^{|2}$. Then $\mathcal{S} \in \wp(\mathbb{Den})$, where $\mathbb{Den} = \mathsf{Mem} \times \mathsf{Mem}$, satisfies NI if and only if :

$$\begin{aligned} f_{\mathsf{L}}(\mathcal{S}^{|2}) &= \{\{\langle \mathsf{m}_1, \mathsf{m}_1' \rangle, \langle \mathsf{m}_2, \mathsf{m}_2' \rangle\} \subseteq \mathcal{S} \mid \mathsf{m}_1 \overset{\mathsf{L}}{=} \mathsf{m}_2\} \\ &\subseteq \\ f_{\mathsf{L}}(\mathsf{NI}^{|2}) &= \{\{\langle \mathsf{m}_1, \mathsf{m}_1' \rangle, \langle \mathsf{m}_2, \mathsf{m}_2' \rangle\} \subseteq \mathbb{Den} \mid \mathsf{m}_1 \overset{\mathsf{L}}{=} \mathsf{m}_2 \wedge \mathsf{m}_1' \overset{\mathsf{L}}{=} \mathsf{m}_2'\} \end{aligned}$$

and

$$\begin{aligned} f_{\neg\mathsf{L}}(\mathcal{S}^{|2}) &= \{\{\langle \mathsf{m}_1, \mathsf{m}_1' \rangle, \langle \mathsf{m}_2, \mathsf{m}_2' \rangle\} \subseteq \mathcal{S} \mid \mathsf{m}_1 \overset{\mathsf{L}}{\neq} \mathsf{m}_2\} \\ &\subseteq \\ f_{\neg\mathsf{L}}(\mathsf{NI}^{|2}) &= \{\{\langle \mathsf{m}_1, \mathsf{m}_1' \rangle, \langle \mathsf{m}_2, \mathsf{m}_2' \rangle \subseteq \mathbb{Den}\} \mid \mathsf{m}_1 \overset{\mathsf{L}}{\neq} \mathsf{m}_2\} \end{aligned}$$

Due to the definition of Non-Interference, $f_{\mathsf{L}}(\mathsf{NI}^{|2})$ contains only sets $\{\langle \mathsf{m}_1, \mathsf{m}_1' \rangle, \langle \mathsf{m}_2, \mathsf{m}_2' \rangle\}$ such that $\mathsf{m}_1' \overset{\mathsf{L}}{=} \mathsf{m}_2'$, so we have to check if $f_{\mathsf{L}}(\mathcal{S}^{|2})$ is contained in $\{\{\langle \mathsf{m}_1, \mathsf{m}_1' \rangle, \langle \mathsf{m}_2, \mathsf{m}_2' \rangle\} \subseteq \mathbb{Den} \mid \mathsf{m}_1 \overset{\mathsf{L}}{=} \mathsf{m}_2 \wedge \mathsf{m}_1' \overset{\mathsf{L}}{=} \mathsf{m}_2'\}$. Furthermore, again due to the definition of Non-Interference, the check $f_{\neg\mathsf{L}}(\mathcal{S}^{|2}) \subseteq f_{\neg\mathsf{L}}(\mathsf{NI}^{|2})$ is not necessary, since the inclusion always holds.

[2] Usually a partition of X is given by means of a partitioning function $f \in X \to \wp(X)$, but it can be equivalently defined by a family of functions $\{f_i \in \wp(X) \to \wp(X)\}$.

7.2.2 Not-Relational Verification

Suppose now that systems execution denotations have the form of pairs, namely we are interested in a relation between the inputs and the outputs of a system (this is the case for Non-Interference). In this setting, given an element $\bar{\sigma} \in \mathbb{D}\mathbb{e}\mathbb{n} \triangleq \Sigma \times \Sigma$ (Σ arbitrary set representing system state denotations) we denote with $\bar{\sigma}_{\vdash}$ its projection on the first element of the pair and with $\bar{\sigma}_{\dashv}$ its projection on the second element. Hyperproperties are in $\wp(\wp(\Sigma \times \Sigma))$ and it is natural to wonder whether we can build a verification mechanism on simpler domains as $\wp(\wp(\Sigma) \times \wp(\Sigma))$ or $\wp(\wp(\Sigma)) \times \wp(\wp(\Sigma))$. Hence, consider the abstraction $\alpha_{st} \in \wp(\wp(\Sigma \times \Sigma)) \to \wp(\wp(\Sigma) \times \wp(\Sigma))$ defined as $\alpha_{st} \triangleq \lambda\mathcal{X} \,.\, \big\{\langle\{\bar{\sigma}_{\vdash} \mid \bar{\sigma} \in X\}, \{\bar{\sigma}_{\dashv} \mid \bar{\sigma} \in X\}\rangle \;\big|\; X \in \mathcal{X}\big\}$, and the further abstraction $\alpha_{\widetilde{st}} \in \wp(\wp(\Sigma) \times \wp(\Sigma)) \to \wp(\wp(\Sigma)) \times \wp(\wp(\Sigma))$, defined as $\alpha_{\widetilde{st}} \triangleq \lambda X \,.\, \langle\{A \mid \langle A, B\rangle \in X\}, \{B \mid \langle A, B\rangle \in X\}\rangle$.

Example 17. Suppose again that $\mathbb{D}\mathbb{e}\mathbb{n}$ is the domain $\mathsf{Mem} \times \mathsf{Mem}$ of input/output memories, and consider the set $\mathcal{X} = \{\{\langle \mathsf{m}_a, \mathsf{m}_b\rangle, \langle \mathsf{m}_a, \mathsf{m}_c\rangle\}, \{\langle \mathsf{m}_a, \mathsf{m}_d\rangle, \langle \mathsf{m}_d, \mathsf{m}_e\rangle\}\} \in \wp(\wp(\mathbb{D}\mathbb{e}\mathbb{n}))$. Then $\alpha_{st}(\mathcal{X})$ is the set $X = \{\langle\{\mathsf{m}_a\}, \{\mathsf{m}_b, \mathsf{m}_c\}\rangle, \langle\{\mathsf{m}_a, \mathsf{m}_d\}, \{\mathsf{m}_d, \mathsf{m}_e\}\rangle\}$ and $\alpha_{\widetilde{st}}(X)$ is the pair $\langle\{\{\mathsf{m}_a\}, \{\mathsf{m}_a, \mathsf{m}_d\}\}, \{\{\mathsf{m}_b, \mathsf{m}_c\}, \{\mathsf{m}_d, \mathsf{m}_e\}\}\rangle$.

By using α_{st} and $\alpha_{nd} \triangleq \alpha_{\widetilde{st}} \circ \alpha_{st}$, which both trivially form a Galois connection with $\wp(\wp(\mathbb{D}\mathbb{e}\mathbb{n}))$, we can define the following particular hyperproperties.

Definition 31 (Not-Relational Hyperproperty)**.** A k-bounded subset-closed hyperproperty $\mathfrak{chyp}$ is *first order not-relational* when for each $\mathcal{S} \in \wp(\mathbb{D}\mathbb{e}\mathbb{n})$:

$$\mathcal{S}^{|k} \subseteq \mathfrak{chyp}^{|k} \Leftrightarrow \alpha_{st}(\mathcal{S}^{|k}) \subseteq \alpha_{st}(\mathfrak{chyp}^{|k})$$

and it is *second order not-relational* when for each $\mathcal{S} \in \wp(\mathbb{D}\mathbb{e}\mathbb{n})$:

$$\mathcal{S}^{|k} \subseteq \mathfrak{chyp}^{|k} \Leftrightarrow \alpha_{nd}(\mathcal{S}^{|k}) \subseteq^2 \alpha_{nd}(\mathfrak{chyp}^{|k})$$

Here $\subseteq^2$ is the product (or component-wise) order $\{\langle\langle\mathcal{X}, \mathcal{Y}\rangle, \langle\mathcal{X}', \mathcal{Y}'\rangle\rangle \mid \mathcal{X} \subseteq \mathcal{X}' \wedge \mathcal{Y} \subseteq \mathcal{Y}'\}$. A hyperproperty which is not first order nor second order not-relational, is called *relational*. A relational hyperproperty expresses a relation between input and output elements of traces (Fig. 7.2 green solid lines on the left). Instead, a first order not-relational hyperproperty expresses a relation between sets of inputs and sets of outputs elements (Fig. 7.2 green solid lines on the center). Finally, a second order not-relational hyperproperty expresses a relation between sets of sets of input and sets of sets of output elements (Fig. 7.2 green solid line on the right). The dotted lines in Fig. 7.2 are examples of spurious traces added by approximating a relational hyperproperty with α_{st} (on the center) and α_{nd} (on the right). The definition gives us other ways for simplifying the verification process for some particular hyperproperties, as we can see in the following example.

Example 18. Non-Interference, as introduced in Example 14, is a first order not-relational 2-bounded hyperproperty. For every set $\{\langle \mathsf{m}_1, \mathsf{m}'_1\rangle, \langle \mathsf{m}_2, \mathsf{m}'_2\rangle\} \in \mathcal{S}^{|2}$ we need to check only if $\mathsf{m}_1 \overset{\text{L}}{=} \mathsf{m}_2$ *and* $\mathsf{m}'_1 \overset{\text{L}}{=} \mathsf{m}'_2$: we do not need to know if m'_1 results from the computation started in m_1. Non-Interference is not second order, in fact knowing that $\mathsf{m}'_1 \overset{\text{L}}{\neq} \mathsf{m}'_2$ is not sufficient for verification, since this set could be obtained by a set such that $\mathsf{m}_1 \overset{\text{L}}{=} \mathsf{m}_2$.

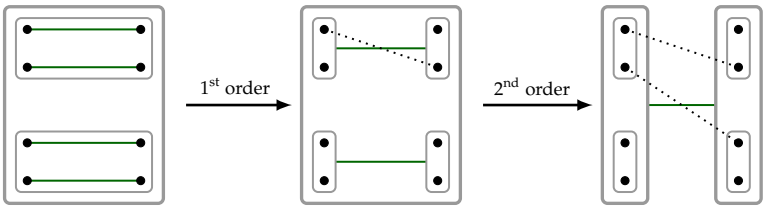


Figure 7.2: Relational and not-relational hyperproperties.

The ideal case is the one of second order not-relational hyperproperties: these latter can be verified computing on $\wp(\wp(\Sigma))$. Fortunately, even if an hyperproperty is not second order but just first order, when it is partitionable then we can still use a verification mechanism on $\wp(\wp(\Sigma))$. Indeed we have the following proposition.

Proposition 10. *If a k-bounded hyperproperty $\mathfrak{chp}$ is partitionable, by the partitioning functions $\{f_i\}_{i\in\Delta}$, and it is first order not-relational, then there exists a family of second order not-relational hyperproperties $\{\mathfrak{chp}_i\}_{i\in\Delta}$ such that for every $\mathcal{S} \in \wp(\mathbb{Dem})$ we have:*

$$\mathcal{S}^{|k} \in \mathfrak{chp}^{|k} \Leftrightarrow \forall i \in \Delta \,.\, \alpha_{nd}(f_i(\mathcal{S}^{|k})) \subseteq^2 \alpha_{nd}(f_i(\mathfrak{chp}_i^{|k}))$$

Again we use Non-Interference as a running example, in order to show the application of the proposition.

Example 19. We can partition Non-Interference, as introduced in Example 14, in two second order not-relational hyperproperties:

$$\mathsf{NI}^{|^2}_{\mathsf{L}} \triangleq \{X \subseteq \mathsf{Mem}\times\mathsf{Mem} \mid X = \{\langle \mathsf{m}_1, \mathsf{m}'_1\rangle, \langle \mathsf{m}_2, \mathsf{m}'_2\rangle\} \wedge \mathsf{m}_1 \overset{\mathsf{L}}{=} \mathsf{m}_2 \wedge \mathsf{m}'_1 \overset{\mathsf{L}}{=} \mathsf{m}'_2\}$$
$$\mathsf{NI}^{|^2}_{\neg\mathsf{L}} \triangleq \{X \subseteq \mathsf{Mem}\times\mathsf{Mem} \mid X = \{\langle \mathsf{m}_1, \mathsf{m}'_1\rangle, \langle \mathsf{m}_2, \mathsf{m}'_2\rangle\} \wedge \mathsf{m}_1 \overset{\mathsf{L}}{\neq} \mathsf{m}_2\}$$

It is easy to see that both $\mathsf{NI}^{|^2}_{\mathsf{L}}$ and $\mathsf{NI}^{|^2}_{\neg\mathsf{L}}$ are second order not-relational hyperproperties. Furthermore, consider the partitioning functions of Example 16, we have that $\mathcal{S}$ satisfies NI if and only if $\alpha_{nd}(f_{\mathsf{L}}(\mathcal{S}^{|^2})) \subseteq^2 \alpha_{nd}(f_{\mathsf{L}}(\mathsf{NI}^{|^2}_{\mathsf{L}}))$ and $\alpha_{nd}(f_{\neg\mathsf{L}}(\mathcal{S}^{|^2})) \subseteq^2 \alpha_{nd}(f_{\neg\mathsf{L}}(\mathsf{NI}^{|^2}_{\neg\mathsf{L}}))$. Since $f_{\neg\mathsf{L}}(\mathsf{NI}^{|^2}_{\neg\mathsf{L}})$ is always satisfied, we can skip the second verification check. Hence we have simplified a lot the verification process.

7.2.3 Example: Verifying Abstract Non-Interference

In this subsection we show how the verification for Abstract Non-Interference, can be made simpler, applying the results of the previous subsections. Non-Interference requires that any change of private data should not be revealed through the observation of the public one, but any real system are intended to leak some kind of information. Hence, a weakening of Non-Interference is necessary.

Among all formal methods for weakening Non-Interference, we adopt the one proposed in [Giacobazzi and Mastroeni, 2004], which is based on abstract interpretation. Abstract Non-Interference considers dependencies between *properties* of values. In particular, it allows some (property of) the confidential information to flow and it considers weaker attackers, i.e. attackers with a restricted observation power of public data. This naturally

deals with declassification: sometimes it is necessary to release some confidential information in order to make a system useful (selective dependencies of [Cohen, 1977]). Hence, Abstract Non-Interference makes Non-Interference parametric on two properties, each one modeling different aspects of the information flow: what the attacker can observe and what information is allowed/not allowed to flow. In general, we can suppose that the attacker may not have the same constraints in observing inputs and outputs but, in the following, we assume that the attacker's observation precision is the same, both in input and in output.

The attacker is characterized as a property ρ representing what can be observed about the public input/output of programs. An abstraction is a property of data, hence an attacker can distinguish data up to particular properties (for instance, an attacker could be able to distinguish the parity of variables but not their sign). As far as declassification is concerned, we suppose to specify what private information is allowed to flow. Let ϕ be the input property representing the input property that may flow in output, without violating the information flow policy: we check (abstract) non-interference only for those private inputs agreeing on the property ϕ. This models the information that may flow, since it is not interesting to check whether its variation is visible through the output.

Formally, variables take values in $\mathbb{Z}$ hence data properties are modeled as upper closure operators on the complete lattice $\langle \wp(\mathbb{Z}), \subseteq, \cup, \cap, \varnothing, \mathbb{Z} \rangle$ [Giacobazzi and Mastroeni, 2018]. We denote with $\mathsf{uco}(\wp(\mathbb{Z}))$ the set of all upper closure operators on $\langle \wp(\mathbb{Z}), \subseteq, \cup, \cap, \varnothing, \mathbb{Z} \rangle$ and we use, from now on, Greek lowercase letters in order to denote its elements. Let $\iota \triangleq \lambda X \,.\, X$ and $\tau \triangleq \lambda X \,.\, \mathbb{Z}$ be the bottom closure (representing the most concrete property) and the top closure (representing the most abstract property) on $\wp(\mathbb{Z})$, respectively. Now we introduce a notation allowing us to compactly compare memories, w.r.t. an abstraction ρ on L variables and an abstraction ϕ on H variables. Given a typing environment $\Gamma \in \mathsf{Var} \to \{\mathsf{L}, \mathsf{H}\}$ and two upper closure operators $\rho, \phi \in \mathsf{uco}(\wp(\mathbb{Z}))$, we define[3] $\nu^{\rho \times \phi} \in \mathsf{Mem} \to (\mathsf{Var} \to \wp(\mathbb{Z}))$ as:

$$\nu^{\rho \times \phi}(\mathsf{m}) \triangleq \lambda \mathsf{x} \,.\, \begin{cases} \rho(\{\mathsf{m}(\mathsf{x})\}) & \text{if } \Gamma(\mathsf{x}) = \mathsf{L} \\ \phi(\{\mathsf{m}(\mathsf{x})\}) & \text{if } \Gamma(\mathsf{x}) = \mathsf{H} \end{cases}$$

Hence $\nu^{\rho \times \phi}(\mathsf{m}) = \nu^{\rho \times \phi}(\mathsf{m}')$ denotes the fact that memories m, m' agree on public variables, up to the abstraction ρ (denoting that the attacker is observing the same public input property), and agree on private variables, up to the abstraction ϕ (denoting that the variations indistinguishable by ϕ may be revealed).

Using this notation, we can elegantly generalize the definition of Non-Interference w.r.t. a property $\phi \in \mathsf{uco}(\wp(\mathbb{Z}))$ of input private variables which may flow and an observable property $\rho \in \mathsf{uco}(\wp(\mathbb{Z}))$ of input/output public variables. A program is (abstract) non-interferent if and only if whenever it starts its computation from memories $\mathsf{m}_1, \mathsf{m}_2$, such that $\nu^{\rho \times \phi}(\mathsf{m}_1) = \nu^{\rho \times \phi}(\mathsf{m}_2)$, then it ends its computation in memories $\mathsf{m}'_1, \mathsf{m}'_2$ such that $\nu^{\rho \times \tau}(\mathsf{m}'_1) = \nu^{\rho \times \tau}(\mathsf{m}'_2)$. This hyperproperty is defined as:

$$\mathsf{ANI}^{\rho}_{\phi} \triangleq \{X \in \wp(\mathbb{D}\mathbb{e}\mathbb{n}) \mid \forall \bar{\sigma}, \bar{\sigma}' \in X \,.\, \nu^{\rho \times \phi}(\bar{\sigma}_{\vdash}) = \nu^{\rho \times \phi}(\bar{\sigma}'_{\vdash}) \Rightarrow \nu^{\rho \times \tau}(\bar{\sigma}_{\dashv}) = \nu^{\rho \times \tau}(\bar{\sigma}'_{\dashv})\}$$

Proposition 11. *For any $\rho, \phi \in \mathsf{uco}(\wp(\mathbb{Z}))$, we have that $\mathsf{ANI}^{\rho}_{\phi}$ is a partitionable first order not-relational 2-bounded subset-closed hyperproperty.*

[3] We take in consideration the two-levels security lattice for classic Non-Interference, but it is straightforward to generalize the definition for arbitrary multi-levels lattices.

Proof. The partitioning functions, indexed by $\Delta \triangleq \{=, \neq\}$, for $\mathsf{ANI}^{\rho}_{\phi}$, ANI in short, are:

$$f_{=} \triangleq \lambda\mathcal{X} \,.\, \left\{ \{\langle \mathsf{m}_1, \mathsf{m}'_1\rangle, \langle \mathsf{m}_2, \mathsf{m}'_2\rangle\} \in \mathcal{X} \;\middle|\; \nu^{\rho\times\phi}(\mathsf{m}_1) = \nu^{\rho\times\phi}(\mathsf{m}_2) \right\}$$
$$f_{\neq} \triangleq \lambda\mathcal{X} \,.\, \left\{ \{\langle \mathsf{m}_1, \mathsf{m}'_1\rangle, \langle \mathsf{m}_2, \mathsf{m}'_2\rangle\} \in \mathcal{X} \;\middle|\; \nu^{\rho\times\phi}(\mathsf{m}_1) \neq \nu^{\rho\times\phi}(\mathsf{m}_2) \right\}$$

Clearly, $\{f_{=}(\mathsf{ANI}), f_{\neq}(\mathsf{ANI})\}$ is a partition of $\mathsf{ANI}^{|^2}$. Furthermore, we have that $\alpha_{st}(\mathsf{ANI}^{|^2})$ is:

$$\left\{ \langle \{\mathsf{m}_1, \mathsf{m}_2 \mid \nu^{\rho\times\phi}(\mathsf{m}_1) = \nu^{\rho\times\phi}(\mathsf{m}_2)\}, \{\mathsf{m}'_1, \mathsf{m}'_2 \mid \nu^{\rho\times\phi}(\mathsf{m}'_1) = \nu^{\rho\times\phi}(\mathsf{m}'_2)\}\rangle \;\middle|\; \langle \mathsf{m}_1, \mathsf{m}'_1\rangle, \langle \mathsf{m}_2, \mathsf{m}'_2\rangle \in \mathbb{D}\mathbb{e}\mathbb{n} \right\}$$
$$\cup \left\{ \langle \{\mathsf{m}_1, \mathsf{m}_2 \mid \nu^{\rho\times\phi}(\mathsf{m}_1) \neq \nu^{\rho\times\phi}(\mathsf{m}_2)\}, \{\mathsf{m}'_1, \mathsf{m}'_2\}\rangle \;\middle|\; \langle \mathsf{m}_1, \mathsf{m}'_1\rangle, \langle \mathsf{m}_2, \mathsf{m}'_2\rangle \in \mathbb{D}\mathbb{e}\mathbb{n} \right\}$$

It is clear that a system $\mathcal{S}$ satisfies ANI if and only if $\alpha_{st}(\mathcal{S}^{|^2}) \subseteq \alpha_{st}(\mathsf{ANI}^{|^2})$. In fact, this later inclusion means that for every pair of executions of $\mathcal{S}$, either the executions start in memories equivalent, modulo $\langle \phi, \rho\rangle$, and end in memories equivalent modulo $\langle \phi, \rho\rangle$ or the executions start in memories not equivalent (which is the definition of ANI). □

Due to Propositions 11 and 10, there exists two second order not-relational hyperproperties $\mathsf{ANI}^{|^2}_{=}$ and $\mathsf{ANI}^{|^2}_{\neq}$ such that $\mathcal{S} \in \mathsf{ANI}$ if and only if $\alpha_{nd}(f_{=}(\mathcal{S}^{|^2})) \subseteq^2 \alpha_{nd}(f_{=}(\mathsf{ANI}^{|^2}_{=}))$ and $\alpha_{nd}(f_{\neq}(\mathcal{S}^{|^2})) \subseteq^2 \alpha_{nd}(f_{\neq}(\mathsf{ANI}^{|^2}_{\neq}))$. These two hyperproperties are:

$$\mathsf{ANI}^{|^2}_{=} \triangleq \left\{ X \subseteq \mathsf{Mem}\times\mathsf{Mem} \;\middle|\; \begin{array}{l} X = \{\langle \mathsf{m}_1, \mathsf{m}'_1\rangle, \langle \mathsf{m}_2, \mathsf{m}'_2\rangle\} \wedge \\ \nu^{\rho\times\phi}(\mathsf{m}_1) = \nu^{\rho\times\phi}(\mathsf{m}_2) \wedge \nu^{\rho\times\tau}(\mathsf{m}'_1) = \nu^{\rho\times\tau}(\mathsf{m}'_2) \end{array} \right\}$$
$$\mathsf{ANI}^{|^2}_{\neq} \triangleq \left\{ X \subseteq \mathsf{Mem}\times\mathsf{Mem} \;\middle|\; X = \{\langle \mathsf{m}_1, \mathsf{m}'_1\rangle, \langle \mathsf{m}_2, \mathsf{m}'_2\rangle\} \wedge \nu^{\rho\times\phi}(\mathsf{m}_1) \neq \nu^{\rho\times\phi}(\mathsf{m}_2) \right\}$$

The verification is then reduced to the check $\alpha_{nd}(f_{=}(\mathcal{S}^{|^2})) \subseteq^2 \alpha_{nd}(f_{=}(\mathsf{ANI}^{|^2}_{=}))$ since, as expected, $\alpha_{nd}(f_{\neq}(\mathcal{S}^{|^2})) \subseteq^2 \alpha_{nd}(f_{\neq}(\mathsf{ANI}^{|^2}_{\neq}))$ is, by definition, always true.

Now we can show how to exploit this fact in order to verify $\mathsf{ANI}^{\rho}_{\phi}$. Given the base semantics $\mathcal{S} \in \wp(\mathbb{D}\mathbb{e}\mathbb{n})$ of a program, for the set of execution denotations domain $\mathbb{D}\mathbb{e}\mathbb{n} \triangleq \mathsf{Mem} \times \mathsf{Mem}$, its collecting semantics, i.e. it strongest hyperproperty, for $\mathbb{D}\mathbb{e}\mathbb{n}$ is $\{\mathcal{S}\}$. We have that the program satisfies ANI if and only if $\mathcal{S} \in \mathsf{ANI}$ or, equivalently, if and only if $\{\mathcal{S}\} \subseteq \mathsf{ANI}$. This boils down to check whether $\alpha_{nd}(f_{=}(\mathcal{S}^{|^2})) \subseteq^2 \alpha_{nd}(f_{=}(\mathsf{ANI}^{|^2}_{=}))$ holds or not. Assume that $\alpha_{nd}(f_{=}(\mathcal{S}^{|^2}))$ is the pair $\langle \mathcal{X}, \mathcal{Y}\rangle$ and $\alpha_{nd}(f_{=}(\mathsf{ANI}^{|^2}_{=}))$ is the pair $\langle \mathcal{X}', \mathcal{Y}'\rangle$, then we have, by definition, $\mathcal{X} \subseteq \mathcal{X}'$. This means that the verification process for Abstract Non-Interference is reduced to the check $\mathcal{Y} \subseteq \mathcal{Y}'$, namely to check whether $\mathcal{Y}$ contains only sets of memories which agree on L variables, modulo the abstraction ρ. So, basically, we can verify Abstract Non-Interference just checking whether $\alpha_{nd}(f_{=}(\mathcal{S}^{|^2}))_{\dashv}$ satisfies a hyperproperty on the set of execution denotations Mem, as stated by the following proposition.

Proposition 12. *A program, with base semantics $\mathcal{S}$, satisfies $\mathsf{ANI}^{\rho}_{\phi} \in \wp(\wp(\mathsf{Mem} \times \mathsf{Mem}))$ if and only if $\alpha_{nd}(f_{=}(\mathcal{S}^{|^2}))_{\dashv} \subseteq \mathtt{equiv}^{\rho}_{\mathsf{L}}$, where $\mathtt{equiv}^{\rho}_{\mathsf{L}} \in \wp(\wp(\mathsf{Mem}))$ is:*

$$\mathtt{equiv}^{\rho}_{\mathsf{L}} \triangleq \left\{ X \subseteq \mathsf{Mem} \;\middle|\; \forall \mathsf{m}, \mathsf{m}' \in X \,.\, \nu^{\rho\times\tau}(\mathsf{m}) = \nu^{\rho\times\tau}(\mathsf{m}') \right\}$$

This simplifies the verification process for Abstract Non-Interference: We can build a verification method computing on $\wp(\wp(\mathsf{Mem}))$ instead of $\wp(\wp(\mathsf{Mem} \times \mathsf{Mem}))$.

7.3 Hypersemantics for Subset-Closed Hyperproperties

As we have observed, in order to verify hyperproperties, we may need to move program semantics into the hyperlevel. Until now, we described the links between base and hypersemantics of a transition system. In this section, we show how to *lift* a given semantic operator, defined on a program P syntax, computing $\mathcal{S}^{\mathsf{P}}_{\text{base}}$. We lift this operator, defined on sets, in order to obtain a corresponding operator, defined on sets of sets, suitable for hyperproperties verification. In the following, we consider denotational semantics, as introduced in Chapter 5, as a base semantics, but the whole framework can be generalized to other types of semantics.

Consider, as a simple example, the post-conditions semantics $[\![\mathsf{P}]\!]^{\curvearrowright} \in \wp(\mathsf{Mem}) \to \wp(\mathsf{Mem})$, introduced in Subsection 5.2.1, of a given program in Imp. Recall that this semantics computes the strongest memory invariant on the last control point of P, obtained by the execution of the program starting in a given set of memories. Basically it computes a relation between input and output (memory) invariants of P.

In order to simplify the presentation, consider the operator $[\![\mathsf{b}]\!]^{\curvearrowright} \in \wp(\mathsf{Mem}) \to \wp(\mathsf{Mem})$ filtering memories, namely $[\![\mathsf{b}]\!]^{\curvearrowright} X \triangleq \{\mathsf{m} \in X \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\}$. Hence, we can express more compactly the post-conditions semantics as follows:

$$[\![\mathsf{P}]\!]^{\curvearrowright}\varnothing \triangleq \varnothing \qquad [\![{}^{\mathsf{i}}\mathsf{c}_1\,{}^{\mathsf{l}}\,.\,{}^{\mathsf{k}}\mathsf{c}_2\,{}^{\mathsf{f}}]\!]^{\curvearrowright} X \triangleq [\![{}^{\mathsf{k}}\mathsf{c}_2\,{}^{\mathsf{f}}]\!]^{\curvearrowright} \circ [\![{}^{\mathsf{i}}\mathsf{c}_1\,{}^{\mathsf{l}}]\!]^{\curvearrowright} X \qquad [\![{}^{\mathsf{i}}\mathsf{skip}\,{}^{\mathsf{f}}]\!]^{\curvearrowright} X \triangleq X$$

$$[\![{}^{\mathsf{i}}\mathsf{x} := \mathsf{a}\,{}^{\mathsf{f}}]\!]^{\curvearrowright} X \triangleq \{\mathsf{m}[\mathsf{x} \leftarrow n] \mid \mathsf{m} \in X \wedge \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\}$$

$$[\![{}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}\,{}^{\mathsf{f}}]\!]^{\curvearrowright} X \triangleq [\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright} X$$

$$[\![{}^{\mathsf{i}}\mathsf{while}\ {}^{\mathsf{l}}\mathsf{b}\ \{\,\mathsf{P}\,\}\,{}^{\mathsf{f}}]\!]^{\curvearrowright} X \triangleq [\![\neg\mathsf{b}]\!]^{\curvearrowright}\big(\mathsf{lfp}^{\subseteq}_{\varnothing}\,\mathsf{F}^{\curvearrowright}\big) \quad \text{where } \mathsf{F}^{\curvearrowright}(T) \triangleq X \cup [\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} T$$

At this point, we have to move base semantics towards sets of sets, namely on $\wp(\wp(\mathsf{Mem}))$, namely, we need to define a semantic operator $(\!|\mathsf{P}|\!)^{\curvearrowright} \in \wp(\wp(\mathsf{Mem})) \to \wp(\wp(\mathsf{Mem}))$ computing it. In this case, we have to lift the base semantic operator, as explained in the previous chapter. Problems arise mainly when we have a fixpoint computation, since for not-recursive statements operators the direct image lift suffices. Suppose then to have a filtering function $(\!|\mathsf{b}|\!)^{\curvearrowright} \in \wp(\wp(\mathsf{Mem})) \to \wp(\wp(\mathsf{Mem}))$ for boolean expressions, defined as $(\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{X} \triangleq \{[\![\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\} \setminus \{\varnothing\}$[4]. The definition of the hypersemantics is just the direct image lift (to sets of sets) of $[\![\mathsf{P}]\!]^{\curvearrowright}$ for every statement, except for the while case (and for sequencing, which is by functions composition). Indeed, we can observe that, at hyperlevel, the semantic operator $(\!|\mathsf{P}|\!)^{\curvearrowright}$ for the while statements does not coincide with the direct image lift of $[\![\mathsf{P}]\!]^{\curvearrowright}$, which would be $(\!|{}^{\mathsf{i}}\mathsf{while}\ {}^{\mathsf{k}}\mathsf{b}\ \{\,\mathsf{P}\,\}\,{}^{\mathsf{f}}|\!)^{\curvearrowright}\mathcal{X} \triangleq (\!|\neg\mathsf{b}|\!)^{\curvearrowright}(\mathsf{lfp}^{\subseteq}_{\varnothing}\,\mathsf{H}^{\curvearrowright})$ with $\mathsf{H}^{\curvearrowright} \triangleq \lambda\mathcal{T}\,.\,\mathcal{X} \cup (\!|\mathsf{P}|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{T}$. Since in the while statement we have a fixpoint computation, this latter solution does not lead to a correct hypersemantics: i.e. $\{[\![\mathsf{P}]\!]^{\curvearrowright} I\} \not\subseteq (\!|\mathsf{P}|\!)^{\curvearrowright}\{I\}$.

Example 20. Let $\mathsf{P} = {}^{0}\mathsf{while}\ {}^{1}(\mathsf{x} < 4)\ \{\ {}^{2}\mathsf{x} := \mathsf{x} + 1\,{}^{3}\ \}\,{}^{4}$. Since P has only one variable, we denote $[\mathsf{x} \mapsto \mathsf{v}]$ just by v and the set of functions $\{[\mathsf{x} \mapsto \mathsf{v}_1], \ldots [\mathsf{x} \mapsto \mathsf{v}_n]\}$ by $\{\mathsf{v}_1, \ldots \mathsf{v}_n\}$. The base semantics, from $I \triangleq \{2, 5\}$, is $[\![\mathsf{P}]\!]^{\curvearrowright}\{2, 5\} = \{4, 5\}$, computed as $\{2, 5\} \xrightarrow{\mathsf{F}^{\curvearrowright}} \{4, 5\}$

[4] We need to remove $\varnothing$ for technical reasons. This is also coherent with our initial choice to the define $\varnothing$ instead of $\{\varnothing\}$ as the false hyperproperty.

where
$$\mathsf{F}^{\cap\!\!\uparrow 0} = \varnothing;\ \ \mathsf{F}^{\cap\!\!\uparrow 1} = \{2,5\};\ \ \mathsf{F}^{\cap\!\!\uparrow 2} = \{2,3,5\};\ \ \mathsf{F}^{\cap\!\!\uparrow 3} = \{2,3,4,5\}$$
The naive lift of the while base semantics would be $(\!|\mathsf{P}|\!)^{\cap\!\!\uparrow}\{\{2,5\}\} = \{\varnothing,\{4\},\{5\}\}$, computed as $\{\{2,5\}\} \xrightarrow{\mathsf{H}^{\cap\!\!\uparrow}} \{\varnothing,\{4\},\{5\}\}$ where

$$\mathsf{H}^{\cap\!\!\uparrow 0} = \varnothing \qquad \mathsf{H}^{\cap\!\!\uparrow 1} = \{\{2,5\}\} \qquad \mathsf{H}^{\cap\!\!\uparrow 2} = \{\{3\},\{2,5\}\} \qquad \mathsf{H}^{\cap\!\!\uparrow 3} = \{\{3\},\{4\},\{2,5\}\}$$
$$\mathsf{H}^{\cap\!\!\uparrow 4} = \{\varnothing,\{3\},\{4\},\{2,5\}\}$$

From the iterates of $\mathsf{F}^{\cap\!\!\uparrow}$ and $\mathsf{H}^{\cap\!\!\uparrow}$ we can observe the monotonicity (and the extensivity) of $\mathsf{F}^{\cap\!\!\uparrow}$ and $\mathsf{H}^{\cap\!\!\uparrow}$, but the hypersemantics is not correct, because $[\![\mathsf{P}]\!]^{\cap\!\!\uparrow}\{2,5\} = \{4,5\} \notin \{\varnothing,\{4\},\{5\}\} = (\!|\mathsf{P}|\!)^{\cap\!\!\uparrow}\{\{2,5\}\}$.

In order to lift the while semantics, we can use one of the solutions proposed in the previous chapter. For instance, changing the computational domain, we can obtain the generic hypersemantics operator for while statements:

$$(\!|\,{}^{\mathsf{i}}\mathsf{while}\,{}^{\mathsf{k}}\mathsf{b}\,\{\,\mathsf{P}\,\}^{\mathsf{f}}\,|\!)^{\cap\!\!\uparrow}\mathcal{X} \triangleq (\!|\neg\mathsf{b}|\!)^{\cap\!\!\uparrow}(\mathsf{lfp}^{\uplus}_{\{\varnothing\}}\,\mathsf{H}^{\cap\!\!\uparrow}_{\mathsf{G}}) \ \text{ where: } \ \mathsf{H}^{\cap\!\!\uparrow}_{\mathsf{G}}(\mathcal{T}) \triangleq \mathcal{X} \uplus \{[\![\mathsf{P}]\!]^{\cap\!\!\uparrow}[\![\mathsf{b}]\!]^{\cap\!\!\uparrow}T \mid T \in \mathcal{T}\}$$

The iterates of this operator, instantiated to Example 20, are:

$$\mathsf{F}^{\cap\!\!\uparrow 0}_{\mathsf{G}} = \{\varnothing\};\ \ \mathsf{F}^{\cap\!\!\uparrow 1}_{\mathsf{G}} = \{\{2,5\}\};\ \ \mathsf{F}^{\cap\!\!\uparrow 2}_{\mathsf{G}} = \{\{2,3,5\}\};\ \ \mathsf{F}^{\cap\!\!\uparrow 3}_{\mathsf{G}} = \{\{2,3,4,5\}\}$$

Then $\biguplus_{0\leq n\leq 3}\mathsf{F}^{\cap\!\!\uparrow n}_{\mathsf{G}}(\{\varnothing\}) = \{\{2,3,4,5\}\}$ and, finally, $(\!|4 \leq x|\!)^{\cap\!\!\uparrow}\{\{2,3,4,5\}\} = \{\{4,5\}\}$, which is exactly $\{[\![\mathsf{P}]\!]^{\cap\!\!\uparrow}\{2,5\}\}$. Of course, we can also consider the best correct approximation, obtaining the subset-closed hypersemantics operator for while statements:

$$(\!|\,{}^{\mathsf{i}}\mathsf{while}\,{}^{\mathsf{k}}\mathsf{b}\,\{\,\mathsf{P}\,\}^{\mathsf{f}}\,|\!)^{\cap\!\!\uparrow}\mathcal{X} \triangleq (\!|\neg\mathsf{b}|\!)^{\cap\!\!\uparrow}(\mathsf{lfp}^{\subseteq}_{\{\varnothing\}}\,\mathsf{H}^{\cap\!\!\uparrow}_{\mathsf{S}}) \ \text{ where: } \ \mathsf{H}^{\cap\!\!\uparrow}_{\mathsf{S}}(\mathcal{T}) \triangleq \wp(\bigcup\mathcal{X} \cup [\![\mathsf{P}]\!]^{\cap\!\!\uparrow}[\![\mathsf{b}]\!]^{\cap\!\!\uparrow}\bigcup\mathcal{T})$$

The iterates of this operator, instantiated to Example 20, are:

$$\mathsf{F}^{\cap\!\!\uparrow 0}_{\mathsf{S}} = \{\varnothing\};\ \ \mathsf{F}^{\cap\!\!\uparrow 1}_{\mathsf{S}} = \wp(\{2,5\});\ \ \mathsf{F}^{\cap\!\!\uparrow 2}_{\mathsf{S}} = \wp(\{2,3,5\});\ \ \mathsf{F}^{\cap\!\!\uparrow 3}_{\mathsf{S}} = \wp(\{2,3,4,5\})$$

Then $\bigcup_{0\leq n\leq 3}\mathsf{F}^{\cap\!\!\uparrow n}_{\mathsf{S}}(\{\varnothing\}) = \wp(\{2,3,4,5\})$ and, finally, $(\!|4 \leq x|\!)^{\cap\!\!\uparrow}\wp(\{2,3,4,5\}) = \wp(\{4,5\})$, which is exactly $\wp([\![\mathsf{P}]\!]^{\cap\!\!\uparrow}\{2,5\})$. Finally, without changing the computational order, we can define two hypersemantics operator for while statements which are in between the generic and the subset-closed versions, namely they strictly contain the first and they are strictly contained in the second. They are defined as $(\!|\,{}^{\mathsf{i}}\mathsf{while}\,{}^{\mathsf{k}}\mathsf{b}\,\{\,\mathsf{P}\,\}^{\mathsf{f}}\,|\!)^{\cap\!\!\uparrow}\mathcal{X} \triangleq (\!|\neg\mathsf{b}|\!)^{\cap\!\!\uparrow}(\mathsf{lfp}^{\subseteq}_{\varnothing}\,\mathsf{H}^{\cap\!\!\uparrow}_{\diamond})$, with $\diamond \in \{\mathrm{I},\mathrm{M}\}$, where:

Inner lift $\mathsf{H}^{\cap\!\!\uparrow}_{\mathrm{I}} \triangleq \lambda\mathcal{T}.\,\{\varnothing\} \cup (\mathcal{X} \uplus (\!|\mathsf{P}|\!)^{\cap\!\!\uparrow}(\!|\mathsf{b}|\!)^{\cap\!\!\uparrow}\mathcal{T})$

Mixed lift $\mathsf{H}^{\cap\!\!\uparrow}_{\mathrm{M}} \triangleq \lambda\mathcal{T}.\,\mathcal{X} \cup \{[\![\mathsf{P}]\!]^{\cap\!\!\uparrow}[\![\mathsf{b}]\!]^{\cap\!\!\uparrow}T \cup [\![\neg\mathsf{b}]\!]^{\cap\!\!\uparrow}T \mid T \in \mathcal{T}\}$

The *Mixed lift* is the instantiation of the hypercollecting semantics of [Assaf et al., 2017] to the denotations domain Mem. Both operators $\mathsf{H}^{\cap\!\!\uparrow}_{\diamond}$ are monotone functions over the complete lattice $\langle\wp(\wp(\mathsf{Mem})),\subseteq,\cup,\cap,\wp(\mathsf{Mem}),\varnothing\rangle$, hence their least fixpoint exists.

Example 21. Consider P of Example 20. The *Inner lift* hypersemantics is $(\!|\mathsf{P}|\!)^{\curvearrowright}\{\{2,5\}\} = \{\varnothing,\{5\},\{4,5\}\}$, computed as $\{\{2,5\}\} \xrightarrow{\mathsf{H}_{\mathrm{I}}^{\curvearrowright}} \{\varnothing,\{5\},\{4,5\}\}$ where

$$\mathsf{H}_{\mathrm{I}}^{\curvearrowright 0} = \varnothing \qquad \mathsf{H}_{\mathrm{I}}^{\curvearrowright 1} = \{\varnothing,\{2,5\}\} \qquad \mathsf{H}_{\mathrm{I}}^{\curvearrowright 2} = \{\varnothing,\{2,5\},\{2,3,5\}\}$$
$$\mathsf{H}_{\mathrm{I}}^{\curvearrowright 3} = \{\varnothing,\{2,5\},\{2,3,5\},\{2,3,4,5\}\}$$

The *Mixed lift* hypersemantics is $(\!|\mathsf{P}|\!)^{\curvearrowright}\{\{2,5\}\} = \{\{5\},\{4,5\}\}$, computed as $\{\{2,5\}\} \xrightarrow{\mathsf{H}_{\mathrm{M}}^{\curvearrowright}} \{\{5\},\{4,5\}\}$ where

$$\mathsf{H}_{\mathrm{M}}^{\curvearrowright\,0} = \varnothing \qquad \mathsf{H}_{\mathrm{M}}^{\curvearrowright\,1} = \{\{2,5\}\} \qquad \mathsf{H}_{\mathrm{M}}^{\curvearrowright\,2} = \{\{2,5\},\{3,5\}\} \qquad \mathsf{H}_{\mathrm{M}}^{\curvearrowright\,3} = \{\{2,5\},\{3,5\},\{4,5\}\}$$

From the iterates of $\mathsf{H}_{\diamond}^{\curvearrowright}$, with $\diamond \in \{\mathrm{I},\mathrm{M}\}$, we can observe the monotonicity (and the extensivity) of $\mathsf{H}_{\diamond}^{\curvearrowright}$. Both hypersemantics are correct, because $[\![\mathsf{P}]\!]^{\curvearrowright}\{2,5\} \in (\!|\mathsf{P}|\!)^{\curvearrowright}\{\{2,5\}\}$.

Let $(\!|\mathsf{P}|\!)_{\mathrm{I}}^{\curvearrowright}$ and $(\!|\mathsf{P}|\!)_{\mathrm{M}}^{\curvearrowright}$ be the hypersemantics defined in terms of the *Inner* and *Mixed* lifts, respectively, for the while case, and in terms of the direct image lift for all the other statements (again, for sequencing, they are defined by functions composition). The generic and subset-closed hypersemantics, that we denote $(\!|\mathsf{P}|\!)_{\mathrm{G}}^{\curvearrowright}$ and $(\!|\mathsf{P}|\!)_{\mathrm{S}}^{\curvearrowright}$, respectively, are clearly correct: the first computes exactly the strongest program (generic) hyperproperty and the second the strongest program subset-closed hyperproperty. It turns out that also the other two hypersemantics introduced are correct.

Theorem 20 (Correctness)**.** *For every* $X \in \wp(\mathsf{Mem})$ *we have*

$$\{[\![\mathsf{P}]\!]^{\curvearrowright}X\} \subseteq (\!|\mathsf{P}|\!)_{\mathrm{I}}^{\curvearrowright}\{X\} \qquad \textit{and} \qquad \{[\![\mathsf{P}]\!]^{\curvearrowright}X\} \subseteq (\!|\mathsf{P}|\!)_{\mathrm{M}}^{\curvearrowright}\{X\}$$

Proof. The proof can be found in Appendix A. □

This result tells us that these hypersemantics can be soundly used for the verification of hyperproperties of P, unfortunately adding some further spurious information not directly due to approximation, i.e. spurious elements of $\wp(\wp(\mathsf{Mem}))$. Luckily, for subset-closed hyperproperties this is not a real concern. In fact when $\mathfrak{cHp} \in \mathtt{SSC}^{\mathtt{H}}$, we have that $\mathsf{P} \models \mathfrak{cHp}$ if and only if $\wp([\![\mathsf{P}]\!]^{\curvearrowright}I) \subseteq \mathfrak{cHp}$. Furthermore, the two hypersemantics introduced above, are related as follows.

Proposition 13. $\forall \mathcal{X} \in \wp(\wp(\mathsf{Mem}))$*:* $(\!|\mathsf{P}|\!)_{\mathrm{M}}^{\curvearrowright}\mathcal{X} \subseteq (\!|\mathsf{P}|\!)_{\mathrm{S}}^{\curvearrowright}\mathcal{X}$ *and* $(\!|\mathsf{P}|\!)_{\mathrm{I}}^{\curvearrowright}\mathcal{X} \subseteq (\!|\mathsf{P}|\!)_{\mathrm{S}}^{\curvearrowright}\mathcal{X}$.

Hence we can state that all the proposed hypersemantics are complete verification methods for subset-closed hyperproperties.

Theorem 21 (Completeness)**.** *Let* $\mathfrak{Hp} \in \mathtt{SSC}^{\mathtt{H}}$*, then:*

$$\mathsf{P} \models \mathfrak{Hp} \;\Leftrightarrow\; (\!|\mathsf{P}|\!)_{\mathrm{I}}^{\curvearrowright}\{I\} \subseteq \mathfrak{Hp} \;\Leftrightarrow\; (\!|\mathsf{P}|\!)_{\mathrm{M}}^{\curvearrowright}\{I\} \subseteq \mathfrak{Hp}$$

Proof. The theorem follows from Proposition 13, from the fact that $(\!|\mathsf{P}|\!)_{\mathrm{S}}^{\curvearrowright}\mathcal{X} \subseteq \wp([\![P]\!]^{\curvearrowright}\bigcup\mathcal{X})$ for every $\mathcal{X}$ in $\wp(\wp(\mathsf{Mem}))$ and by correctness (Theorem 20). □

Hence the theorem says that even if the hypersemantics insert spurious information, that information do not lowers the precision of the analysis, when we deal with subset-closed hyperproperties. Note that $(\!|\mathsf{P}|\!)_{\mathrm{G}}^{\curvearrowright}$ is complete for generic hyperproperties, not only for the subset-closed one.

A More Complex Example: Partial Trace Semantics. The constructions presented above can be applied also to more complex base semantics. For instance, we show now how we can compute the partial trace semantics $\tau^{\vec{\alpha}}$ of a program and how to lift it to the hyperlevel. The partial trace semantics is a complete verification method for safety trace properties on finite executions. First, consider the following semantic operator $[\![\mathsf{P}]\!]^{\vec{\alpha}} \in \wp(\Sigma^{\vec{\alpha}}) \to \wp(\Sigma^{\vec{\alpha}})$, defined for every $\mathsf{P} \in \mathsf{Imp}$ as follow:

$$[\![\mathsf{P}]\!]^{\vec{\alpha}}\varnothing \triangleq \varnothing \qquad [\![{}^{\mathtt{i}}\mathsf{c}_1{}^{\mathtt{l}} \,.\, {}^{\mathtt{k}}\mathsf{c}_2{}^{\mathtt{f}}]\!]^{\vec{\alpha}} X \triangleq X \cup [\![{}^{\mathtt{k}}\mathsf{c}_2{}^{\mathtt{f}}]\!]^{\vec{\alpha}} \circ [\![{}^{\mathtt{i}}\mathsf{c}_1{}^{\mathtt{l}}]\!]^{\vec{\alpha}} X$$

$$[\![{}^{\mathtt{i}}\mathsf{skip}{}^{\mathtt{f}}]\!]^{\vec{\alpha}} X \triangleq X \cup \{ \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle\langle \mathtt{f}, \mathsf{m}\rangle \mid \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle \in X \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \}$$

$$[\![{}^{\mathtt{i}}\mathsf{x} \coloneqq \mathsf{a}{}^{\mathtt{f}}]\!]^{\vec{\alpha}} X \triangleq X \cup \{ \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle\langle \mathtt{f}, \mathsf{m}[\mathsf{x} \leftarrow\!\shortmid n]\rangle \mid \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle \in X \wedge \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \}$$

$$[\![{}^{\mathtt{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}{}^{\mathtt{f}}]\!]^{\vec{\alpha}} X \triangleq \begin{array}{l} X \cup Y_1 \cup Y_2 \cup \\ \{ \bar{\sigma}\langle \mathtt{t}, \mathsf{m}\rangle\langle \mathtt{f}, \mathsf{m}\rangle \mid \bar{\sigma}\langle \mathtt{t}, \mathsf{m}\rangle \in Y_1 \cup Y_2 \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \} \end{array}$$

$$\text{with:} \quad \mathsf{P}_1 = {}^{\mathtt{l}}\mathsf{c}_1{}^{\mathtt{k}} \quad \mathsf{P}_2 = {}^{\mathtt{h}}\mathsf{c}_2{}^{\mathtt{d}} \quad \mathtt{t} \in \{\mathtt{k}, \mathtt{d}\}$$
$$Y_1 = [\![\mathsf{P}_1]\!]^{\vec{\alpha}} \{ \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle\langle \mathtt{l}, \mathsf{m}\rangle \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt} \wedge \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle \in X \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \}$$
$$Y_2 = [\![\mathsf{P}_2]\!]^{\vec{\alpha}} \{ \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle\langle \mathtt{h}, \mathsf{m}\rangle \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{ff} \wedge \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle \in X \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \}$$

$$[\![{}^{\mathtt{i}}\mathsf{while}\ {}^{\mathtt{l}}\mathsf{b}\ \{\,\mathsf{P}\,\}{}^{\mathtt{f}}]\!]^{\vec{\alpha}} X \triangleq \begin{array}{l} X \cup \mathsf{lfp}^{\subseteq}_{\varnothing} \mathsf{F}^{\vec{\alpha}} \cup \\ \{ \bar{\sigma}\langle \mathtt{l}, \mathsf{m}\rangle\langle \mathtt{f}, \mathsf{m}\rangle \mid \bar{\sigma}\langle \mathtt{l}, \mathsf{m}\rangle \in \mathsf{lfp}^{\subseteq}_{\varnothing} \mathsf{F}^{\mathcal{R}} \wedge \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{ff} \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \} \end{array}$$

$$\text{with:} \quad \mathsf{P} = {}^{\mathtt{k}}\mathsf{c}^{\mathtt{t}} \quad Y = [\![\mathsf{P}]\!]^{\vec{\alpha}} \{ \bar{\sigma}\langle \mathtt{l}, \mathsf{m}\rangle\langle \mathtt{k}, \mathsf{m}\rangle \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt} \wedge \bar{\sigma}\langle \mathtt{l}, \mathsf{m}\rangle \in T \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \}$$
$$\mathsf{F}^{\vec{\alpha}}(T) \triangleq \begin{array}{l} \{ \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle\langle \mathtt{l}, \mathsf{m}\rangle \mid \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle \in X \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \} \cup Y \cup \\ \{ \bar{\sigma}\langle \mathtt{t}, \mathsf{m}\rangle\langle \mathtt{l}, \mathsf{m}\rangle \mid \bar{\sigma}\langle \mathtt{t}, \mathsf{m}\rangle \in Y \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \} \end{array}$$

The function $\mathsf{F}^{\vec{\alpha}} \in \wp(\Sigma^{\vec{\alpha}}) \to \wp(\Sigma^{\vec{\alpha}})$ is monotone on the CPO $\langle \wp(\Sigma^{\vec{\alpha}}), \subseteq, \cup, \varnothing \rangle$, hence its least fixpoint $\bigcup_{n<\omega} \mathsf{F}^{\vec{\alpha}\,n}(\varnothing)$ exists. Indeed, the partial trace semantics $\tau^{\vec{\alpha}}$ of P coincides with $[\![\mathsf{P}]\!]^{\vec{\alpha}} I$. Here I is the set $\{\langle \mathtt{i}, \mathsf{m}\rangle \mid \mathsf{m} \in \mathsf{Mem}^{\mathsf{P}}\}$, assuming $\mathsf{P} = {}^{\mathtt{i}}\mathsf{c}^{\mathtt{f}}$. As an example, we define the semantics computing $\{\tau^{\vec{\alpha}}\}$. Consider the following semantic operator $(\!|\mathsf{P}|\!)^{\vec{\alpha}} \in \wp(\wp(\Sigma^{\vec{\alpha}})) \to \wp(\wp(\Sigma^{\vec{\alpha}}))$, defined for every $\mathsf{P} \in \mathsf{Imp}$ as follow:

$$(\!|\mathsf{P}|\!)^{\vec{\alpha}}\varnothing \triangleq \varnothing \qquad (\!| {}^{\mathtt{i}}\mathsf{c}_1{}^{\mathtt{l}} \,.\, {}^{\mathtt{k}}\mathsf{c}_2{}^{\mathtt{f}} |\!)^{\vec{\alpha}} \mathcal{X} \triangleq \{ [\![{}^{\mathtt{i}}\mathsf{c}_1{}^{\mathtt{l}} \,.\, {}^{\mathtt{k}}\mathsf{c}_2{}^{\mathtt{f}}]\!]^{\vec{\alpha}} X \mid X \in \mathcal{X} \}$$

$$(\!| {}^{\mathtt{i}}\mathsf{skip}{}^{\mathtt{f}} |\!)^{\vec{\alpha}} \mathcal{X} \triangleq \{ [\![{}^{\mathtt{i}}\mathsf{skip}{}^{\mathtt{f}}]\!]^{\vec{\alpha}} X \mid X \in \mathcal{X} \}$$

$$(\!| {}^{\mathtt{i}}\mathsf{x} \coloneqq \mathsf{a}{}^{\mathtt{f}} |\!)^{\vec{\alpha}} X \triangleq \{ [\![{}^{\mathtt{i}}\mathsf{x} \coloneqq \mathsf{a}{}^{\mathtt{f}}]\!]^{\vec{\alpha}} X \mid X \in \mathcal{X} \}$$

$$(\!| {}^{\mathtt{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}{}^{\mathtt{f}} |\!)^{\vec{\alpha}} X \triangleq \{ [\![{}^{\mathtt{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}{}^{\mathtt{f}}]\!]^{\vec{\alpha}} X \mid X \in \mathcal{X} \}$$

$$(\!| {}^{\mathtt{i}}\mathsf{while}\ {}^{\mathtt{l}}\mathsf{b}\ \{\,\mathsf{P}\,\}{}^{\mathtt{f}} |\!)^{\vec{\alpha}} \mathcal{X} \triangleq \left\{ \left\{ \bar{\sigma}\langle \mathtt{l}, \mathsf{m}\rangle\langle \mathtt{f}, \mathsf{m}\rangle \;\middle|\; \begin{array}{l} \bar{\sigma}\langle \mathtt{l}, \mathsf{m}\rangle \in X \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \\ \wedge \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{ff} \end{array} \right\} \;\middle|\; X \in \mathsf{lfp}^{\subseteq}_{\{\varnothing\}} \mathsf{F}^{\vec{\alpha}}_{\mathsf{G}} \right\}$$

$$\text{with:} \quad \mathsf{P} = {}^{\mathtt{k}}\mathsf{c}^{\mathtt{t}}$$
$$\mathcal{Y} = \{ [\![\mathsf{P}]\!]^{\vec{\alpha}} \{ \bar{\sigma}\langle \mathtt{l}, \mathsf{m}\rangle\langle \mathtt{k}, \mathsf{m}\rangle \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt} \wedge \bar{\sigma}\langle \mathtt{l}, \mathsf{m}\rangle \in T \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \} \mid T \in \mathcal{T} \}$$
$$\mathsf{F}^{\vec{\alpha}}_{\mathsf{G}}(\mathcal{T}) \triangleq \begin{array}{l} \{ \{ \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle\langle \mathtt{l}, \mathsf{m}\rangle \mid \bar{\sigma}\langle \mathtt{i}, \mathsf{m}\rangle \in X \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \} \mid X \in \mathcal{X} \} \uplus \mathcal{Y} \uplus \\ \{ \{ \bar{\sigma}\langle \mathtt{t}, \mathsf{m}\rangle\langle \mathtt{l}, \mathsf{m}\rangle \mid \bar{\sigma}\langle \mathtt{t}, \mathsf{m}\rangle \in Y \wedge \bar{\sigma} \in \Sigma_\epsilon^{\vec{\alpha}} \} \mid Y \in \mathcal{Y} \} \end{array}$$

The partial trace generic hypersemantics $\tau_{\mathfrak{o}}^{\vec{\alpha}}$ of P coincides with $(\!|\mathsf{P}|\!)^{\vec{\alpha}}\{I\}$, where I is the set $\{\langle \underline{\mathtt{i}}, \mathsf{m}\rangle \mid \mathsf{m} \in \mathsf{Mem}^{\mathsf{P}}\}$, assuming $\mathsf{P} = {}^{\underline{\mathtt{i}}}\mathsf{c}^{\underline{\mathtt{f}}}$.

7.3.1 On Bounded Hyperproperties

We have already seen that for a k-bounded subset-closed hyperproperty $\mathfrak{chp}$ the verification concerns the subsets of the base semantics with cardinality k. Suppose that the base semantics is $\tau^{\curvearrowright}$, namely the post-conditions semantics. This means that $\mathsf{P} \models \mathfrak{chp}$ if and only if $\{X \subseteq \tau^{\curvearrowright} \mid |X| = k\} \subseteq \mathfrak{chp}$. But this boils down to check $[\![\mathsf{P}]\!]^{\curvearrowright}X \in \mathfrak{chp}$, for every $X \subseteq I$ such that $|X| = k$. Now consider the set $I^{|k} \triangleq \{X \subseteq I \mid |X| = k\}$. The following theorem tells us how we can use the post-conditions hypersemantics for verifying bounded hyperproperties.

Theorem 22. *Given* $\mathfrak{chp} \in \mathtt{SSC}_k^{\mathtt{H}}$*, we have that* $\mathsf{P} \models \mathfrak{chp}$ *if and only if* $(\!|\mathsf{P}|\!)^{\curvearrowright}I^{|k} \subseteq \mathfrak{chp}$.

Proof. By correctness (Theorem 20) and completeness (Theorem 21), for subset-closed hyperproperties, of the post-conditions hypersemantics, we have that $\{[\![\mathsf{P}]\!]^{\curvearrowright}X \mid X \in I^{|k}\} \subseteq (\!|\mathsf{P}|\!)^{\curvearrowright}I^{|k}$. Then, recalling that we are in a deterministic setting, we have that $\{[\![\mathsf{P}]\!]^{\curvearrowright}X \mid X \in I^{|k}\} = \{X \subseteq [\![\mathsf{P}]\!]^{\curvearrowright}I \mid |X| = k\}$. Due to Proposition 9 the theorem follows. □

In the next chapter we will see how to apply Theorem 22 in order to verify information flow specifications, which are formalized as 2-bounded hyperproperties.

8 Application: Verification of Information Flows

Until now we have seen how to define hypersemantics and hyperdomains. In this chapter we give an example of application of the theoretical results presented. We take in consideration information flows, hence the goal of this chapter is to define a verification mechanism for (Abstract) Non-Interference. In particular we define an abstract semantics approximating the hypersemantics presented at the end of Chapter 7.

The original formulation of *Non-Interference* [Cohen, 1977] takes in consideration only two security levels: *private* (H), i.e. information that have to be kept secret, and *public* (L), i.e. information that could be freely released. A program is said *non-interferent* if there are no information flows from private (input) variables to public (output) variables, and it is said *interferent* otherwise. In the following, we define the classic notion of Non-Interference for programs in Imp. A program $\mathsf{P} = {}^{\mathtt{i}}\mathsf{c}^{\mathtt{f}}$ is non-interferent if and only if for every pair of memories $\mathsf{m}_1, \mathsf{m}_2$, such that $\mathsf{m}_1 \overset{\mathsf{L}}{=} \mathsf{m}_2$, $\langle\langle \mathtt{i}, \mathsf{m}_1\rangle, \mathsf{P}\rangle \rightarrowtriangle^* \langle \mathtt{f}, \mathsf{m}_1'\rangle$ and $\langle\langle \mathtt{i}, \mathsf{m}_2\rangle, \mathsf{P}\rangle \rightarrowtriangle^* \langle \mathtt{f}, \mathsf{m}_2'\rangle$, we have $\mathsf{m}_1' \overset{\mathsf{L}}{=} \mathsf{m}_2'$. As usual, the relation $\overset{\mathsf{L}}{=}$ says that two memories are equal modulo public variables, namely $\mathsf{m} \overset{\mathsf{L}}{=} \mathsf{m}'$ if and only if $\forall \mathsf{x} \,.\, \Gamma(\mathsf{x}) = \mathsf{L} \Rightarrow \mathsf{m}(\mathsf{x}) = \mathsf{m}'(\mathsf{x})$. This specification checks the input/output relation between executions, so we represent programs computations with just the initial and the final states. Furthermore, it is termination insensitive, hence we ignore divergent computations. The denotations domain is $\mathbb{Den} \triangleq \Sigma^+ = \Sigma \times \Sigma$, where we recall that $\Sigma = \mathsf{Lab} \times \mathsf{Mem}$.

Remark. We could have defined Non-Interference on the more concrete denotations domain $\Sigma^{\overrightarrow{\infty}}$ and retrieve the definition of the specification on Σ^+ by abstraction, but this is only a conceptual step, since Σ^+ is the most abstract domain sufficiently precise to express Non-Interference.

Non-Interference is an hyperproperty, hence we need to define it on $\wp(\wp(\Sigma \times \Sigma))$.

Definition 32 (Non-Interference for Imp)**.** The classic notion of *Non-Interference*, NI in short, for programs in Imp is defined as:

$$\mathsf{NI} \triangleq \bigcup_{\mathtt{i},\mathtt{f} \in \mathsf{Lab}} \left\{ X \subseteq \Sigma \times \Sigma \;\middle|\; \begin{array}{l} \forall \langle\langle \mathtt{i}, \mathsf{m}_1\rangle, \langle \mathtt{f}, \mathsf{m}_1'\rangle\rangle, \langle\langle \mathtt{i}, \mathsf{m}_2\rangle, \langle \mathtt{f}, \mathsf{m}_2'\rangle\rangle \in X \,. \\ \mathsf{m}_1 \overset{\mathsf{L}}{=} \mathsf{m}_2 \Rightarrow \mathsf{m}_1' \overset{\mathsf{L}}{=} \mathsf{m}_2' \end{array} \right\}$$

Basically, the definition says that a program is non-interferent when its execution starting from two arbitrary memories L-equivalent yields two memories L-equivalent. The definition, is implicitly parametric on a typing environment Γ, which is supposed to be defined for every variable.

Weakening Non-Interference. The limitation of the above notion is that it is extremely restrictive. Indeed, Non-Interference requires that any change of private data has not to be revealed through the observation of the public one, but any real system is intended to leak some kind of information (one classic example is the form for passwords check). Hence, a weakening of Non-Interference is necessary.

As already highlighted, we adopt Abstract Non-Interference [Giacobazzi and Mastroeni, 2004], as a formal method to weak Non-Interference. Abstract Non-Interference makes Non-Interference parametric on two properties, each one modeling different aspects of the information flow: what the attacker can observe and what information is allowed/not allowed to flow. In general, we can suppose that the attacker may not have the same constraints in observing inputs and outputs but, in the following, we assume that the attacker's observation precision is the same, both in input and in output.

The attacker is characterized as a property ρ representing what can be observed about the public input/output of programs. An abstraction is a property of data, hence an attacker can distinguish data up to particular properties (for instance, an attacker could be able to distinguish the parity of variables but not their sign). As far as declassification is concerned, we suppose to specify what private information is allowed to flow. Let ϕ be the input property that may flow in output, without violating the information flow policy: we check (abstract) non-interference only for those private inputs agreeing on the property ϕ. This models the information that may flow, since it is not interesting to check whether its variation is visible through the output.

Using the notation introduced in Subsection 7.2.2, we can elegantly generalize the definition of Non-Interference w.r.t. a property $\phi \in \mathsf{uco}(\wp(\mathbb{Z}))$ of input private variables which may flow and an observable property $\rho \in \mathsf{uco}(\wp(\mathbb{Z}))$ of input/output public variables. A program $\mathsf{P} = {}^{\underline{\mathtt{i}}}\mathsf{c}^{\underline{\mathtt{f}}}$ is abstract non-interferent if and only if for every pair of memories $\mathsf{m}_1, \mathsf{m}_2$, such that $\nu^{\rho\times\phi}(\mathsf{m}_1) = \nu^{\rho\times\phi}(\mathsf{m}_2)$, $\langle\langle \underline{\mathtt{i}}, \mathsf{m}_1\rangle, \mathsf{P}\rangle \rightarrowtriangle^* \langle \underline{\mathtt{f}}, \mathsf{m}'_1\rangle$ and $\langle\langle \underline{\mathtt{i}}, \mathsf{m}_2\rangle, \mathsf{P}\rangle \rightarrowtriangle^* \langle \underline{\mathtt{f}}, \mathsf{m}'_2\rangle$, we have $\nu^{\rho\times\tau}(\mathsf{m}'_1) = \nu^{\rho\times\tau}(\mathsf{m}'_2)$.

Again, the denotations domain is $\mathbb{D}\mathtt{en} \triangleq \Sigma^+$, hence Abstract Non-Interference is the following hyperproperty on $\wp(\wp(\Sigma\times\Sigma))$.

Definition 33 (Abstract Non-Interference)**.** Let $\phi \in \mathsf{uco}(\wp(\mathbb{Z}))$ be the property of input private variables which may flow and $\rho \in \mathsf{uco}(\wp(\mathbb{Z}))$ be the observable property of input/output public variables. *Abstract Non-Interference* w.r.t. ϕ and ρ, ANI^ρ_ϕ for short, is

$$\mathsf{ANI}^\rho_\phi \triangleq \bigcup_{\underline{\mathtt{i}},\underline{\mathtt{f}}\in\mathsf{Lab}} \left\{ X \subseteq \Sigma\times\Sigma \;\middle|\; \begin{array}{l} \forall\langle\langle \underline{\mathtt{i}}, \mathsf{m}_1\rangle,\langle \underline{\mathtt{f}}, \mathsf{m}'_1\rangle\rangle, \langle\langle \underline{\mathtt{i}}, \mathsf{m}_2\rangle,\langle \underline{\mathtt{f}}, \mathsf{m}'_2\rangle\rangle \in X\,. \\ \nu^{\rho\times\phi}(\mathsf{m}_1) = \nu^{\rho\times\phi}(\mathsf{m}_2) \Rightarrow \nu^{\rho\times\tau}(\mathsf{m}'_1) = \nu^{\rho\times\tau}(\mathsf{m}'_2) \end{array} \right\}$$

This means that a program P satisfies Abstract Non-Interference relatively to a public input/output observation ρ and a private input property ϕ that may flow if, whenever the public input values have the same property ρ and the private input values have the same property ϕ, then the execution of P leads to ρ-indistinguishable public values. It is worth noting that Non-Interference is an instance of ANI, with $\rho = \iota$, since NI deals with all-power observers, namely attackers observing values in the most precise way, and $\phi = \tau$, meaning that no declassification is allowed since all values have the property τ. Hence, $\mathsf{NI} = \mathsf{ANI}^\iota_\tau$. In other words, since input/output L variables need to have the property "to be equal", we model ρ as the identity abstraction ι. Dually, in NI there is no declassification, hence we have

$\phi = \tau$, meaning that we need to check the dependence for every combination of H inputs (in fact $\tau(\{n\}) = \tau(\{n'\})$, for every pair of elements $n, n' \in \mathbb{Z}$).

8.1 Abstract Hypersemantics for Abstract Non-Interference

Since ANI is parametric on the property to be checked, an unique verifier is not possible. In this section we give a hypersemantics and a *parametric abstract semantics* for the verification of Abstract Non-Interference. This means that we give one hypersemantics, whose abstract versions, parametric on the ANI properties to check, are useful for ANI verification. In Section 8.2 we go deeply in the details of a verifier for Non-Interference. A prototype analyzer has been written in order to test the abstract semantics.

For doing so, we follow the construction introduced in Section 7.2 for k-bounded subset-closed hyperproperties (indeed ANI is 2-bounded). First, we give the definition of the hypersemantics, computing at the level of sets of sets. Then we instantiate the hyperlevel constants domain of Section 6.2 to Abstract Non-Interference. The latter, in its original formulation, may be not machine-representable (it may have an uncountable set of elements). This depends on the structure of the domain ρ. For such cases we show how to approximate the hyperlevel constants domain, in order to make its implementation feasible. Finally we show how to design the abstract semantics.

8.1.1 The Hypersemantics

Since Abstract Non-Interference is 2-bounded, it is equisatisfiable to the hyperproperty $(\mathsf{ANI}^\rho_\phi)^{|2} \triangleq \{X \subseteq \mathsf{ANI}^\rho_\phi \mid |X| = 2\}$. Furthermore, as we have seen in Sec. 7.2, we can verify Abstract Non-Interference in a simpler domain, namely we can collect states instead of input/output traces of states. We can partition ANI in two simpler hyperproperties:

$$(\mathsf{ANI}^\rho_\phi)^{=} \triangleq \bigcup_{\mathtt{l},\mathtt{f} \in \mathsf{Lab}} \left\{ X \subseteq \Sigma \times \Sigma \;\middle|\; \begin{array}{l} X = \{\langle\langle \mathtt{l}, \mathsf{m}_1\rangle, \langle \mathtt{f}, \mathsf{m}'_1\rangle\rangle, \langle\langle \mathtt{l}, \mathsf{m}_2\rangle, \langle \mathtt{f}, \mathsf{m}'_2\rangle\rangle\} \wedge \\ \nu^{\rho\times\phi}(\mathsf{m}_1) = \nu^{\rho\times\phi}(\mathsf{m}_2) \wedge \nu^{\rho\times\tau}(\mathsf{m}'_1) = \nu^{\rho\times\tau}(\mathsf{m}'_2) \end{array} \right\}$$

$$(\mathsf{ANI}^\rho_\phi)^{\neq} \triangleq \bigcup_{\mathtt{l},\mathtt{f} \in \mathsf{Lab}} \left\{ X \subseteq \Sigma \times \Sigma \;\middle|\; \begin{array}{l} X = \{\langle\langle \mathtt{l}, \mathsf{m}_1\rangle, \langle \mathtt{f}, \mathsf{m}'_1\rangle\rangle, \langle\langle \mathtt{l}, \mathsf{m}_2\rangle, \langle \mathtt{f}, \mathsf{m}'_2\rangle\rangle\} \wedge \\ \nu^{\rho\times\phi}(\mathsf{m}_1) \neq \nu^{\rho\times\phi}(\mathsf{m}_2) \end{array} \right\}$$

Then we can skip the check for $(\mathsf{ANI}^\rho_\phi)^{\neq}$ since it is always satisfied, by every program. Furthermore, we can note that $(\mathsf{ANI}^\rho_\phi)^{=}$ is a second-order not-relational hyperproperty, meaning that it is equivalent to the hyperproperty $\langle \mathcal{X}, \mathcal{Y}\rangle \in \wp(\wp(\Sigma)) \times \wp(\wp(\Sigma))$, defined as:

$$\mathcal{X} \triangleq \bigcup_{\mathtt{l} \in \mathsf{Lab}} \{X \subseteq \Sigma \mid X = \{\langle \mathtt{l}, \mathsf{m}_1\rangle, \langle \mathtt{l}, \mathsf{m}_2\rangle\} \wedge \nu^{\rho\times\phi}(\mathsf{m}_1) = \nu^{\rho\times\phi}(\mathsf{m}_2)\}$$

$$\mathcal{Y} \triangleq \bigcup_{\mathtt{f} \in \mathsf{Lab}} \{X \subseteq \Sigma \mid X = \{\langle \mathtt{f}, \mathsf{m}'_1\rangle, \langle \mathtt{f}, \mathsf{m}'_2\rangle\} \wedge \nu^{\rho\times\tau}(\mathsf{m}'_1) = \nu^{\rho\times\tau}(\mathsf{m}'_2)\}$$

In the definition above, control points are totally irrelevant, indeed we have that $\langle \mathcal{X}, \mathcal{Y}\rangle$ is isomorphic to $\langle \mathcal{X}', \mathcal{Y}'\rangle$, where:

$$\mathcal{X}' \triangleq \{X \subseteq \mathsf{Mem} \mid X = \{\mathsf{m}_1, \mathsf{m}_2\} \wedge \nu^{\rho\times\phi}(\mathsf{m}_1) = \nu^{\rho\times\phi}(\mathsf{m}_2)\}$$

$$\mathcal{Y}' \triangleq \{X \subseteq \mathsf{Mem} \mid X = \{\mathsf{m}'_1, \mathsf{m}'_2\} \wedge \nu^{\rho\times\tau}(\mathsf{m}'_1) = \nu^{\rho\times\tau}(\mathsf{m}'_2)\}$$

This basically means that it suffices to check whether $[\![\mathsf{P}]\!]^{\curvearrowright} X \in \mathcal{Y}'$, for every $X \in \mathcal{X}'$. We can verify this at once using a post-conditions hypersemantics $(\!|\mathsf{P}|\!)^{\curvearrowright}$, namely checking whether $(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{X}' = \mathcal{Y}'$. This observations induce the following proposition.

Proposition 14. $\mathsf{P} \models \mathsf{ANI}^{\rho}_{\phi}$ *if and only if* $(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}^2_{\rho,\phi} \subseteq \mathtt{equiv}^{\rho}_{\mathsf{L}}$, *where:*

$$\mathcal{I}^2_{\rho,\phi} \triangleq \{\{\mathsf{m},\mathsf{m}'\} \subseteq \mathsf{Mem}^{\mathsf{P}} \mid \nu^{\rho\times\phi}(\mathsf{m}) = \nu^{\rho\times\phi}(\mathsf{m})\}$$
$$\mathtt{equiv}^{\rho}_{\mathsf{L}} \triangleq \{X \subseteq \mathsf{Mem} \mid \forall \mathsf{m},\mathsf{m}' \in X \,.\, \nu^{\rho\times\tau}(\mathsf{m}) = \nu^{\rho\times\tau}(\mathsf{m}')\}$$

As we have seen in Chapter 7, we can have different correct post-conditions hypersemantics, depending on how we define the semantics for loops. We chose to adopt the following solution. The post-conditions hypersemantics $(\!|\mathsf{P}|\!)^{\curvearrowright} \in \wp(\wp(\mathsf{Mem})) \to \wp(\wp(\mathsf{Mem}))$ is:

$$(\!|\mathsf{P}|\!)^{\curvearrowright}\varnothing \triangleq \varnothing \qquad (\!|{}^{\mathtt{i}}\mathsf{c}_3{}^{\mathtt{l}} \,.\, {}^{\mathtt{k}}\mathsf{c}_2{}^{\mathtt{f}}|\!)^{\curvearrowright}\mathcal{X} \triangleq (\!|{}^{\mathtt{k}}\mathsf{c}_2{}^{\mathtt{f}}|\!)^{\curvearrowright} \circ (\!|{}^{\mathtt{i}}\mathsf{c}_1{}^{\mathtt{l}}|\!)^{\curvearrowright}\mathcal{X} \qquad (\!|{}^{\mathtt{i}}\mathsf{skip}^{\mathtt{f}}|\!)^{\curvearrowright}\mathcal{X} \triangleq \mathcal{X}$$

$$(\!|{}^{\mathtt{i}}\mathsf{x} := \mathsf{a}^{\mathtt{f}}|\!)^{\curvearrowright}\mathcal{X} \triangleq \{\{\mathsf{m}[\mathsf{x} \leftarrow n] \mid \mathsf{m} \in X \wedge \langle \mathsf{a},\mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \mathcal{X}\}$$

$$(\!|{}^{\mathtt{i}}\mathsf{if\ b\ then\ \{\,P_1\,\}\ else\ \{\,P_2\,\}}^{\mathtt{f}}|\!)^{\curvearrowright}X \triangleq \{[\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}X \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright}X \mid X \in \mathcal{X}\}$$

$$(\!|{}^{\mathtt{i}}\mathsf{while}\ {}^{\mathtt{l}}\mathsf{b}\ \{\,\mathsf{P}\,\}^{\mathtt{f}}|\!)^{\curvearrowright}\mathcal{X} \triangleq (\!|\neg\mathsf{b}|\!)^{\curvearrowright}(\mathsf{lfp}^{\subseteq}_{\varnothing}\,\mathsf{H}^{\curvearrowright}) \text{ where:}$$
$$\mathsf{H}^{\curvearrowright}(\mathcal{T}) \triangleq \mathcal{X} \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}T \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}T \mid T \in \mathcal{T}\}$$

This semantics is very simple (we do not even need to change the computational domain) and yet it is complete for subset-closed hyperproperties (Theorem 21), as Abstract Non-Interference.

8.1.2 Towards Abstraction

Unfortunately, $(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}^2_{\rho,\phi}$ and $\mathtt{equiv}^{\rho}_{\mathsf{L}}$ are not computable in general, hence we need approximations. In order to compute a sound approximation of $(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}^2_{\rho,\phi}$, we rely on abstract interpretation. The first step is to define the abstract domain used to verify Abstract Non-Interference.

8.1.2.1 The Abstract Domain for Abstract Non-Interference.

The domain is an instance of the *hyperlevel* (*abstract*) *constants* (Section 6.2), and it checks whether a set of sets of values contains constant sets modulo ρ. Namely it contains only sets of values indistinguishable by ρ. In the following, all abstractions α are continuous (i.e. they preserve the least upper bound of chains), so their left adjoint α^{-} always exists[1].

The domain is an abstraction of $\wp(\wp(\mathbb{Z}))$. We have seen that a hyperdomain, i.e. a domain suitable for the verification of hyperproperties, can be decomposed in an inner abstraction, approximating traces, and in an outer abstraction, approximating properties of traces. Following this idea, we use as inner abstraction just ρ. Instead, the outer abstraction checks whether the inner abstraction always returns (atomic) constant values, not necessarily the same. For instance, suppose ρ can distinguish values up-to their sign, then $\{\{1\},\{-1,-3\}\}$ contains only (atomic) constant sets, instead $\{\{1\},\{2,-3\}\}$ contains also a not-constant set (in this case ρ maps $\{2,-3\}$ to $\mathbb{Z}$).

[1] This is a sufficient condition in order to form a Galois connection.

Given a upper closure operator $\rho \in \mathsf{uco}(\wp(\mathbb{Z}))$, we denote with Atm^ρ the set of its atoms, namely the set $\mathrm{Atm}^\rho \triangleq \{X \in \rho(\wp(\mathbb{Z})) \mid \forall Y \in \rho(\wp(\mathbb{Z}))\,.\, Y \subseteq X \Rightarrow (Y = \varnothing \vee Y = X)\}$[2] of elements covering the bottom ($\varnothing$ in this case). As explained in Section 6.2, we assume that Atm^ρ forms a partition of $\mathbb{Z}$. In order to perform hyperlevel constants on ρ we consider the set of its atoms: these latter precisely identify the properties of concrete values observed in ρ. The hyperlevel (abstract) constants domain for ρ is $\rho_{\mathfrak{hc}} \triangleq \wp(\mathrm{Atm}^\rho) \cup \{\rho(\wp(\mathbb{Z}))\}$. The abstraction function $\alpha_{\mathfrak{hc}} \in \wp(\wp(\mathbb{Z})) \to \rho_{\mathfrak{hc}}$ is $\alpha_{\mathfrak{hc}}(\mathcal{X}) \triangleq \varnothing$ if $\mathcal{X} = \varnothing$, $\alpha_{\mathfrak{hc}}(\mathcal{X}) \triangleq \{\rho(X) \mid X \in \mathcal{X}\}$ if $\{\rho(X) \mid X \in \mathcal{X}\} \subseteq \mathrm{Atm}^\rho$ and $\alpha_{\mathfrak{hc}}(\mathcal{X}) \triangleq \rho(\wp(\mathbb{Z}))$ otherwise. Its left adjoint is the identity function, namely $\gamma_{\mathfrak{hc}} \triangleq \alpha^-_{\mathfrak{hc}} = \mathsf{id}$. Then we have the Galois insertion:

$$\langle \wp(\wp(\mathbb{Z})), \subseteq \rangle \underset{\alpha_{\mathfrak{hc}}}{\overset{\gamma_{\mathfrak{hc}}}{\leftrightarrows}} \langle \rho_{\mathfrak{hc}}, \subseteq \rangle$$

Example 22. Let ρ be the upper closure operator distinguishing the sign of integers, namely $\rho = \mathsf{Sgn}$ defined in Section 6.2. The set of its atoms is $\mathrm{Atm}^{\mathsf{Sgn}} = \{\mathbb{Z}_{<0}, \mathbb{Z}_{=0}, \mathbb{Z}_{>0}\}$. The hyperlevel Sgn-constants domain is $\mathsf{Sgn}_{\mathfrak{hc}} \triangleq \wp(\mathrm{Atm}^{\mathsf{Sgn}}) \cup \{\mathsf{Sgn}(\wp(\mathbb{Z}))\}$, and $\alpha_{\mathfrak{hc}}(\mathcal{X}) \triangleq \mathcal{X}$ if, $\mathcal{X} = \varnothing$ or $\forall X \in \mathcal{X}\,.\,\mathsf{Sgn}(X) \in \mathrm{Atm}^{\mathsf{Sgn}}$, and $\alpha_{\mathfrak{hc}}(\mathcal{X}) = \mathsf{Sgn}(\wp(\mathbb{Z}))$ otherwise.

The set of sets $\mathcal{X} = \{\{-1,-2\},\{3,5,4\}\}$ is abstracted in $\{\mathbb{Z}_{<0}, \mathbb{Z}_{>0}\}$, meaning that the set is constant w.r.t. Sgn. Instead, $\mathcal{Y} = \{\{-1,3\},\{2,4,9\}\}$ is abstracted to $\mathsf{Sgn}(\wp(\mathbb{Z}))$ since there are no atoms approximating $\mathsf{Sgn}(\{-1,3\}) = \mathbb{Z}$, meaning that $\mathcal{Y}$ is not constant. Suppose now that $\mathcal{X}, \mathcal{Y}$ are the sets of possible sets of values a variable x may take; $\alpha_{\mathfrak{hc}}(\mathcal{X}) \subseteq \mathrm{Atm}^{\mathsf{Sgn}}$ means that $\forall X \in \mathcal{X}$ and $\forall n, n' \in X\,.\,\mathsf{Sgn}(\{n\}) = \mathsf{Sgn}(\{n'\})$, i.e. the variable x is constant w.r.t. Sgn. Conversely, $\alpha_{\mathfrak{hc}}(\mathcal{Y}) = \mathsf{Sgn}(\wp(\mathbb{Z}))$ means that $\exists Y \in \mathcal{Y}$ such that $\exists n, n' \in Y\,.\,\mathsf{Sgn}(\{n\}) \neq \mathsf{Sgn}(\{n'\})$, i.e. the variable x is not constant w.r.t. Sgn.

This domain is sufficient for Abstract Non-Interference verification, as we will prove in Theorem 23 but it may be not machine-representable. We solve the problem approximating the domain with a finite version.

In the Case of Uncountable Domains. In the presentation of $\rho_{\mathfrak{hc}}$ we used an upper closure operator to describe the abstract domain. In an implementation of this latter one would use a machine-representable description of the domain, isomorphic to $\rho_{\mathfrak{hc}}$, defining all lattice operators accordingly. If Atm^ρ is finite, no problems arise since $\wp(\mathrm{Atm}^\rho)$ is still finite (this is the case, for $\rho = \mathsf{Sgn}$). If Atm^ρ is infinite, we have that $\wp(\mathrm{Atm}^\rho)$ is uncountable and hence we need to define a simpler domain, which is machine-representable but still able to verify Abstract Non-Interference (this is the case, for instance, when ρ is the domain for constants propagation: its set of atoms is isomorphic to $\mathbb{Z}$). The solution we adopt is to abstract precisely the singletons of atoms and to lose precision on other elements of the poweset.

Let $\overline{\mathrm{Atm}}^\rho$ be a set isomorphic to Atm^ρ, aiming at representing sets containing only one singleton, i.e. of the form $\{\mathfrak{a}\}$, given $\mathfrak{a} \in \mathrm{Atm}^\rho$, which is the information we want to observe precisely, and let $^- \in \mathrm{Atm}^\rho \to \overline{\mathrm{Atm}}^\rho$ a bijection. Furthermore, we denote by $\mathcal{A}^\rho$ the abstract element representing the set of all atoms. Then we define $\mathtt{C}^\rho \triangleq \{\bar{\mathfrak{a}} \mid \mathfrak{a} \in \mathrm{Atm}^\rho\} \cup \{\bot, \top, \mathcal{A}^\rho\}$, with the partial order $\trianglelefteq \subseteq \mathtt{C}^\rho \times \mathtt{C}^\rho$ defined as $c_1 \trianglelefteq c_2 \triangleq (c_1 = \bot \vee c_1 = c_2 \vee (c_1 = \bar{\mathfrak{a}} \wedge$

[2] We use the fact that an upper closure operator can be identified with the set of its fixpoints $\rho(\wp(\mathbb{Z})) = \{X \in \wp(\mathbb{Z}) \mid \rho(X) = X\}$.

$c_2 = \mathcal{A}^\rho) \vee c_2 = \top$). Now consider the abstraction $\alpha_{\mathtt{C}^\rho} \in \rho_{\mathfrak{hc}} \to \mathtt{C}^\rho$ and the concretization $\gamma_{\mathtt{C}^\rho} \in \mathtt{C}^\rho \to \rho_{\mathfrak{hc}}$:

$$\alpha_{\mathtt{C}^\rho}(\mathcal{X}) \triangleq \begin{cases} \bot & \text{if } \mathcal{X} = \varnothing \\ \bar{\mathfrak{a}} & \text{if } \mathcal{X} = \{\mathfrak{a}\} \\ \mathcal{A}^\rho & \text{if } \mathcal{X} \subseteq \mathrm{Atm}^\rho \wedge |\mathcal{X}| > 1 \\ \top & \text{otherwise} \end{cases} \qquad \gamma_{\mathtt{C}^\rho}(c) \triangleq \begin{cases} \varnothing & \text{if } c = \bot \\ \{\mathfrak{a}\} & \text{if } c = \bar{\mathfrak{a}} \\ \mathrm{Atm}^\rho & \text{if } c = \mathcal{A}^\rho \\ \rho(\wp(\mathbb{Z})) & \text{otherwise} \end{cases}$$

Then we have the Galois insertion $(\rho_{\mathfrak{hc}}, \alpha_{\mathtt{C}^\rho}, \gamma_{\mathtt{C}^\rho}, \mathtt{C}^\rho)$. The domain $\langle \mathtt{C}^\rho, \trianglelefteq, \veebar, \barwedge, \bot, \top \rangle$ is a complete lattice where $\veebar, \barwedge \in \mathtt{C}^\rho \times \mathtt{C}^\rho \to \mathtt{C}^\rho$ are:

$$c_1 \veebar c_2 \triangleq \begin{cases} \top & \text{if } c_1 = \top \vee c_2 = \top \\ \bot & \text{if } c_1 = \bot \wedge c_2 = \bot \\ \bar{\mathfrak{a}} & \text{if } (c_1 = \bot \wedge c_2 = \bar{\mathfrak{a}}) \vee (c_1 = \bar{\mathfrak{a}} \wedge c_2 = \bot) \vee (c_1 = \bar{\mathfrak{a}} \wedge c_2 = \bar{\mathfrak{a}}) \\ \mathcal{A}^\rho & \text{otherwise} \end{cases}$$

$$c_1 \barwedge c_2 \triangleq \begin{cases} \top & \text{if } c_1 = \top \wedge c_2 = \top \\ \bot & \text{if } c_1 = \bot \vee c_2 = \bot \vee (c_1 = \bar{\mathfrak{a}}_1 \neq \bar{\mathfrak{a}}_2 = c_2) \\ \bar{\mathfrak{a}} & \text{if } (c_1 = \bar{\mathfrak{a}} \wedge c_2 \in \{\mathcal{A}^\rho, \top\}) \vee (c_2 = \bar{\mathfrak{a}} \wedge c_1 \in \{\mathcal{A}^\rho, \top\}) \\ \mathcal{A}^\rho & \text{otherwise} \end{cases}$$

By composition, $\alpha_{h^\rho} \triangleq \alpha_{\mathtt{C}^\rho} \circ \alpha_{\mathfrak{hc}}$ and $\gamma_{h^\rho} \triangleq \gamma_{\mathfrak{hc}} \circ \gamma_{\mathtt{C}^\rho}$ form the Galois insertion:

$$\langle \wp(\wp(\mathbb{Z})), \subseteq \rangle \underset{\alpha_{h^\rho}}{\overset{\gamma_{h^\rho}}{\leftrightarrows}} \langle \mathtt{C}^\rho, \trianglelefteq \rangle$$

Example 23. The classic domain for constants propagation is given by the upper closure operator $\mathtt{Cprp}$, defined as $\mathtt{Cprp}(X) \triangleq X$ if $X \in \{\{n\} \mid n \in \mathbb{Z}\} \cup \{\varnothing\}$ and $\mathtt{Cprp}(X) \triangleq \mathbb{Z}$ otherwise. The set of its atoms is $\mathrm{Atm}^{\mathtt{Cprp}} = \{\{n\} \mid n \in \mathbb{Z}\}$, which is isomorphic to $\mathbb{Z}$ and hence it is (countably) infinite. In this case, we can define $\mathtt{C}^{\mathtt{Cprp}}$ as the set $\mathbb{Z} \cup \{\bot, \top, \mathcal{A}^{\mathtt{Cprp}}\}$ and $\alpha_{\mathtt{C}^{\mathtt{Cprp}}} \in \mathtt{Cprp}_{\mathfrak{hc}} \to \mathtt{C}^{\mathtt{Cprp}}$ as: $\alpha_{\mathtt{C}^{\mathtt{Cprp}}}(\mathcal{X}) \triangleq \bot$ if $\mathcal{X} = \varnothing$; $\alpha_{\mathtt{C}^{\mathtt{Cprp}}}(\mathcal{X}) \triangleq n$ if $\mathcal{X} = \{\{n\}\}$; $\alpha_{\mathtt{C}^{\mathtt{Cprp}}}(\mathcal{X}) \triangleq \mathcal{A}^{\mathtt{Cprp}}$ if $\mathcal{X} \subseteq \mathrm{Atm}^{\mathtt{Cprp}} \wedge |\mathcal{X}| > 1$; $\alpha_{\mathtt{C}^{\mathtt{Cprp}}}(\mathcal{X}) = \top$ otherwise. Note that, in this case, $\overline{\mathrm{Atm}}^{\mathtt{Cprp}} = \mathbb{Z}$ and the bijection $^-$ maps sets $\{n\} \in \mathrm{Atm}^{\mathtt{Cprp}}$ to integers $n \in \mathbb{Z}$.

Memories Point-Wise Lift. Once we define the abstract domain for (sets of sets of) values, as usual in static analysis we need to extend it to (sets of sets of) memories. From now on, for simplicity, we use $\langle \mathtt{C}^\rho, \trianglelefteq, \veebar, \barwedge, \bot, \top \rangle$ as the values-domain for ANI. Note that the construction given in the previous paragraph can be applied also to domains $\rho_{\mathfrak{hc}}$ already machine-representable.

First of all, consider a "double" non-relational abstraction for sets of sets of memories, since we do not need to track relations between *different* variables. In classic static analysis, a non-relational abstraction for memories abstracts each program variable independently, thus forgetting any relation between variables. We can apply this concept to any functional space, meaning that given a set of functions we are not interested in the relations between objects and their images through the functions, as we do in the Non-Relational Abstraction

of Chapter 2.3. The Non-Relational Abstraction of a set of memories is an abstract domain of $\wp(\mathsf{Mem})$. We can can lift this abstraction to sets of sets, as shown in Section 6.2. The inner abstraction is $\alpha_{nr} \in \wp(\mathsf{Mem}) \to \dot{\mathsf{Mem}}$, where we recall that $\dot{\mathsf{Mem}} \triangleq \mathsf{Var} \to \wp(\mathbb{Z})$. Then, we lift this latter to sets obtaining the Galois insertion:

$$\langle \wp(\wp(\mathsf{Mem})), \subseteq \rangle \underset{\mathcal{L}(\alpha_{nr})}{\overset{\mathcal{G}(\alpha_{nr})}{\leftrightarrows}} \langle \wp(\dot{\mathsf{Mem}}), \subseteq \rangle$$

Now we can apply again the Non-Relational Abstraction of Section 2.3 to $\wp(\dot{\mathsf{Mem}})$, namely we have the Galois insertion, where $\ddot{\mathsf{Mem}} \triangleq \mathsf{Var} \to \wp(\wp(\mathbb{Z}))$:

$$\langle \wp(\dot{\mathsf{Mem}}), \subseteq \rangle \underset{\dot{\alpha}_{nr}}{\overset{\dot{\gamma}_{nr}}{\leftrightarrows}} \langle \ddot{\mathsf{Mem}}, \ddot{\subseteq} \rangle$$
$$\dot{\alpha}_{nr}(X) \triangleq \lambda \mathsf{x} \,.\, \{\dot{\mathsf{m}}(x) \mid \dot{\mathsf{m}} \in X\} \qquad \dot{\gamma}_{nr}(\ddot{\mathsf{m}}) \triangleq \{\dot{\mathsf{m}} \mid \forall \mathsf{x} \in \mathsf{Var} \,.\, \dot{\mathsf{m}}(\mathsf{x}) \in \ddot{\mathsf{m}}(\mathsf{x})\}$$

By composition, we have the abstraction between $\wp(\wp(\mathsf{Mem}))$ and $\ddot{\mathsf{Mem}}$, formalized by the Galois insertion:

$$\langle \wp(\wp(\mathsf{Mem})), \subseteq \rangle \underset{\alpha_{nnr}}{\overset{\gamma_{nnr}}{\leftrightarrows}} \langle \ddot{\mathsf{Mem}}, \dot{\subseteq} \rangle$$
$$\alpha_{nnr}(\mathcal{X}) \triangleq \dot{\alpha}_{nr} \circ \mathcal{L}(\alpha_{nr})(\mathcal{X}) = \lambda \mathsf{x} \,.\, \{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X\} \mid X \in \mathcal{X}\}$$
$$\gamma_{nnr}(\ddot{\mathsf{m}}) \triangleq \mathcal{G}(\alpha_{nr}) \circ \dot{\gamma}_{nr}(\ddot{\mathsf{m}}) = \{X \mid \forall \mathsf{x} \in \mathsf{Var} \,.\, \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X\} \in \ddot{\mathsf{m}}(\mathsf{x})\}$$

Applying the Pointwise Construction introduced in Section 2.3, we obtain the complete lattice $\langle \mathsf{Var} \to \mathtt{C}^\rho, \dot{\trianglelefteq}, \dot{\vee}, \dot{\wedge}, \lambda \mathsf{x} \,.\, \bot, \lambda \mathsf{x} \,.\, \top \rangle$. Then, applying the Pointwise Abstraction of Section 2.3, we have the following Galois connection:

$$\langle \ddot{\mathsf{Mem}}, \dot{\subseteq} \rangle \underset{\dot{\alpha}_{h^\rho}}{\overset{\dot{\gamma}_{h^\rho}}{\leftrightarrows}} \langle \mathsf{Var} \to \mathtt{C}^\rho, \dot{\trianglelefteq} \rangle$$
$$\dot{\alpha}_{h^\rho}(\ddot{\mathsf{m}}) \triangleq \lambda \mathsf{x} \,.\, \alpha_{h^\rho} \circ \ddot{\mathsf{m}}(\mathsf{x}) \qquad \dot{\gamma}_{h^\rho}(\mathfrak{m}) \triangleq \lambda \mathsf{x} \,.\, \gamma_{h^\rho} \circ \mathfrak{m}(\mathsf{x})$$

Finally, by composition, we obtain the Galois connection $(\wp(\wp(\mathsf{Mem})), \alpha_{\mathfrak{m}}, \gamma_{\mathfrak{m}}, \mathsf{Var} \to \mathtt{C}^\rho)$ where $\alpha_{\mathfrak{m}} \triangleq \dot{\alpha}_{h^\rho} \circ \alpha_{nnr}$ and $\gamma_{\mathfrak{m}} \triangleq \gamma_{nnr} \circ \dot{\gamma}_{h^\rho}$. We denote with Mem^ρ the set $\mathsf{Var} \to \mathtt{C}^\rho$ and we call its elements $\mathfrak{m}$ *abstract memories*. In order to simplify the notation, we let $\sqsubseteq \triangleq \dot{\trianglelefteq}$, $\sqcup \triangleq \dot{\vee}$, $\sqcap \triangleq \dot{\wedge}$, $\mathfrak{m}_\bot \triangleq \lambda \mathsf{x} \,.\, \bot$ and $\mathfrak{m}_\top \triangleq \lambda \mathsf{x} \,.\, \top$.

Example 24. Let us show how this abstraction works. Let $\mathsf{m}_a = [\mathsf{x} \mapsto 0 \; \mathsf{y} \mapsto 0], \mathsf{m}_b = [\mathsf{x} \mapsto 0 \; \mathsf{y} \mapsto 1], \mathsf{m}_c = [\mathsf{x} \mapsto 1 \; \mathsf{y} \mapsto 0], \mathsf{m}_d = [\mathsf{x} \mapsto 1 \; \mathsf{y} \mapsto 1]$ programs memories. Then we have: $\alpha_{\mathfrak{m}}(\{\{\mathsf{m}_a, \mathsf{m}_b\}, \{\mathsf{m}_c, \mathsf{m}_d\}\}) = [\mathsf{x} \mapsto \alpha_{h^\rho}(\{\{0\}, \{1\}\}) \; \mathsf{y} \mapsto \alpha_{h^\rho}(\{\{0, 1\}\})] = [\mathsf{x} \mapsto \mathcal{A}^\rho \; \mathsf{y} \mapsto \top]$. Indeed $\mathsf{m}_a, \mathsf{m}_b$ both assign 0 to x and $\mathsf{m}_c, \mathsf{m}_d$ both provide 1 to x, while $\mathsf{m}_a, \mathsf{m}_b$ provide both 0 and 1 to y and $\mathsf{m}_c, \mathsf{m}_d$ provide both 0 and 1 to y.

It is worth noting that, given $\mathcal{X} \subseteq \mathtt{equiv}^\rho_\mathsf{L}$, then every set in $\mathcal{X}$ contains L-equivalent memories, modulo ρ. This implies $\alpha_{\mathfrak{m}}(\mathcal{X})(\mathsf{x}) \trianglelefteq \mathcal{A}^\rho$, for each L variable x. So, the Abstract Non-Interference check $(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}^2_{\rho,\phi} \subseteq \mathtt{equiv}^\rho_\mathsf{L}$ becomes equivalent to checking, for each L variable x, whether $\alpha_{\mathfrak{m}}((\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}^2_{\rho,\phi})(\mathsf{x}) \trianglelefteq \mathcal{A}^\rho$ holds or not.

Finally, we can show how the abstract domain Mem^ρ can be used for Abstract Non-Interference verification, with the following theorem.

Theorem 23. *Let $\mathfrak{m}^\rho \in \mathsf{Mem}^\rho$ be the abstract memory defined as $\mathfrak{m}^\rho(\mathsf{x}) \triangleq \mathcal{A}^\rho$ if $\Gamma(\mathsf{x}) = \mathsf{L}$ and $\mathfrak{m}^\rho(\mathsf{x}) \triangleq \top$ otherwise. Then $\mathsf{P} \models \mathsf{ANI}^\rho_\phi$ if and only if $\alpha_{\mathfrak{m}}((\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}^2_{\rho,\phi}) \sqsubseteq \mathfrak{m}^\rho$.*

Proof. By definition of Galois connection: $\alpha_{\mathfrak{m}}(\{\!|\mathsf{P}|\!\}^{\curvearrowright}\mathcal{I}^2_{\rho,\phi}) \sqsubseteq \mathfrak{m}^\rho$ if and only if $\{\!|\mathsf{P}|\!\}^{\curvearrowright}\mathcal{I}^2_{\rho,\phi} \subseteq \gamma_{\mathfrak{m}}(\mathfrak{m}^\rho)$. It is clear that $\gamma_{\mathfrak{m}}(\mathfrak{m}^\rho) \subseteq \mathtt{equiv}^\rho_{\mathsf{L}}$, since the concretization of $\mathfrak{m}^\rho$ contains L-equivalent memories, modulo ρ, by construction. Hence, due to Prop. 14, the theorem is proved. □

Theorem 23 does not dispense us from computing the concrete hypersemantics hence, as usual in abstract interpretation, we define an abstract semantics computing on the abstract domain, i.e. a function in $\mathsf{Mem}^\rho \to \mathsf{Mem}^\rho$. Unfortunately, the constraint to have the same domain in input and output (of the abstract semantics) and the fact that our abstract domain maps to τ every H variable, do not allow us to verify ANI with declassification. The abstract semantics presented in the next subsection cannot keep in consideration the declassification ϕ. Indeed, it verifies the hyperproperty ANI^ρ_τ (without declassification), which implies anyway ANI^ρ_ϕ, for any possible ϕ [Giacobazzi and Mastroeni, 2018]. Clearly this is a limitation, we planned to extend the presented approach, in order to deal with declassification, as a future work.

8.1.3 The Parametric Abstract Semantics

Now we have to show how to compute a program's hypersemantics, parametric on the observation level ρ, on the proposed abstract domain. The abstract semantics for programs relies on the abstract semantics for boolean and arithmetic expressions, given in Figure 8.1. The abstract semantics for arithmetic expressions $\{\!|\mathsf{a}|\!\}^{\curvearrowright}_\rho \in \mathsf{Mem}^\rho \to \mathtt{C}^\rho$ evaluates to an abstract value and it relies on the abstract mathematical operations given in Figure 8.1. This semantics must be such that abstract assignments are sound approximations of the concrete ones. Since we are in a (double) non-relational setting the soundness requirement is: $\{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \subseteq \gamma_{h^\rho}\{\!|\mathsf{a}|\!\}^{\curvearrowright}_\rho\mathfrak{m}$. The proof of soundness for arithmetic expressions can be found in Appendix A. This latter assumes that abstract mathematical operations $\oplus^\rho \in \mathtt{C}^\rho \times \mathtt{C}^\rho \to \mathtt{C}^\rho$ are defined such that they are sound w.r.t. the concrete ones $\oplus$, with $\oplus \in \{+, -, *\}$. This technically means that they must satisfy the following constraint[3]:

$$\{\{n \oplus m \mid n \in X \wedge m \in Y\} \mid X \in \gamma_{h^\rho}(c_1) \wedge Y \in \gamma_{h^\rho}(c_2)\} \subseteq \gamma_{h^\rho}(c_1 \oplus^\rho c_2)$$

The constraint basically requires that every possible result obtained applying the concrete operation is contained in the concretization of the application of the abstract operator. We cannot give further details since the abstract operations depend on the chosen abstraction ρ. In Section 8.2 we will show a more detailed example, when we explain the abstract semantics for Non-Interference.

The abstract semantics for boolean expressions $\{\!|\mathsf{b}|\!\}^{\curvearrowright}_\rho \in \mathsf{Mem}_\rho \to \mathsf{Mem}^\rho$ is an abstract filtering function, and it relies on the abstract logical operations given in Fig. 8.1. To simplify, we assume that all negations $\neg$ have been removed using De Morgan's laws and usual arithmetic laws: $\neg(\mathsf{b}_1 \vee \mathsf{b}_2) \equiv (\neg\mathsf{b}_1) \wedge (\neg\mathsf{b}_2)$, $\neg(\mathsf{a}_1 < \mathsf{a}_2) \equiv (\mathsf{a}_2 \leq \mathsf{a}_1)$, etc. This semantics must be sound w.r.t. the concrete hypersemantics for booleans, namely $\{\{\mathsf{m} \in X \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} = \{\!|\mathsf{b}|\!\}^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}\{\!|\mathsf{b}|\!\}^{\curvearrowright}_\rho\mathfrak{m}$. The proof of soundness for boolean expressions can be found in Appendix A.

[3] Recall that our abstract semantics is built after the double non-relational abstraction α_{nnr}, hence it abstracts $\ddot{\mathsf{Mem}} = \mathsf{Var} \to \wp(\wp(\mathbb{Z}))$.

Arithmetic expressions: $(\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho} \in \mathsf{Mem}^{\rho} \to \mathtt{C}^{\rho}$ with $\oplus \in \{+, -, *\}$

$$(\!|\mathsf{n}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq \alpha_{h^{\rho}}(\{\{n\}\}) \qquad (\!|\mathsf{x}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq \mathfrak{m}(\mathsf{x}) \qquad (\!|(\mathsf{a})|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq (\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$$

$$(\!|\mathsf{a}_1 \oplus \mathsf{a}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq (\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \oplus^{\rho} (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$$

Boolean expressions: $(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho} \in \mathsf{Mem}^{\rho} \to \mathsf{Mem}^{\rho}$ with $\bowtie \in \{=, \neq, <, \leq\}$

$$(\!|\mathtt{tt}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq \mathfrak{m} \qquad (\!|\mathtt{ff}|\!)^{\curvearrowright}_{\rho} \triangleq \bot^{\natural} \qquad (\!|(\mathsf{b})|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq (\!|\mathsf{b}|\!)^{\sharp}\mathfrak{m} \qquad (\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq (\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \sqcap (\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$$

$$(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq \begin{array}{l}\text{let } \mathfrak{n} = (\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \sqcup (\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \text{ in} \\ \lambda \mathsf{x}\,.\, (\![\, \mathfrak{m}(\mathsf{x}) = \top \wedge \mathsf{x} \in \mathsf{vars}(\mathsf{b}_1) \cup \mathsf{vars}(\mathsf{b}_2) \,?\, \top : \mathfrak{n}(\mathsf{x}) \,]\!)\end{array}$$

$$(\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq \begin{array}{l}\text{let } \langle c_1, c_2\rangle = (\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \bowtie^{\rho} (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \text{ in} \\ \bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{n} \trianglelefteq c_1\} \sqcap \bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{n} \trianglelefteq c_2\}\end{array}$$

Figure 8.1: Abstract semantics for expressions.

Remark. In this case we cannot rely, as in the standard case of classic program verification for trace properties, on the identity function as a sound approximation: $(\!|\mathsf{b}|\!)^{\curvearrowright}$ is not reductive. The only trivial correct approximation is $\lambda \mathsf{x}\,.\,\top$.

In order to obtain a sound semantics, we need to define the abstract comparators $\bowtie^{\rho} \in \mathtt{C}^{\rho} \times \mathtt{C}^{\rho} \to \mathtt{C}^{\rho} \times \mathtt{C}^{\rho}$ such that they are sound w.r.t. the concrete ones $\bowtie$, with $\bowtie \in \{=, \neq, <, \leq\}$. This technically means that they must satisfy the following constraint:

$$\begin{array}{l}\text{let } \mathcal{X} = \{\{\langle n, m\rangle \mid n \in X \wedge m \in Y \wedge n \bowtie m\} \mid X \in \gamma_{h^{\rho}}(c_1) \wedge Y \in \gamma_{h^{\rho}}(c_2)\} \text{ in} \\ \langle\{\{n \mid \langle n, m\rangle \in X\} \mid X \in \mathcal{X}\}, \{\{m \mid \langle n, m\rangle \in X\} \mid X \in \mathcal{X}\}\rangle \subseteq^2 \gamma_{h^{\rho}}(c_1 \bowtie^{\rho} c_2)\end{array}$$

The constraint requires that every possible pair of values making true the concrete comparator is contained in the concretization of the application of the abstract comparator. In Figure 8.1, the memory $\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}\mathfrak{n} \trianglelefteq c\}$ can be approximated with a *backward* abstract semantics for arithmetic expressions ${}^{\mathcal{B}}(\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho} \in \mathsf{Mem}^{\rho} \to (\mathtt{C}^{\rho} \to \mathsf{Mem}^{\rho})$. The meaning of ${}^{\mathcal{B}}(\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}(\mathfrak{m})(c) = \mathfrak{m}'$ is that $\mathfrak{m}'$ is a refinement of $\mathfrak{m}$, i.e. $\mathfrak{m}' \sqsubseteq \mathfrak{m}$, such that $(\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}' \trianglelefteq c$.

Finally, we need two auxiliary functions $\mathsf{vars}^{\top}_{\mathfrak{m}}(\mathsf{b})$ and $\mathsf{vars}^{:=}(\mathsf{P})$, returning the set of variables occurring in b having value $\top$ when evaluated in $\mathfrak{m}$ and the set of *modified variables* in P, respectively. The first is straightforward to compute: $\mathsf{vars}^{\top}_{\mathfrak{m}}(\mathsf{b}) \triangleq \{\mathsf{x} \in \mathsf{vars}(\mathsf{b}) \mid \mathfrak{m}(\mathsf{x}) = \top\}$ and $\mathsf{vars}(\mathsf{b})$ is just a syntactic check. The second involves semantic information, hence it is not trivial to compute. Naively, we can use a simple syntactic approach for approximating the set of variables which may be modified during P executions. Indeed, the function $\mathsf{vars}^{:=}(\mathsf{P})$ returns the set of variables occurring in P on the left-hand side of an assignment, which is easy implementable as a syntactic check. Now we have all the ingredients needed

to define the abstract hypersemantics $(\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho} \in \mathsf{Mem}^{\rho} \to \mathsf{Mem}^{\rho}$, which is[4]:

$$(\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}_{\bot} \triangleq \mathfrak{m}_{\bot} \qquad (\!|{}^{\mathtt{i}}\mathsf{c}_1{}^{\mathtt{l}} \,.\, {}^{\mathtt{k}}\mathsf{c}_2{}^{\mathtt{f}}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq (\!|{}^{\mathtt{k}}\mathsf{c}_2{}^{\mathtt{f}}|\!)^{\curvearrowright}_{\rho}(\!|{}^{\mathtt{i}}\mathsf{c}_1{}^{\mathtt{l}}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \qquad (\!|{}^{\mathtt{i}}\mathsf{skip}^{\mathtt{f}}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq \mathfrak{m}$$

$$(\!|{}^{\mathtt{i}}\mathsf{x} := \mathsf{a}^{\mathtt{f}}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq \mathfrak{m}[\mathsf{x} \leftarrow\!\!\shortmid (\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}]$$

$$(\!|{}^{\mathtt{i}}\mathsf{if\ b\ then\ \{\,P_1\,\}\ else\ \{\,P_2\,\}}^{\mathtt{f}}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq \mathfrak{w} \text{ where}$$
$$\text{let } \mathfrak{n} = (\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\rho}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \sqcup (\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\rho}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \text{ in}$$
$$\mathfrak{w} = \lambda\mathsf{x}\,.\begin{cases}\mathfrak{n}(\mathsf{x}) & \text{if } \mathsf{x} \notin \mathsf{vars}^{:=}(\mathsf{P}_1) \cup \mathsf{vars}^{:=}(\mathsf{P}_2) \vee \mathfrak{n}(\mathsf{x}) \lneq \mathcal{A}^{\rho} \vee \mathsf{vars}^{\top}_{\mathfrak{m}}(\mathsf{b}) = \varnothing \\ \top & \text{otherwise}\end{cases}$$

$$(\!|{}^{\mathtt{i}}\mathsf{while}\ {}^{\mathtt{k}}\mathsf{b}\ \{\,\mathsf{P}\,\}^{\mathtt{f}}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \triangleq (\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\rho}\big(\mathsf{lfp}^{\sqsubseteq}_{\mathfrak{m}_{\bot}}\ \lambda\mathfrak{n}\,.\,\mathfrak{m} \sqcup (\!|{}^{\mathtt{i}}\mathsf{if\ b\ then\ \{\,P\,\}\ else\ \{\ }^{\mathtt{h}}\mathsf{skip}^{\mathtt{h}}\ \}^{\mathtt{f}}|\!)^{\curvearrowright}_{\rho}\mathfrak{n}\big)$$

with $^{\mathtt{h}}$ fresh label (it is indeed not necessary, since the post-conditions hypersemantics does not take into account labels). The abstract semantics is quite standard for all statements, except for conditionals. We will explain here only this latter, which exploits the following idea. For every variable, we compute the join between its value resulting after the execution of the true branch and its value resulting after the execution of the false branch. This is done in order to track the forbidden flows (implicit or explicit) generated inside the two branches. In fact, a L variable has value $\top$ after the join if in at least one of the branches it has value $\top$ (meaning that there is a forbidden flow). After this check we need to take in consideration the implicit flows generated by the conditional statement itself. Indeed, first we suppose that if there is at least one variable with value $\top$ *before* the boolean guard is evaluated, then all variables modified in the conditional branches have a forbidden flow (a variable has value $\top$ only if it is a H variable or if it has been "influenced" by a H variable). This is done setting to $\top$ all modified variables. Note that if, for some reasons, a H variable is not $\top$ during this check, the flow is correctly not set. This procedure is sound but not so precise. In order to enhance precision, we exploit our abstract domain. In particular, we do not set to $\top$ the variables which have the same constant value $\bar{\mathfrak{a}}$ in both branches (this is the condition $\mathfrak{n}(\mathsf{x}) \lneq \mathcal{A}^{\rho}$) because this means that at the end of the conditional statement the variable has always a constant value, modulo ρ. If there are no $\top$-valued variables into the guard b (this is the condition $\mathsf{vars}^{\top}_{\mathfrak{m}}(\mathsf{b}) = \varnothing$), then no variables are set to $\top$: the resulting flows are those generated into the two branches of the conditional.

We can prove that our abstract semantics (parametric on the observation ρ) is sound w.r.t. the concrete hypersemantics, and that it can be used for Abstract Non-Interference verification. This is stated in the following two theorems.

Theorem 24 (Soundness)**.** *The abstract hypersemantics is sound w.r.t. the concrete hypersemantics: for every* $\mathfrak{m} \in \mathsf{Mem}^{\rho}$ *we have* $(\!|\mathsf{P}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}(\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$.

Sketch. We just have to prove that the abstract hypersemantics approximates the best correct approximation of the concrete hypersemantics in Mem^{ρ}, namely $\alpha_{\mathfrak{m}} \circ (\!|\mathsf{P}|\!)^{\curvearrowright} \circ \gamma_{\mathfrak{m}} \mathrel{\dot{\sqsubseteq}} (\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho}$. The proof relies on the soundness of the abstract semantics for arithmetic and boolean expressions and it is for structural induction. We omit here the full proof, we just show, as

[4] In the definition of the conditional statement, $\lneq$ denotes the strict version of $\trianglelefteq$.

example, the case for programs sequences (the full proof can be found in Appendix A).

$$\begin{aligned}
\alpha_{\mathfrak{m}}(\!|\, {}^{\underline{i}}\mathsf{c}_1{}^{\underline{l}} \fatsemi {}^{\underline{k}}\mathsf{c}_2{}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) &= \alpha_{\mathfrak{m}}(\!|\, {}^{\underline{k}}\mathsf{c}_2{}^{\underline{f}} \,|\!)^{\curvearrowright}(\!|\, {}^{\underline{i}}\mathsf{c}_1{}^{\underline{l}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) && \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright} \\
&\sqsubseteq \alpha_{\mathfrak{m}}(\!|\, {}^{\underline{k}}\mathsf{c}_2{}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}\alpha_{\mathfrak{m}}(\!|\, {}^{\underline{i}}\mathsf{c}_1{}^{\underline{l}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) && \| \text{ extensivity of } \gamma_{\mathfrak{m}}\alpha_{\mathfrak{m}} \\
&\sqsubseteq (\!|\, {}^{\underline{k}}\mathsf{c}_2{}^{\underline{f}} \,|\!)^{\curvearrowright}_{\rho}(\!|\, {}^{\underline{i}}\mathsf{c}_1{}^{\underline{l}} \,|\!)^{\curvearrowright}_{\rho}(\mathfrak{m}) && \| \text{ inductive hypothesis} \\
&= (\!|\, {}^{\underline{i}}\mathsf{c}_1{}^{\underline{l}} \fatsemi {}^{\underline{k}}\mathsf{c}_2{}^{\underline{f}} \,|\!)^{\curvearrowright}_{\rho}(\mathfrak{m}) && \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright}_{\rho}
\end{aligned}$$

□

With the abstract semantics we can define an effective verification method for Abstract Non-Interference $\mathsf{ANI}^{\rho}_{\tau}$. Recall that the classic NI coincides with $\mathsf{ANI}^{\iota}_{\tau}$, meaning that our verification method can be used to approximate also Non-Interference (we will see in detail the abstract hypersemantics for NI in the next section).

Theorem 25 (ANI Verification)**.** *We have that* $\mathsf{P} \models \mathsf{ANI}^{\rho}_{\tau}$ *if* $(\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}^{\rho} \sqsubseteq \mathfrak{m}^{\rho}$.

Proof. Note that $\alpha_{\mathfrak{m}}(\mathcal{I}^2_{\rho,\tau}) \sqsubseteq \mathfrak{m}^{\rho}$ since $\mathcal{I}^2_{\rho,\tau}$ contains only sets of memories agreeing on L variables, modulo ρ. This means that $\alpha_{\mathfrak{m}}(\mathcal{I}^2_{\rho,\tau})(\mathsf{x}) \trianglelefteq \mathcal{A}^{\rho} = \mathfrak{m}^{\rho}(\mathsf{x})$, for each L variable x. For H variables y, $\alpha_{\mathfrak{m}}(\mathcal{I}^2_{\rho,\tau})(\mathsf{y}) \trianglelefteq \top = \mathfrak{m}^{\rho}(\mathsf{y})$ trivially holds. Then the proof is given by the following chain of implications.

$$\begin{array}{l}
(\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}^{\rho} \sqsubseteq \mathfrak{m}^{\rho} \\
\Downarrow \quad \| \text{ monotonicity of } (\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho} \text{and } \alpha_{\mathfrak{m}}(\mathcal{I}^2_{\rho,\tau}) \sqsubseteq \mathfrak{m}^{\rho} \\
(\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho}\alpha_{\mathfrak{m}}(\mathcal{I}^2_{\rho,\tau}) \sqsubseteq (\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}^{\rho} \sqsubseteq \mathfrak{m}^{\rho} \\
\Downarrow \quad \| \text{ soundness of } (\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho} \text{(Theorem 24) and } \alpha_{\mathfrak{m}}, \gamma_{\mathfrak{m}} \text{ adjunction: } \forall \mathcal{X} \in \wp(\wp(\mathsf{Mem})) \,.\, \alpha_{\mathfrak{m}}(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{X} \sqsubseteq (\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho}\alpha_{\mathfrak{m}}(\mathcal{X}) \\
\alpha_{\mathfrak{m}}((\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}^2_{\rho,\tau}) \sqsubseteq (\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho}\alpha_{\mathfrak{m}}(\mathcal{I}^2_{\rho,\tau}) \sqsubseteq (\!|\mathsf{P}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}^{\rho} \sqsubseteq \mathfrak{m}^{\rho} \\
\Downarrow \quad \| \text{ Theorem 23} \\
\mathsf{P} \models \mathsf{ANI}^{\rho}_{\tau}
\end{array}$$

□

This means that we can check Abstract Non-Interference simply by checking that each set of computations, starting from L-equivalent memories, modulo ρ, provides only results indistinguishable by ρ.

Termination and Precision. The finite height of the hyperdomain C^{ρ} guarantees the termination of the analysis and the structure of the domain allows us to compute loops fixpoints quickly. Hence there is no need for a widening operator in order to speed-up the analysis or to force termination.

For what concerns precision, we tested our abstract semantics schemata, defining an analyzer for $\mathsf{ANI}^{\mathsf{Sgn}}_{\tau}$. This latter is quite precise in general but, unfortunately, it raises false alarms in some trivial situations. For instance is not able to say that the program $\mathsf{P} = \mathsf{a} := \mathsf{x} - \mathsf{x}$ does not leak sensitive information when a is public and x is private. The main source of imprecision of our semantics schemata is the lack of relational information between variables. Indeed, our analysis is not-relational, meaning that we do not explicitly track relations

between *different* variables. We can increase the precision pairing Mem^{ρ} with a relational abstraction of $\wp(\wp(\mathsf{Mem}))$. For instance we can define an abstract domain tracking equalities between variables. This latter, combined with a numerical domain such as the one for intervals will improve the precision w.r.t. implicit flows and with programs like P introduced above. We have to mention that our choice to approximate the set of modified variables in a very simple syntactic way surely increases the number of false alarms. With a more semantic analysis we can gain a lot of precision. We will discuss more about precision at the end of the next section, after introducing the abstract hypersemantics for Non-Interference.

8.2 Implementation: the NonInterfer Static Analyzer

In this section we apply the results about ANI verification in order to verify the classic Non-Interference. Basically, we instantiate the abstract semantics with $\rho = \iota$ and $\phi = \tau$, indeed $\mathsf{NI} = \mathsf{ANI}^{\iota}_{\tau}$. The collecting hypersemantics is the same as the one proposed in the previous section, hence we give only the abstract version for the verification of Non-Interference. We instantiate again the hyperlevel constants domain to Non-Interference. The latter, in its original formulation, is not machine-representable, namely it has an uncountable set of elements. Moreover, it contains infinite ascending chains (i.e. it is not ACC), inducing potential computation divergence. Hence, we approximate it in order to make its implementation feasible. Finally we give the definition of the abstract hypersemantics.

We have implemented the abstract hypersemantics as a verifier for Non-Interference, called nonInterfer.

Analogously to the ANI case, we can verify Non-Interference in a simpler domain, namely we can collect states instead of input/output traces of states. We can partition NI in two simpler hyperproperties:

$$\mathsf{NI}^{=} \triangleq \bigcup_{\mathtt{i},\mathtt{f}\in\mathsf{Lab}} \left\{ X \subseteq \Sigma\times\Sigma \;\middle|\; \begin{array}{l} X = \{\langle\langle \mathtt{i}, \mathsf{m}_1\rangle,\langle \mathtt{f}, \mathsf{m}'_1\rangle\rangle, \langle\langle \mathtt{i}, \mathsf{m}_2\rangle,\langle \mathtt{f}, \mathsf{m}'_2\rangle\rangle\} \wedge \\ \nu^{\iota\times\tau}(\mathsf{m}_1) = \nu^{\iota\times\tau}(\mathsf{m}_2) \wedge \nu^{\iota\times\tau}(\mathsf{m}'_1) = \nu^{\iota\times\tau}(\mathsf{m}'_2) \end{array} \right\}$$

$$\mathsf{NI}^{\neq} \triangleq \bigcup_{\mathtt{i},\mathtt{f}\in\mathsf{Lab}} \left\{ X \subseteq \Sigma\times\Sigma \;\middle|\; \begin{array}{l} X = \{\langle\langle \mathtt{i}, \mathsf{m}_1\rangle,\langle \mathtt{f}, \mathsf{m}'_1\rangle\rangle, \langle\langle \mathtt{i}, \mathsf{m}_2\rangle,\langle \mathtt{f}, \mathsf{m}'_2\rangle\rangle\} \wedge \\ \nu^{\iota\times\tau}(\mathsf{m}_1) \neq \nu^{\iota\times\tau}(\mathsf{m}_2) \end{array} \right\}$$

Then we can skip the check for $\mathsf{NI}^{\neq}$ since it is always satisfied, by every program. Furthermore, we can note that $\mathsf{NI}^{=}$ is a second-order not-relational hyperproperty, meaning that it is equivalent to the pair $\langle \mathcal{X}, \mathcal{Y}\rangle$, where:

$$\mathcal{X} \triangleq \bigcup_{\mathtt{i}\in\mathsf{Lab}} \{X \subseteq \Sigma \mid X = \{\langle \mathtt{i}, \mathsf{m}_1\rangle, \langle \mathtt{i}, \mathsf{m}_2\rangle\} \wedge \nu^{\iota\times\tau}(\mathsf{m}_1) = \nu^{\iota\times\tau}(\mathsf{m}_2)\}$$

$$\mathcal{Y} \triangleq \bigcup_{\mathtt{f}\in\mathsf{Lab}} \{X \subseteq \Sigma \mid X = \{\langle \mathtt{f}, \mathsf{m}'_1\rangle, \langle \mathtt{f}, \mathsf{m}'_2\rangle\} \wedge \nu^{\iota\times\tau}(\mathsf{m}'_1) = \nu^{\iota\times\tau}(\mathsf{m}'_2)\}$$

In the definition above, control points are totally irrelevant, indeed we have that $\langle \mathcal{X}, \mathcal{Y}\rangle$ is isomorphic to $\langle \mathcal{X}', \mathcal{Y}'\rangle$, where:

$$\mathcal{X}' \triangleq \{X \subseteq \mathsf{Mem} \mid X = \{\mathsf{m}_1, \mathsf{m}_2\} \wedge \nu^{\iota\times\tau}(\mathsf{m}_1) = \nu^{\iota\times\tau}(\mathsf{m}_2)\}$$

$$\mathcal{Y}' \triangleq \{X \subseteq \mathsf{Mem} \mid X = \{\mathsf{m}'_1, \mathsf{m}'_2\} \wedge \nu^{\iota\times\tau}(\mathsf{m}'_1) = \nu^{\iota\times\tau}(\mathsf{m}'_2)\}$$

This basically means that it suffices to check whether $[\![\mathsf{P}]\!]^{\curvearrowright} X \in \mathcal{Y}'$, for every $X \in \mathcal{X}'$. We can verify this at once using a post-conditions hypersemantics $(\!|\mathsf{P}|\!)^{\curvearrowright}$, namely checking whether $(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{X}' = \mathcal{Y}'$. This observations induce the following proposition.

Proposition 15. $\mathsf{P} \models \mathsf{NI}$ *if and only if* $(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}^2_{\mathtt{L}} \subseteq \mathtt{equiv}_{\mathtt{L}}$, *where:*

$$\mathcal{I}^2_{\mathtt{L}} \triangleq \{X \subseteq \mathsf{Mem} \mid X = \{\mathsf{m}, \mathsf{m}'\} \wedge \nu^{\iota\times\tau}(\mathsf{m}) = \nu^{\iota\times\tau}(\mathsf{m})\}$$
$$\mathtt{equiv}_{\mathtt{L}} \triangleq \{X \subseteq \mathsf{Mem} \mid \forall \mathsf{m}, \mathsf{m}' \in X \,.\, \nu^{\iota\times\tau}(\mathsf{m}) = \nu^{\iota\times\tau}(\mathsf{m}')\}$$

As we have seen in Chapter 7, we can have different correct post-conditions hypersemantics, depending on how we define the semantics for loops. We chose to adopt again the one defined in Subsection 8.1.1.

8.2.1 Towards Abstraction

Unfortunately, $(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}^2_{\mathtt{L}}$ and $\mathtt{equiv}_{\mathtt{L}}$ are not computable in general, hence we need approximations. In order to compute a sound approximation of $(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}^2_{\mathtt{L}}$, we rely on abstract interpretation. The first step is to define the abstract domain used to verify Non-Interference.

8.2.1.1 The Abstract Domain for Non-Interference

The domain is an instance of the *hyperlevel* (*abstract*) *constants* (Section 6.2), and it checks whether a set of sets of values contains constant sets. In the following, all abstraction functions α are additive (i.e. they preserve the least upper bound of chains), hence their left adjoint α^- always exists.

The domain is an abstraction of $\wp(\wp(\mathbb{Z}))$. In this case, we use as inner abstraction the one for the classic constant propagation domain: it represents precisely the singletons $\{n\}$, for every $n \in \mathbb{Z}$, and it abstracts to $\mathbb{Z}$ everything else. We have introduced this latter domain in Example 23, with the upper closure operator $\mathtt{Cprp}$. The outer abstraction checks whether the inner abstraction always returns constant values, not necessarily the same. For instance, $\{\{1\},\{2\}\}$ contains only constant sets, instead $\{\{1\},\{2,3\}\}$ contains also a not-constant set (in this case the inner abstraction maps $\{2,3\}$ to $\mathbb{Z}$).

Let $\mathtt{Cprp}_{\mathfrak{hc}} \triangleq \wp(\{\{n\} \mid n \in \mathbb{Z}\}) \cup \{\mathtt{Cprp}(\wp(\mathbb{Z}))\}$; $\alpha_{\mathfrak{hc}}(\mathcal{X}) \triangleq \mathcal{X}$ if $\mathcal{X} \subseteq \{\{n\} \mid n \in \mathbb{Z}\}$ and $\alpha(\mathcal{X}) \triangleq \alpha\mathtt{Cprp}(\wp(\mathbb{Z}))$ otherwise; and $\gamma_{\mathfrak{hc}} \triangleq \alpha^-_{\mathfrak{hc}} = \mathsf{id}$. Then we have the Galois insertion:

$$\langle \wp(\wp(\mathbb{Z})), \subseteq\rangle \underset{\alpha_{\mathfrak{hc}}}{\overset{\gamma_{\mathfrak{hc}}}{\leftrightarrows}} \langle \mathtt{Cprp}_{\mathfrak{hc}}, \subseteq\rangle$$

This domain is sufficient for Non-Interference verification but, as already pointed out, it is not machine-representable. For this reason we need to perform a further approximation. Since $\mathtt{Cprp}_{\mathfrak{hc}}$ has an uncountable set of elements, we define a simpler domain, which is machine-representable but still able to verify Non-Interference. We can set $\overline{\mathrm{Atm}}^{\mathtt{Cprp}} \triangleq \mathbb{Z}$ as the set isomorphic to $\mathrm{Atm}^{\mathtt{Cprp}} = \{\{n\} \mid n \in \mathbb{Z}\}$, aiming at representing sets containing only one singleton, i.e. of the form $\{\{n\}\}$, which is the information we want to observe precisely, and let $^- \in \mathrm{Atm}^{\mathtt{Cprp}} \to \mathbb{Z}$ a bijection. Furthermore, we denote by $\kappa \triangleq \mathcal{A}^{\mathtt{Cprp}}$ the abstract element representing the set of all singletons (i.e. the atoms). Then we define $\mathtt{C}^{\mathtt{Cprp}} \triangleq \{\bar{n} \mid n \in \mathbb{Z}\} \cup \{\bot, \top, \kappa\}$, with the partial order $\trianglelefteq \subseteq \mathtt{C}^{\mathtt{Cprp}} \times \mathtt{C}^{\mathtt{Cprp}}$ defined as $c_1 \trianglelefteq c_2 \triangleq (c_1 = \bot \vee c_1 =$

$c_2 \vee (c_1 = \bar{n} \wedge c_2 = \kappa) \vee c_2 = \top$). Now consider the abstraction $\alpha_{\mathtt{C}^{\mathtt{Cprp}}} \in \mathtt{Cprp}_{\mathfrak{hc}} \to \mathtt{C}^{\mathtt{Cprp}}$ and the concretization $\gamma_{\mathtt{C}^{\mathtt{Cprp}}} \in \mathtt{C}^{\mathtt{Cprp}} \to \mathtt{Cprp}_{\mathfrak{hc}}$:

$$\alpha_{\mathtt{C}^{\mathtt{Cprp}}}(\mathcal{X}) \triangleq \begin{cases} \bot & \text{if } \mathcal{X} = \varnothing \\ \bar{n} & \text{if } \mathcal{X} = \{\{n\}\} \\ \kappa & \text{if } \mathcal{X} \subseteq \{\{n\} \mid n \in \mathbb{Z}\} \wedge |\mathcal{X}| > 1 \\ \top & \text{otherwise} \end{cases} \qquad \gamma_{\mathtt{C}^{\mathtt{Cprp}}}(c) \triangleq \begin{cases} \varnothing & \text{if } c = \bot \\ \{\{n\}\} & \text{if } c = \bar{n} \\ \{\{n\} \mid n \in \mathbb{Z}\} & \text{if } c = \kappa \\ \mathtt{Cprp}(\wp(\mathbb{Z})) & \text{otherwise} \end{cases}$$

We have the Galois insertion $(\mathtt{Cprp}_{\mathfrak{hc}}, \alpha_{\mathtt{C}^{\mathtt{Cprp}}}, \gamma_{\mathtt{C}^{\mathtt{Cprp}}}, \mathtt{C}^{\mathtt{Cprp}})$. The domain $\langle \mathtt{C}^{\mathtt{Cprp}}, \trianglelefteq, \veebar, \barwedge, \bot, \top \rangle$ is a compete lattice where $\veebar, \barwedge \in \mathtt{C}^{\mathtt{Cprp}} \times \mathtt{C}^{\mathtt{Cprp}} \to \mathtt{C}^{\mathtt{Cprp}}$ are:

$$c_1 \veebar c_2 \triangleq \begin{cases} \top & \text{if } c_1 = \top \vee c_2 = \top \\ \bot & \text{if } c_1 = \bot \wedge c_2 = \bot \\ \bar{n} & \text{if } (c_1 = \bot \wedge c_2 = \bar{n}) \vee (c_1 = \bar{n} \wedge c_2 = \bot) \vee (c_1 = \bar{n} \wedge c_2 = \bar{n}) \\ \kappa & \text{otherwise} \end{cases}$$

$$c_1 \barwedge c_2 \triangleq \begin{cases} \top & \text{if } c_1 = \top \wedge c_2 = \top \\ \bot & \text{if } c_1 = \bot \vee c_2 = \bot \vee (c_1 = \bar{n} \neq \bar{m} = c_2) \\ \bar{n} & \text{if } (c_1 = \bar{n} \wedge c_2 \in \{\kappa, \top\}) \vee (c_2 = \bar{n} \wedge c_1 \in \{\kappa, \top\}) \\ \kappa & \text{otherwise} \end{cases}$$

By composition, $\alpha_{h^{\mathtt{Cprp}}} \triangleq \alpha_{\mathtt{C}^{\mathtt{Cprp}}} \circ \alpha_{\mathfrak{hc}}$ and $\gamma_{h^{\mathtt{Cprp}}} \triangleq \gamma_{\mathfrak{hc}} \circ \gamma_{\mathtt{C}^{\mathtt{Cprp}}}$ form the Galois insertion:

$$\langle \wp(\wp(\mathbb{Z})), \subseteq \rangle \underset{\alpha_{h^{\mathtt{Cprp}}}}{\overset{\gamma_{h^{\mathtt{Cprp}}}}{\leftrightarrows}} \langle \mathtt{C}^{\mathtt{Cprp}}, \trianglelefteq \rangle$$

This domain approximates the set of sets of values a variable may have. Finally, in order to track information flows we need to work on memories instead of on values. Consider again the "double" non-relational abstraction α_{nnr} introduced in Subsection 8.1.2. Applying the Pointwise Construction introduced in Section 2.3, we obtain the complete lattice $\langle \mathsf{Var} \to \mathtt{C}^{\mathtt{Cprp}}, \dot{\trianglelefteq}, \dot{\veebar}, \dot{\barwedge}, \lambda \mathsf{x} \,.\, \bot, \lambda \mathsf{x} \,.\, \top \rangle$. We can compose point-wise the hyperlevel constants abstraction with α_{nnr}, obtaining $\alpha_{\mathfrak{m}} \triangleq \dot{\alpha}_{h^{\mathtt{Cprp}}} \circ \alpha_{nnr}$. This latter forms, paired with $\gamma_{\mathfrak{m}} \triangleq \alpha_{\mathfrak{m}}^{-} = \gamma_{nnr} \circ \dot{\gamma}_{h^{\mathtt{Cprp}}}$, the Galois connection:

$$\langle \wp(\wp(\mathsf{Mem})), \subseteq \rangle \underset{\alpha_{\mathfrak{m}}}{\overset{\gamma_{\mathfrak{m}}}{\leftrightarrows}} \langle \mathsf{Var} \to \mathtt{C}^{\mathtt{Cprp}}, \dot{\trianglelefteq} \rangle$$

We denote with $\mathsf{Mem}^{\mathtt{Cprp}}$ the set $\mathsf{Var} \to \mathtt{C}^{\mathtt{Cprp}}$ and we call its elements $\mathfrak{m}$ *abstract memories*. In order to simplify the notation, we let $\sqsubseteq \triangleq \dot{\trianglelefteq}$, $\sqcup \triangleq \dot{\veebar}$, $\sqcap \triangleq \dot{\barwedge}$, $\mathfrak{m}_{\bot} \triangleq \lambda \mathsf{x} \,.\, \bot$ and $\mathfrak{m}_{\top} \triangleq \lambda \mathsf{x} \,.\, \top$.

Example 25. Continuing the previous Example 24, we have that $\mathcal{I}_{\mathsf{L}}^2 = \{\{\mathsf{m}_a, \mathsf{m}_b\}, \{\mathsf{m}_c, \mathsf{m}_d\}\}$ and $\alpha_{\mathfrak{m}}(\{\{\mathsf{m}_a, \mathsf{m}_b\}, \{\mathsf{m}_c, \mathsf{m}_d\}\}) = [\mathsf{x} \mapsto \alpha_{h^{\mathtt{Cprp}}}(\{\{0\}, \{1\}\})\ \mathsf{y} \mapsto \alpha_{h^{\mathtt{Cprp}}}(\{\{0, 1\}\})] = [\mathsf{x} \mapsto \kappa\ \mathsf{y} \mapsto \top]$. Indeed $\mathsf{m}_a, \mathsf{m}_b$ both provide 0 to x and $\mathsf{m}_c, \mathsf{m}_d$ both provide 1 to x, while $\mathsf{m}_a, \mathsf{m}_b$ provide both 0 and 1 to y and $\mathsf{m}_c, \mathsf{m}_d$ provide both 0 and 1 to y. Similarly, $(\!|\mathsf{P}|\!)^{\frown}\mathcal{I}_{\mathsf{L}}^2 = \{\{\mathsf{m}_b, \mathsf{m}_d\}\}$ and $\alpha_{\mathfrak{m}}(\{\{\mathsf{m}_b, \mathsf{m}_d\}\}) = [\mathsf{x} \mapsto \alpha_{h^{\mathtt{Cprp}}}(\{\{0, 1\}\})\ \mathsf{y} \mapsto \alpha_{h^{\mathtt{Cprp}}}(\{\{1\}\})] = [\mathsf{x} \mapsto \top\ \mathsf{y} \mapsto \overline{1}]$.

It is worth noting that, given $\mathcal{X} \subseteq \mathtt{equiv}_{\mathsf{L}}$, then every set in $\mathcal{X}$ contains L-equivalent memories. This implies $\alpha_{\mathfrak{m}}(\mathcal{X})(\mathsf{x}) \trianglelefteq \kappa$, for each L variable x. So, the Non-Interference check

$(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}_{\mathsf{L}}^2 \subseteq \mathtt{equiv}_{\mathsf{L}}$ becomes equivalent to checking whether $\alpha_{\mathfrak{m}}((\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}_{\mathsf{L}}^2)(\mathsf{x}) \trianglelefteq \kappa$, for each L variable x. Indeed the program of Example 25 is interferent, since computing $\alpha_{\mathfrak{m}}((\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}_{\mathsf{L}}^2)$ results in L variable x having value $\top$.

Finally, we can show how the abstract domain $\mathsf{Mem}^{\mathsf{cprp}}$ can be used for Non-Interference verification, with the following theorem.

Theorem 26. *Let* $\mathfrak{m}^{\mathtt{NI}} \in \mathsf{Mem}^{\mathsf{cprp}}$ *be the abstract memory defined as* $\mathfrak{m}^{\mathtt{NI}}(\mathsf{x}) \triangleq \kappa$ *if* $\Gamma(\mathsf{x}) = \mathsf{L}$ *and* $\mathfrak{m}^{\mathtt{NI}}(\mathsf{x}) \triangleq \top$ *otherwise. Then* $\mathsf{P} \models \mathtt{NI}$ *if and only if* $\alpha_{\mathfrak{m}}((\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}_{\mathsf{L}}^2) \sqsubseteq \mathfrak{m}^{\mathtt{NI}}$.

Proof. By definition of Galois connection: $\alpha_{\mathfrak{m}}((\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}_{\mathsf{L}}^2) \sqsubseteq \mathfrak{m}^{\mathtt{NI}}$ if and only if $(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}_{\mathsf{L}}^2 \subseteq \gamma_{\mathfrak{m}}(\mathfrak{m}^{\mathtt{NI}})$. It is clear that $\gamma_{\mathfrak{m}}(\mathfrak{m}^{\mathtt{NI}}) \subseteq \mathtt{equiv}_{\mathsf{L}}$, since the concretization of $\mathfrak{m}^{\mathtt{NI}}$ contains L-equivalent memories, by construction. Hence, due to Prop. 15, the theorem is proved. □

Hence, in order to verify Non-Interference it is sufficient to have an abstract semantics, computing on $\mathsf{Mem}^{\mathsf{cprp}}$, which approximates $(\!|\mathsf{P}|\!)^{\curvearrowright}$. Indeed we can prove $\mathsf{P} \models \mathsf{NI}$ by computing an over-approximation of $(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{I}_{\mathsf{L}}^2$ in $\mathsf{Mem}^{\mathsf{cprp}}$.

8.2.2 The Abstract Semantics for Non-Interference

Finally, we have to show how to compute a program's hypersemantics on the proposed abstract domain. The abstract semantics for programs relies on the abstract semantics for arithmetic expressions, given in Figure 8.2. The abstract semantics for arithmetic expressions $(\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{C}} \in \mathsf{Mem}^{\mathsf{cprp}} \to \mathsf{C}^{\mathsf{cprp}}$ evaluates to an abstract value and it relies on the abstract mathematical operations given in Figure 8.2. This semantics must be such that abstract assignments are sound approximations of the concrete ones. Since we are in a (double) non-relational setting the soundness requirement is: $\{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \subseteq \gamma_{h^{\mathsf{cprp}}}(\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}$ (the proof can be found in Appendix A). We obtain this defining abstract operations $\oplus^{\mathsf{c}} \in \mathsf{C}^{\mathsf{cprp}} \times \mathsf{C}^{\mathsf{cprp}} \to \mathsf{C}^{\mathsf{cprp}}$ such that they are sound w.r.t. the concrete ones $\oplus$, with $\oplus \in \{+, -, *\}$. This technically means that they satisfy the following constraint:

$$\{\{n \oplus m \mid n \in X \wedge m \in Y\} \mid X \in \gamma_{h^{\mathsf{cprp}}}(c_1) \wedge Y \in \gamma_{h^{\mathsf{cprp}}}(c_2)\} \subseteq \gamma_{h^{\mathsf{cprp}}}(c_1 \oplus^{\mathsf{c}} c_2)$$

The constraint basically requires that every possible result obtained applying the concrete operation is contained in the concretization of the application of the abstract operator. The proof of the soundness of the abstract mathematical operations can be found in Appendix A.

The abstract semantics for boolean expression $(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}} \in \mathsf{Mem}^{\mathsf{cprp}} \to \mathsf{Mem}^{\mathsf{cprp}}$ is an abstract filtering function, and it relies on the abstract logical operations given in Figure 8.3. To simplify, we assume that all negations $\neg$ have been removed using DeMorgan's laws and usual arithmetic laws: $\neg(\mathsf{b}_1 \vee \mathsf{b}_2) \equiv (\neg\mathsf{b}_1) \wedge (\neg\mathsf{b}_2)$, $\neg(\mathsf{a}_1 < \mathsf{a}_2) \equiv (\mathsf{a}_2 \leq \mathsf{a}_1)$, etc. This semantics must be sound w.r.t. the concrete hypersemantics for booleans, namely $\{\{\mathsf{m} \in X \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} = (\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}$ (the proof can be found in Appendix A). Also in this case we cannot rely, as in the standard case of classic program verification for trace properties, on the identity function as a sound approximation: $(\!|\mathsf{b}|\!)^{\curvearrowright}$ is not reductive. In order to obtain a sound semantics, we have defined the abstract comparators $\bowtie^{\mathsf{c}} \in \mathsf{C}^{\mathsf{cprp}} \times \mathsf{C}^{\mathsf{cprp}} \to \mathsf{C}^{\mathsf{cprp}} \times \mathsf{C}^{\mathsf{cprp}}$ such that they are sound w.r.t. the concrete ones $\bowtie$, with $\bowtie \in \{=, \neq, <, \leq\}$.

Arithmetic expressions: $(\!|\mathsf{a}|\!)^{\rightarrow}_{\mathsf{C}} \in \mathsf{Mem}^{\mathsf{Cprp}} \to \mathsf{C}^{\mathsf{Cprp}}$ with $\oplus \in \{+, -, *\}$

$$(\!|\mathsf{n}|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m} \triangleq \bar{n} \qquad (\!|\mathsf{x}|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m} \triangleq \mathfrak{m}(\mathsf{x}) \qquad (\!|(\mathsf{a})|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m} \triangleq (\!|\mathsf{a}|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m} \qquad (\!|\mathsf{a}_1 \oplus \mathsf{a}_2|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m} \triangleq (\!|\mathsf{a}_1|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m} \oplus^{\mathsf{c}} (\!|\mathsf{a}_2|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m}$$

Abstract mathematical operations: $\oplus^{\mathsf{c}} \in \mathsf{C}^{\mathsf{Cprp}} \times \mathsf{C}^{\mathsf{Cprp}} \to \mathsf{C}^{\mathsf{Cprp}}$

$\oplus^{\mathsf{c}}$	$\bot$	$\bar{n}$	κ	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\bar{m}$	$\bot$	$\overline{m \oplus n}$	κ	$\top$
κ	$\bot$	κ	κ	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$

Figure 8.2: Abstract semantics for arithmetic expressions.

This technically means that they satisfy the following constraint:

$$\begin{aligned}&\text{let } \mathcal{X} = \{\{\langle n, m\rangle \mid n \in X \wedge m \in Y \wedge n \bowtie m\} \mid X \in \gamma_{h^{\mathsf{Cprp}}}(c_1) \wedge Y \in \gamma_{h^{\mathsf{Cprp}}}(c_2)\} \text{ in}\\ &\langle\{\{n \mid \langle n, m\rangle \in X\} \mid X \in \mathcal{X}\}, \{\{m \mid \langle n, m\rangle \in X\} \mid X \in \mathcal{X}\}\rangle \subseteq^2 \gamma_{h^{\rho}}(c_1 \bowtie^{\mathsf{c}} c_2)\end{aligned}$$

The constraint basically requires that every possible pair of values making true the concrete comparator is contained in the concretization of the application of the abstract comparator. The proof of the soundness of the abstract logical operations can be found in Appendix A. In Figure 8.3, the memory $\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{n} \trianglelefteq c_1\}$ can be approximated with a *backward* abstract semantics for arithmetic expressions ${}^{\mathcal{B}}(\!|\mathsf{a}|\!)^{\rightarrow}_{\mathsf{C}} \in \mathsf{Mem}^{\mathsf{Cprp}} \to (\mathsf{C}^{\mathsf{Cprp}} \to \mathsf{Mem}^{\mathsf{Cprp}})$. ${}^{\mathcal{B}}(\!|\mathsf{a}|\!)^{\rightarrow}_{\mathsf{C}}(\mathfrak{m})(c) = \mathfrak{m}'$ means that $\mathfrak{m}'$ is a refinement of $\mathfrak{m}$, i.e. $\mathfrak{m}' \sqsubseteq \mathfrak{m}$, such that $(\!|\mathsf{a}|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m}' \trianglelefteq c$.

Example 26. Let us see how to compute $(\!|\mathsf{x} < 2|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m}$ where $\mathfrak{m} = [\mathsf{x} \mapsto \bar{1}\ \mathsf{y} \mapsto \kappa]$. We have $(\!|\mathsf{x}|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m} = \bar{1}$, $(\!|2|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m} = \bar{2}$ and $\bar{1} <^{\mathsf{c}} \bar{2} = \langle\bar{1}, \bar{2}\rangle$. Then $\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{x}|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{n} \trianglelefteq \bar{1}\} = [\mathsf{x} \mapsto \bar{1}\ \mathsf{y} \mapsto \kappa]$ and $\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|2|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{n} \trianglelefteq \bar{2}\} = [\mathsf{x} \mapsto \bar{1}\ \mathsf{y} \mapsto \kappa]$. Finally, $[\mathsf{x} \mapsto \bar{1}\ \mathsf{y} \mapsto \kappa] \sqcap [\mathsf{x} \mapsto \bar{1}\ \mathsf{y} \mapsto \kappa] = [\mathsf{x} \mapsto \bar{1}\ \mathsf{y} \mapsto \kappa]$, which is indeed equal to $\mathfrak{m}$. Suppose now to compute the negation of this boolean expression, namely we want to compute $(\!|2 \leq \mathsf{x}|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{m}$. In this case we have $\bar{2} \leq^{\mathsf{c}} \bar{1} = \langle\top, \top\rangle$, $\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|2|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{n} \trianglelefteq \top\} = [\mathsf{x} \mapsto \bar{1}\ \mathsf{y} \mapsto \kappa]$ and $\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{x}|\!)^{\rightarrow}_{\mathsf{C}}\mathfrak{n} \trianglelefteq \top\} = [\mathsf{x} \mapsto \bar{1}\ \mathsf{y} \mapsto \kappa]$. Hence we obtain, again, $[\mathsf{x} \mapsto \bar{1}\ \mathsf{y} \mapsto \kappa]$ as result.

Finally, we need two auxiliary functions $\mathsf{vars}^{\top}_{\mathfrak{m}}(\mathsf{b})$ and $\mathsf{vars}^{:=}(\mathsf{P})$, returning the set of variables occurring in b having value $\top$ when evaluated in $\mathfrak{m}$ and the set of *modified variables* in P, respectively. The first is straightforward to compute: $\mathsf{vars}^{\top}_{\mathfrak{m}}(\mathsf{b}) \triangleq \{\mathsf{x} \in \mathsf{vars}(\mathsf{b}) \mid \mathfrak{m}(\mathsf{x}) = \top\}$ and $\mathsf{vars}(\mathsf{b})$ is just a syntactic check. The second involves semantic information, hence it is not trivial to compute. Naively, we can use a simple syntactic approach for approximating the set of variables which may be modified during P executions. Indeed, the function $\mathsf{vars}^{:=}(\mathsf{P})$ returns the set of variables occurring in P on the left-hand side of an assignment, which is easy implementable as a syntactic check. We plan to enhance our abstract semantics with a semantic check for modified variables, in order to increase precision, as a future work.

Now we have all the ingredients needed to define the abstract hypersemantics $(\!|\mathsf{P}|\!)^{\rightarrow}_{\mathsf{C}} \in$

Boolean expressions: $(\!|\mathsf{b}|\!)^{\sqsubset}_{\mathsf{C}} \in \mathsf{Mem}^{\mathsf{Cprp}} \to \mathsf{Mem}^{\mathsf{Cprp}}$ with $\bowtie \in \{=, \neq, <, \leq\}$

$$(\!|\mathsf{tt}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \triangleq \mathfrak{m} \qquad (\!|\mathsf{ff}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \triangleq \mathfrak{m}_{\bot} \qquad (\!|(\mathsf{b})|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \triangleq (\!|\mathsf{b}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m}$$

$$(\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \triangleq (\!|\mathsf{b}_1|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \sqcap (\!|\mathsf{b}_2|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m}$$

$$(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \triangleq \begin{array}{l}\text{let } \mathfrak{n} = (\!|\mathsf{b}_1|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \sqcup (\!|\mathsf{b}_2|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \text{ in} \\ \lambda \mathsf{x}\,.\, [\![\, \mathfrak{m}(\mathsf{x}) = \top \wedge \mathsf{x} \in \mathsf{vars}(\mathsf{b}_1) \cap \mathsf{vars}(\mathsf{b}_2) \,?\, \top : \mathfrak{n}(\mathsf{x}) \,]\!]\end{array}$$

$$(\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \triangleq \begin{array}{l}\text{let } \langle c_1, c_2\rangle = (\!|\mathsf{a}_1|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \bowtie^{\mathsf{c}} (\!|\mathsf{a}_2|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \text{ in} \\ \bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_1|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{n} \trianglelefteq c_1\} \sqcap \bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_2|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{n} \trianglelefteq c_2\}\end{array}$$

Abstract logical operations: $\bowtie^{\mathsf{c}} \in \mathsf{C}^{\mathsf{Cprp}} \times \mathsf{C}^{\mathsf{Cprp}} \to \mathsf{C}^{\mathsf{Cprp}} \times \mathsf{C}^{\mathsf{Cprp}}$

$$c_1 \bowtie^{\mathsf{c}} c_2 \triangleq \begin{cases} \langle \bot, \bot\rangle & \text{if } c_1 = \bot \text{ or } c_2 = \bot \\ \langle \bar{n}, \bar{n}\rangle & \text{if } c_1 = c_2 = \bar{n}, \bowtie \in \{=, \leq\} \\ \langle \bar{n}, \bar{m}\rangle & \text{if } c_1 = \bar{n}, c_2 = \bar{m}, n < m, \bowtie \in \{<, \leq, \neq\} \\ \langle \bar{n}, \bar{m}\rangle & \text{if } c_1 = \bar{n}, c_2 = \bar{m}, n > m, \bowtie \in \{\neq\} \\ \langle \top, \top\rangle & \text{otherwise} \end{cases}$$

Figure 8.3: Abstract semantics for boolean expressions.

$\mathsf{Mem}^{\mathsf{Cprp}} \to \mathsf{Mem}^{\mathsf{Cprp}}$, which is:

$$(\!|\mathsf{P}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m}_{\bot} \triangleq \mathfrak{m}_{\bot} \qquad (\!|\,{}^{\mathtt{i}}\mathsf{c}_1\,{}^{\mathtt{l}} \,.\, {}^{\mathtt{k}}\mathsf{c}_2\,{}^{\mathtt{f}}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \triangleq (\!|\,{}^{\mathtt{k}}\mathsf{c}_2\,{}^{\mathtt{f}}|\!)^{\sqsubset}_{\mathsf{C}}(\!|\,{}^{\mathtt{i}}\mathsf{c}_1\,{}^{\mathtt{l}}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \qquad (\!|\,{}^{\mathtt{i}}\mathsf{skip}\,{}^{\mathtt{f}}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \triangleq \mathfrak{m}$$

$$(\!|\,{}^{\mathtt{i}}\mathsf{x} := \mathsf{a}\,{}^{\mathtt{f}}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \triangleq \mathfrak{m}[\mathsf{x} \hookleftarrow (\!|\mathsf{a}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m}]$$

$$(\!|\,{}^{\mathtt{i}}\mathsf{if\ b\ then\ \{\,P_1\,\}\ else\ \{\,P_2\,\}}\,{}^{\mathtt{f}}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \triangleq \mathfrak{w} \text{ where}$$
$$\text{let } \mathfrak{n} = (\!|\mathsf{P}_1|\!)^{\sqsubset}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \sqcup (\!|\mathsf{P}_2|\!)^{\sqsubset}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \text{ in}$$
$$\mathfrak{w} = \lambda \mathsf{x}\,.\begin{cases} \mathfrak{n}(\mathsf{x}) & \text{if } \mathsf{x} \notin \mathsf{vars}^{:=}(\mathsf{P}_1) \cup \mathsf{vars}^{:=}(\mathsf{P}_2) \vee \mathfrak{n}(\mathsf{x}) \not\trianglelefteq \kappa \vee \mathsf{vars}^{\top}_{\mathfrak{m}}(\mathsf{b}) = \varnothing \\ \top & \text{otherwise} \end{cases}$$

$$(\!|\,{}^{\mathtt{i}}\mathsf{while}\,{}^{\mathtt{k}}\mathsf{b}\,\{\,\mathsf{P}\,\}\,{}^{\mathtt{f}}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{m} \triangleq (\!|\neg\mathsf{b}|\!)^{\sqsubset}_{\mathsf{C}}\big(\mathsf{lfp}^{\sqsubset}_{\mathfrak{m}_{\bot}}\ \lambda\mathfrak{n}\,.\,\mathfrak{m} \sqcup (\!|\,{}^{\mathtt{i}}\mathsf{if\ b\ then\ \{\,P\,\}\ else\ \{}\,{}^{\mathtt{h}}\mathsf{skip}\,{}^{\mathtt{h}}\,\mathsf{\}}\,{}^{\mathtt{f}}|\!)^{\sqsubset}_{\mathsf{C}}\mathfrak{n}\big)$$

with $\mathtt{h}$ fresh label (it is indeed not necessary, since the post-conditions hypersemantics does not take into account labels). The abstract semantics is quite standard for all statements, except for conditionals. We will explain here only this latter, which exploits the following idea. For every variable, we make the join between its value resulting after the execution of the true branch and its value resulting after the execution of the false branch. This is done in order to track the forbidden flows (implicit or explicit) generated inside the two branches. In fact, a L variable has value $\top$ after the join if in at least one of the branches it has value $\top$ (meaning that there is a forbidden flow). After this check we need to take in consideration the implicit flows generated by the conditional statement itself. Indeed, first we suppose that if there is at least one variable with value $\top$ *before* the boolean guard is evaluated, then all variables modified in the conditional branches have a forbidden flow (a variable has

value $\top$ only if it is a H variable or if it has been "influenced" by a H variable). This is done by setting to $\top$ all modified variables. Note that if, for some reasons, a H variable is not $\top$ during this check, the flow is correctly not set. This procedure is sound but not so precise. In order to enhance precision, we exploit our abstract domain. In particular, we do not set to $\top$ the variables which have the same constant value $\bar{n}$ in both branches (this is the condition $\mathfrak{n}(x) \trianglelefteq \kappa$) because this means that at the end of the conditional statement the variable has always a constant value. If there are no $\top$-valued variables into the guard b (this is the condition $\mathsf{vars}^{\top}_{\mathfrak{m}}(b) = \varnothing$), then no variables are set to $\top$: the resulting flows are those generated into the two branches of the conditional.

We proved that our abstract semantics is sound w.r.t. the concrete hypersemantics, and that it can be used for Non-Interference verification. This is stated in the following two theorems.

Theorem 27 (Soundness)**.** *The abstract hypersemantics is sound w.r.t. the concrete hypersemantics: for every* $\mathfrak{m} \in \mathsf{Mem}^{\mathsf{cprp}}$ *we have* $(\!|P|\!)^{\natural}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}(\!|P|\!)^{\natural}_{C}\mathfrak{m}$.

Proof. (Sketch) We just have to prove that the abstract hypersemantics approximates the best correct approximation of the concrete hypersemantics in $\mathsf{Mem}^{\mathsf{cprp}}$, namely $\alpha_{\mathfrak{m}} \circ (\!|P|\!)^{\natural} \circ \gamma_{\mathfrak{m}} \mathrel{\dot{\sqsubseteq}} (\!|P|\!)^{\natural}_{C}$. The proof relies on the soundness of the abstract semantics for arithmetic and boolean expressions and it is done by structural induction. We omit here the full proof, we just show as example the case for assignments (the full proof can be found in Appendix A).

$$
\begin{array}{ll}
& \alpha_{\mathfrak{m}}(\!|\, {}^{\mathtt{i}}x := a\, {}^{\mathtt{f}} |\!|^{\natural}\gamma_{\mathfrak{m}}(\mathfrak{m}) \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\natural} \\
& \alpha_{\mathfrak{m}}(\{\{\mathfrak{m}[x \leftarrowtail n] \mid \mathfrak{m} \in X \wedge \langle a, \mathfrak{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}) \\
= & \quad \|\alpha_{\mathfrak{m}} = \dot{\alpha}_{h^{\mathsf{cprp}}} \circ \alpha_{nnr} \text{ and definition of } \alpha_{nnr} \\
& \dot{\alpha}_{h^{\mathsf{cprp}}} \circ (\lambda y\,.\, \{\{\mathfrak{m}(y) \mid \mathfrak{m} \in X\} \mid X \in \{\{\mathfrak{m}[x \leftarrowtail n] \mid \mathfrak{m} \in X \wedge \langle a, \mathfrak{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}\}) \\
= & \\
& \dot{\alpha}_{h^{\mathsf{cprp}}} \circ (\lambda y\,.\, [\![\, y = x \mathbin{?} \{\{n \mid \exists \mathfrak{m} \in X\,.\, \langle a, \mathfrak{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \mathbin{:} \{\{\mathfrak{m}(y) \mid \mathfrak{m} \in X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \,]\!]) \\
\sqsubseteq & \quad \| \text{ soundness of } (\!|a|\!)^{\natural}_{C} \\
& \dot{\alpha}_{h^{\mathsf{cprp}}} \circ (\lambda y\,.\, [\![\, y = x \mathbin{?} \gamma_{h^{\mathsf{cprp}}}(\!|a|\!)^{\natural}_{C}\mathfrak{m} \mathbin{:} \{\{\mathfrak{m}(y) \mid \mathfrak{m} \in X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \,]\!]) \\
= & \quad \| \text{ definition of } \dot{\alpha}_{h^{\mathsf{cprp}}} \\
& \lambda y\,.\, [\![\, y = x \mathbin{?} \alpha_{h^{\mathsf{cprp}}}\gamma_{h^{\mathsf{cprp}}}(\!|a|\!)^{\natural}_{C}\mathfrak{m} \mathbin{:} \alpha_{h^{\mathsf{cprp}}}(\{\{\mathfrak{m}(y) \mid \mathfrak{m} \in X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}) \,]\!] \\
= & \quad \| \text{ definition of } \gamma_{\mathfrak{m}} \\
& \lambda y\,.\, [\![\, y = x \mathbin{?} \alpha_{h^{\mathsf{cprp}}}\gamma_{h^{\mathsf{cprp}}}(\!|a|\!)^{\natural}_{C}\mathfrak{m} \mathbin{:} \alpha_{h^{\mathsf{cprp}}}\gamma_{h^{\mathsf{cprp}}}(\mathfrak{m}(y)) \,]\!] \\
\sqsubseteq & \quad \| \text{ reductivity of } \alpha_{h^{\mathsf{cprp}}}\gamma_{h^{\mathsf{cprp}}} \\
& \lambda y\,.\, [\![\, y = x \mathbin{?} (\!|a|\!)^{\natural}_{C}\mathfrak{m} \mathbin{:} \mathfrak{m}(y) \,]\!] \\
= & \\
& \mathfrak{m}[x \leftarrowtail (\!|a|\!)^{\natural}_{C}\mathfrak{m}] \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\natural}_{C} \\
& (\!|\, {}^{\mathtt{i}}x := a\, {}^{\mathtt{f}} |\!|^{\natural}_{C}\mathfrak{m}
\end{array}
$$

□

With the abstract semantics just introduced we can define an effective verification method for Non-Interference.

Theorem 28 (NI Verification)**.** *We have that* $\mathsf{P} \models \mathsf{NI}$ *if* $(\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}^{\mathsf{NI}} \sqsubseteq \mathfrak{m}^{\mathsf{NI}}$.

Proof. Note that $\alpha_{\mathfrak{m}}(\mathcal{I}^2_{\mathsf{L}}) \sqsubseteq \mathfrak{m}^{\mathsf{NI}}$ since $\mathcal{I}^2_{\mathsf{L}}$ contains only sets of memories agreeing on L variables. This means that $\alpha_{\mathfrak{m}}(\mathcal{I}^2_{\mathsf{L}})(\mathsf{x}) \trianglelefteq \kappa = \mathfrak{m}^{\mathsf{NI}}(\mathsf{x})$, for each L variable x. For H variables y, $\alpha_{\mathfrak{m}}(\mathcal{I}^2_{\mathsf{L}})(\mathsf{y}) \trianglelefteq \top = \mathfrak{m}^{\mathsf{NI}}(\mathsf{y})$ trivially holds. Then the proof is given by the following implications:

$$
\begin{array}{l}
(\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}^{\mathsf{NI}} \sqsubseteq \mathfrak{m}^{\mathsf{NI}} \\
\quad \Downarrow \qquad \| \text{ monotonicity of } (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}} \text{ and } \alpha_{\mathfrak{m}}(\mathcal{I}^2_{\mathsf{L}}) \sqsubseteq \mathfrak{m}^{\mathsf{NI}} \\
(\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}\alpha_{\mathfrak{m}}(\mathcal{I}^2_{\mathsf{L}}) \sqsubseteq (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}^{\mathsf{NI}} \sqsubseteq \mathfrak{m}^{\mathsf{NI}} \\
\quad \Downarrow \qquad \| \text{ soundness of } (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}} \text{ (Theprem 27) and } \alpha_{\mathfrak{m}},\gamma_{\mathfrak{m}} \text{ adjunction: } \forall \mathcal{X} \in \wp(\wp(\mathsf{Mem})) \,.\, \alpha_{\mathfrak{m}}(\!|\mathsf{P}|\!)^{\curvearrowright}\mathcal{X} \sqsubseteq (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}\alpha_{\mathfrak{m}}(\mathcal{X}) \\
\alpha_{\mathfrak{m}}((\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}\mathcal{I}^2_{\mathsf{L}}) \sqsubseteq (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}\alpha_{\mathfrak{m}}(\mathcal{I}^2_{\mathsf{L}}) \sqsubseteq (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}^{\mathsf{NI}} \sqsubseteq \mathfrak{m}^{\mathsf{NI}} \\
\quad \Downarrow \qquad \| \text{ Theorem 26} \\
\mathsf{P} \models \mathsf{NI}
\end{array}
$$

□

This means that we can check Non-Interference simply by checking that each set of computations, starting from L-equivalent memories, provides only singletons as results.

8.2.3 The Prototype Analyzer

With the only aim of proving the feasibility of the proposed approach, and in particular of the abstract hypersemantics, we have written a prototype analyzer, called nonInterfer, in Java SE 10 for Imp programs, which implements the abstract hypersemantics of the previous subsection.

8.2.3.1 Validation

Since our analyzer is built for a toy language, there are no benchmark tests sets. So we have measured speed and precision of the tool building our own tests set. We have written 25 non-interferent programs and 25 interferent programs, with different levels of complexity. As expected, the prototype does not output false negatives, i.e. all interferent programs are discovered. For what concerns precision, the analyzer marks 3 programs as interferent even if they actually satisfy Non-Interference. In Figure 8.4 we have four example programs, where variables a, b are public and variables x, y are private. The initial abstract memory $\mathfrak{m}^{\mathsf{NI}}$ is $[\mathsf{a} \mapsto \kappa \;\; \mathsf{b} \mapsto \kappa \;\; \mathsf{x} \mapsto \top \;\; \mathsf{y} \mapsto \top]$. We have that $(\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}^{\mathsf{NI}} = [\mathsf{a} \mapsto \bar{0} \;\; \mathsf{b} \mapsto \bar{0} \;\; \mathsf{x} \mapsto \top \;\; \mathsf{y} \mapsto \top] \sqsubseteq \mathfrak{m}^{\mathsf{NI}}$, meaning that the analyzer correctly marks P_1 as non-interferent. Analyzing program P_2, the verifier is able to catch an implicit indirect flow, in fact $(\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}^{\mathsf{NI}} = [\mathsf{a} \mapsto \top \;\; \mathsf{b} \mapsto \top \;\; \mathsf{x} \mapsto \top \;\; \mathsf{y} \mapsto \top] \not\sqsubseteq \mathfrak{m}^{\mathsf{NI}}$ (i.e. P_2 is correctly marked as interferent). Unfortunately, our analyzer signals a false alarm in program P_3, indeed $(\!|\mathsf{P}_3|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}^{\mathsf{NI}} = [\mathsf{a} \mapsto \top \;\; \mathsf{b} \mapsto \kappa \;\; \mathsf{x} \mapsto \top \;\; \mathsf{y} \mapsto \top] \not\sqsubseteq \mathfrak{m}^{\mathsf{NI}}$, even if the program is non-interferent. Finally, we have a precise result on the more complex program P_4, indeed $(\!|\mathsf{P}_4|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}^{\mathsf{NI}} = [\mathsf{a} \mapsto \bar{0} \;\; \mathsf{b} \mapsto \kappa \;\; \mathsf{x} \mapsto \top \;\; \mathsf{y} \mapsto \top] \sqsubseteq \mathfrak{m}^{\mathsf{NI}}$. Note that classic security

Program P_1:

$^{0}\mathsf{a} := 0^{1}$.
^{1}if $(\mathsf{b} < \mathsf{x})$ then $\{\ ^{2}\mathsf{b} := \mathsf{a} * 3^{3}\ \}$
else $\{\ ^{4}\mathsf{b} := \mathsf{a} - ((2 * \mathsf{a}) - \mathsf{a})^{5}\ \}^{6}$

Program P_2:

$^{0}\mathsf{a} := \mathsf{x}^{1}$.
^{1}if $(\mathsf{b} < \mathsf{a})$ then $\{\ ^{2}\mathsf{b} := 2^{3}\ \}$
else $\{\ ^{4}\mathsf{b} := 3^{5}\ \}^{6}$

Program P_3:

$^{0}\mathsf{a} := \mathsf{x} - \mathsf{x}^{1}$

Program P_4:

$^{0}\mathsf{a} := 0^{1}$.
^{1}while $^{2}(\mathsf{x} < \mathsf{y})$ {
$^{3}\mathsf{x} := \mathsf{x} + 1^{4}$.
^{4}while $^{5}(\mathsf{a} < \mathsf{x})$ $\{\ ^{6}\mathsf{a} := \mathsf{a} + 2^{7}\ \}^{8}$.
$^{8}\mathsf{a} := 0^{9}$
$\}^{10}$

Figure 8.4: Example programs.

type systems for information flows (derived from [Volpano, Irvine, and Smith, 1996]) are not able to type correctly the program P_4, namely they rise a false alarm in this case.

The finite height of the hyper abstract domain $\mathbb{C}^{\text{cprp}}$ guarantees the termination of the analysis. Furthermore, the structure of the domain allows us to compute loops fixpoints quickly, hence there is no need for a widening operator in order to speed-up the analysis. On our tests set, which comprises quite small hand-made programs, the analyzer is very fast. The analysis time is around 120 milliseconds in average, running on a commodity hardware[5]. We also tested the prototype on bigger programs, generated automatically with the tool Grammarinator[Hodován and Kiss, 2018]. The analyzer is able to handle programs with hundreds of lines of code, basically with the same speed time. As expected, the analyzer shows some slowdowns when programs use lots of variables. Nevertheless, its running time is lower than 600 milliseconds even on programs with more than 500 variables.

Indeed the analyzer exhibits a good trade-off between verification speed and precision (in general). Unfortunately, our analyzer is not precise in some trivial situations, like in P_3, hence in the next paragraphs we discuss about how it is possible to obtain better results.

8.2.3.2 Improving Precision

The analyzer has a good precision overall but it signals false alarms in some, sometimes very trivial, cases. We mentioned in the previous subsection that our current approach for the approximation of modified variables is a very simple syntactic check. With a more semantic analysis we can gain precision and do not rise false alarms for programs like P_3 of Figure 8.4. Apart from this detail, the sources of imprecision of our semantics are basically two: the approximation added making the hyperlevel constants domain machine-representable and the lack of relational information between variables. In this section we deal with these two issues.

Tuning the Hyperlevel Constants Domain. The original hyperlevel constants domain of Section 6.2 contains all the elements of the powerset of $\{\{n\} \mid n \in \mathbb{Z}\}$, meaning that every possible combination of constant sets is taken into account. This makes the domain

[5]Laptop with Arch Linux 64-bit (kernel 4.17.5-1), Intel Core i7-7700HQ CPU, 8GiB RAM and SSD storage.

very precise but not machine-representable, as already observed. In our implementation we have chosen to represent precisely only the singletons $\{\{n\}\}$, abstracted to $\bar{n}$, and the set of all singleton sets $\{\{n\} \mid n \in \mathbb{Z}\}$, abstracted to κ. In order to enhance precision we could extend our domain $\mathsf{C}^{\mathsf{Cprp}}$ with pairs of constant sets, namely we can represent sets of the form $\{\{n\}, \{m\}\}$, with $n, m \in \mathbb{Z}$. But we can gain more precision taking into account triples, quadruples and so on. Hence we can infinitely tune the precision of the analyzer. Clearly the more elements we add to the domain and the more space is consumed by the analyzer and the more abstract operations are complex. So, the trade-off between precision and performance of the analysis depends on the analyzer's context of application.

Add Relational Information. Our analysis is not-relational, meaning that we do not explicitly track relations between different variables. We can increase the precision pairing $\mathsf{Mem}^{\mathsf{Cprp}}$ with a relational abstraction of $\wp(\wp(\mathsf{Mem}))$. For instance we can define an abstract domain tracking equalities between variables. This latter, combined with a numerical domain such as the one for intervals will improve the precision w.r.t. implicit flows.

```
0 b := 1 1 .
1 if (b = x) then {
    2 a := 3 1 .
    2 while 3 (a ≠ 1) {
        4 a := a − 1 1 .
        4 b := a 5
    } 6
  else { 7 a := 1 5 } 8
```

Take, as example, the program here on the left, where variables a, b are L while variable x is H. Our analyzer signals a false alarm, since the program is non-interferent but our analysis outputs an abstract memory assigning $\top$ to all variables. With an interval analysis we are able to find that variable a is equal to $[1, 1]$ at the end of the while and with a domain tracking equalities we can deduce the same for variable b. Hence, at the end of the program we can improve our analysis obtaining the abstract memory $[\mathsf{a} \mapsto \bar{1}\ \mathsf{b} \mapsto \bar{1}\ \mathsf{x} \mapsto \top]$, allowing us to prove that the program is non-interferent.

9 Concluding Remarks

With this thesis we made a little step towards the verification of hyperproperties by means of abstract interpretation. We have introduced a formal framework for modeling system semantics at the same level as hyperproperties, namely at sets of sets of executions level. These more expressive hypersemantics not only allow us to provide weaker forms of satisfiability but provide a methodology allowing us to lift static analysis (for hyperproperties) directly at the hyper level. We believe that this approach could provide a deep insight and useful formal tools also for tackling the problem of analyzing analyzers, aiming at systematically analyzing static analyses [Giacobazzi, Logozzo, and Ranzato, 2015; Cousot, Giacobazzi, and Ranzato, 2019]. In particular, we believe that hyperdomains, introduced in Section 6.2, can be used not only for hyperproperties verification but also for this latter purpose.

In the thesis, we also made a little step into the understanding of hyperproperties. In particular, we reasoned on a subset of subset-closed hyperproperties, which is more suitable for verification. For subset-closed hyperproperties we can prove that a program does not satisfies the specification by finding a subset of the program semantics which does not satisfy the hyperproperty. If we can limit the cardinality of these refuting witnesses we obtain the bounded subset-closed hyperproperties. These latter generalize k-hypersafety and some hyperliveness, so they capture a lot of interesting systems specifications. In this work, we described how it is possible to leverage the standard abstract interpretation based static analysis framework in order to verify bounded subset-closed hyperproperties. In particular, we showed how to lift a standard semantic operator to sets of sets and how to build hyper abstract domains. Putting all the ingredients together, we specified the recipe for defining an hyperanalysis, namely a static analysis for (bounded) hyperproperties.

We have also investigated the definition of systems specifications in a parametric setting. The first parameter distinguishes if the specifications are trace properties or hyperproperties. The first are simpler to check (they can be verified observing single executions) but they lose the power to express specifications describing relations between executions. The second parameter concerns what kind of executions denotations we are able to express: only finite, only infinite or mixed (finite and infinite) executions. We have analyzed how the well known safety/liveness classification of trace properties changes in relation with the latter two parameters. Some work in this direction was already done by Roşu [Roşu, 2012], but only for the safety part and only for trace properties.

The beauty of the safety/liveness classification is its topological interpretation, which allows us to decompose every trace property in its safety part and its liveness part. This means that we can decompose the verification process in two, more simpler, parts as well. To the best of our knowledge, this topologies were specified only for trace properties on

infinite executions [Manna and Pnueli, 1995] and for hyperproperties on infinite executions [Clarkson and Schneider, 2010]. Our work gives a topological interpretation also for the others combinations: trace properties on finite and on mixed executions, hyperproperties on finite and on mixed executions. We proved that in each combinations the safety/hypersafety are the closed sets in the corresponding topology and the liveness/hyperliveness are the dense sets. This means that the "decomposition method" can be applied in all six cases, not only in the infinite executions ones.

Finally, we showed step by step how to build a static analyzer for a particular bounded hyperproperty, namely for (Abstract) Non-Interference, based on abstract interpretation. The tool is sound, meaning that it will not signals false negatives, i.e. if the analyzer returns that a program is non-interferent then it is guaranteed that it satisfies Non-Interference. We have soundness by design, exploiting the framework of abstract interpretation. We follow the theoretical results obtained in Chapter 7 and we made the abstract domain computer-representable, hence we show how to make the analysis feasible. Furthermore, we simplified the process of Non-Interference verification, moving from a semantics computing on input/output traces to a simpler semantics computing on memories.

We implemented the analyzer, called nonInterfer, in order to validate the abstract semantics. The testing on the prototype has lead to very promising results, in particular w.r.t. analysis speed. Non-Interference verification is undecidable, hence we have obviously false negative, namely sometimes the analyzer marks as interferent a program which actually satisfies Non-interference. Despite its simplicity and speed, the analyzer is quite precise, at least as precise as classic (security) type systems for Non-Interference.

To the best of our knowledge, all the works about Non-Interference, except for [Assaf et al., 2017], verify (or, more often, enforce) it "downgrading" the hyperproperty at the standard level of sets of traces, namely at the level of the program (base) semantics. As we have seen in Subsection 6.1.2, approximating an hyperproperty with a stronger trace property leads to a coarse analysis. With our framework we follow the opposite direction, namely we lift the semantics at the same level of the hyperproperty, namely at the level of sets of sets of traces. Our idea was to build an hyper abstract interpreter for abstract non-interference, which would be more precise, but still sound, w.r.t. actual approaches.

9.1 Related Works

To the best of our knowledge, there are very few works trying to verify (Abstract) Non-Interference, or hyperproperties in general, without resorting to a stronger trace property (as explained in Section 6.1.2). The closest related works are [Assaf et al., 2017] and [Urban and Müller, 2018], which both deal with hyperproperties by means of abstract interpretation. In the second, the authors define an hyperproperty called Input Data Usage, claiming that it generalizes a lot of notion of information flows, comprising Non-Interference. They propose an ad-hoc hypersemantics useful to verify that hyperproperty. Then they show how it is possible to obtain, by abstraction of their semantics, some known syntactic verification methods for information flows. Nevertheless, they do not introduce abstract semantics suitable for verifying information flows by abstract interpretation. The work of [Assaf et al., 2017] introduces a *hypercollecting semantics*, i.e. the first example of what we call hypersemantics. Nevertheless, their work is focused on information flows, indeed they pro-

pose two abstract semantics: one for qualitative Non-Interference and one for quantitative Non-Interference. Quantitative information flows can be seen as a way of declassification, but they are not directly comparable with Abstract Non-Interference (specifically, quantitative information flows are not k-bounded). In fact this latter aims to check interference between properties of data whilst quantitative information flows just allow to leak bits of confidential information. For the qualitative case, we have that our abstract domain is not directly comparable with the "dependences" abstract domain of [Assaf et al., 2017]. Nevertheless our abstract semantics is able to state correctly Non-Interference of the program in Listing 5 of [Assaf et al., 2017] without any tuning for precision. The dependences abstract semantics of [Assaf et al., 2017] needs to add the Intervals domain in order to reach this level of precision. Finally, to the best of our knowledge, none of the previous works have an implementation, even for a toy language, supporting their theoretical results.

Another related work is [Antonopoulos et al., 2017], where authors propose a methodology for proving the absence of timing channels. This work is based on the idea of "decomposition instead of self-composition". The idea is to partition the program semantics and to analyze each partition with standard methods. Their approach is similar to our method to simplify the verification of subset-closed hyperproperties. Nevertheless, for each partition they verify a classic trace property, instead we verify an hyperproperty. This leads us to better results w.r.t. precision.

Concerning k-hypersafety, these hyperproperties can be verified with a classic mechanism for safety trace properties on the k times self-composed system. The self-composition can be sequential, parallel or in an interleaving manner and a lot of works applied this methodology [Barthe, D'Argenio, and Rezk, 2004; Terauchi and Aiken, 2005]. Unfortunately, this approach seems to be computationally too expensive to be used in practice, as observed in [Antonopoulos et al., 2017]. Besides the reduction to safety, in [Agrawal and Bonakdarpour, 2016] the authors introduce a runtime refutation method for k-hypersafety, based on a three-valued logic. Similarly, [Finkbeiner, Rabe, and Sánchez, 2015; Clarkson et al., 2014] define hyperlogics (HyperLTL and HyperCTL/CTL*), i.e. extensions of temporal logic able to quantify over multiple traces. Some algorithms for model-checking in these extended temporal logics exist, but only for particular decidable fragments, since the model-checking problem for these logics is, in general, undecidable.

Classic methods for Non-Interference verification, which do not take in consideration its hyperproperty nature, comprise the type systems *à la* Volpano [Volpano, Irvine, and Smith, 1996]. These latter perform just syntactic checks, with our approach we have more precision since we can exploit semantic information. Furthermore, none of them have been extended to Abstract Non-Interference. Some new logic-based approaches that showed up recently seem very promising, like the epistemic temporal logic of [Balliu, Dam, and Le Guernic, 2011] and SecLTL [Dimitrova et al., 2012]. They extend classic temporal logics with modalities useful for the verification of Non-Interference. Again, these works focus on Non-Interference only. It is not so easy to compare our work with these latter, we let as a future work to deepen the link between these logics and our abstract hypersemantics.

9.2 Future Directions

As a future work, it would be interesting to extend our theoretical work presented in Section 4.2.2 to the safety/progress classification [Chang, Manna, and Pnueli, 1992], which is orthogonal to the safety/liveness classification but it gives a fine-grained characterization of not-safety specifications.

As another interesting line of work, we plan to deepen the link between hypersemantics and hyperdomains with the problem of analyzing program analyses. The connection between this latter and the work presented in this thesis seems quite strong, as argued in Chapter 6.

Focusing on static program analysis for hyperproperties, there are still a lot of work to do. We have investigate the problem mainly for subset-closed hyperproperties, but some interesting specifications fall outside this latter. Furthermore, it could be interesting to extend classic methods for liveness trace properties verification to the hyper level.

In Chapter 8, we showed in detail how to design static analyzers useful to verify Abstract Non-Interference, based on abstract interpretation. The analyzers are sound, meaning that they will not signal false negatives, i.e. if the analyzer returns that a program is non-interferent then it is guaranteed that it satisfies ANI. We have soundness by design, exploiting the framework of abstract interpretation. We make the hyper (abstract) constants domain of Section 6.2 computer-representable (when needed), hence we show how to make the analysis feasible. Furthermore, we simplified the verification process for some particular subset-closed hyperproperties, moving from a semantics computing on input/output traces to a simpler semantics computing on memories.

Abstract Non-Interference verification is undecidable, hence we have obviously false negatives, namely sometimes the analyzer marks as interferent a program which actually satisfies ANI. As a future work, we want to increase the precision of the abstract hypersemantics, adding the possibility to track relational information between different variables. Finally, we want to investigate how it is possible to overcome the problem pointed out in Section 8.1.2, in order to take into account the verification of Abstract Non-Interference with declassification. Finally, since the results on Imp are promising, we plan to start the porting of our analyzer for Non-Interference nonInterfer to a real-world programming language, in order to validate its performance on more challenging test sets.

A

Long Proofs

This appendix contains the long proofs of results presented in the thesis. The proofs are listed in order of appearance and grouped by the section/subsection in which they are referred. Some other theorems/lemmas, which are needed by the main proofs, are presented and proved here as well.

Subsection 4.2.2

► *Lemma 3*

Proof. Note that $\mathit{LimPrfCl}(X) = \mathtt{prf}^{\uparrow}(X) \cup \{\bar{\sigma} \in \Sigma^{\omega} \mid \forall \bar{\sigma}' \in \Sigma^{+}.(\bar{\sigma}' \leq^{\mathrm{pf}} \bar{\sigma} \Rightarrow \bar{\sigma}' \in \mathtt{prf}^{\uparrow}(X)\} = \mathtt{prf}^{\uparrow}(X) \cup \{\bar{\sigma} \in \Sigma^{\omega} \mid \mathtt{prf}(\bar{\sigma}) \subseteq \mathtt{prf}^{\uparrow}(X)\}$. The proof id divided in two cases.

Safety case

First we prove $\mathfrak{C}_{\Sigma^{\infty}} \subseteq \mathsf{Safety}^{\infty}$. Let $X \in \mathfrak{C}_{\Sigma^{\infty}}$. For all $\bar{\sigma} \in X$ we have that $\bar{\sigma} \in \mathtt{prf}^{\uparrow}(X)$ or $\bar{\sigma} \in \{\bar{\sigma} \in \Sigma^{\omega} \mid \mathtt{prf}(\bar{\sigma}) \subseteq \mathtt{prf}^{\uparrow}(X)\}$ and, both cases, imply $\mathtt{prf}(\bar{\sigma}) \subseteq X$. In fact:

$$\bar{\sigma} \in \mathtt{prf}^{\uparrow}(X) \Rightarrow \mathtt{prf}(\bar{\sigma}) \subseteq \mathtt{prf}^{\uparrow}(X) \subseteq X$$
$$\bar{\sigma} \in \{\bar{\sigma} \in \Sigma^{\omega} \mid \mathtt{prf}(\bar{\sigma}) \subseteq \mathtt{prf}^{\uparrow}(X)\} \Rightarrow \mathtt{prf}(\bar{\sigma}) \subseteq \mathtt{prf}^{\uparrow}(X) \subseteq X$$

For all $\bar{\sigma} \notin X$ we have that $\bar{\sigma} \notin \mathtt{prf}^{\uparrow}(X)$ and $\bar{\sigma} \notin \{\bar{\sigma} \in \Sigma^{\omega} \mid \mathtt{prf}(\bar{\sigma}) \subseteq \mathtt{prf}^{\uparrow}(X)\}$. If $\bar{\sigma}$ is finite then $\bar{\sigma} \notin \mathtt{prf}^{\uparrow}(X)$ implies $\mathtt{prf}(\bar{\sigma}) \not\subseteq \mathtt{prf}^{\uparrow}(X)$, since $\bar{\sigma} \in \mathtt{prf}(\bar{\sigma})$. Otherwise $\mathtt{prf}(\bar{\sigma}) \not\subseteq \mathtt{prf}^{\uparrow}(X)$ is obvious. Note that $\mathtt{prf}^{\uparrow}(X) = X \cap \Sigma^{+}$ hence, in both cases, we have $\mathtt{prf}(\bar{\sigma}) \not\subseteq X$. All this means that $X \in \mathsf{Safety}^{\infty}$. Now we prove $\mathsf{Safety}^{\infty} \subseteq \mathfrak{C}_{\Sigma^{\infty}}$. Let $X \in \mathsf{Safety}^{\infty}$. For all $\bar{\sigma} \in X$ we have that $\mathtt{prf}(\bar{\sigma}) \subseteq X$ and hence $\mathtt{prf}(\bar{\sigma}) \subseteq \mathtt{prf}^{\uparrow}(X)$. If $\bar{\sigma}$ is finite then $\bar{\sigma} \in \mathtt{prf}^{\uparrow}(X)$ otherwise $\bar{\sigma} \in \{\bar{\sigma} \in \Sigma^{\omega} \mid \mathtt{prf}(\bar{\sigma}) \subseteq \mathtt{prf}^{\uparrow}(X)\}$, so $\bar{\sigma} \in \mathit{LimPrfCl}(X)$. So $X \in \mathfrak{C}_{\Sigma^{\infty}}$.

Liveness case

First we prove $\mathfrak{D}_{\Sigma^{\infty}} \subseteq \mathsf{Liveness}^{\infty}$. So let $X \in \mathfrak{D}_{\Sigma^{\infty}}$. From the fact that $\mathit{LimPrfCl}(X) = \Sigma^{\infty}$ it follows that $\mathtt{prf}^{\uparrow}(X) = \Sigma^{+}$. This implies $\Sigma^{+} \subseteq \mathtt{prf}^{\uparrow}(X)$ and hence $X \in \mathsf{Liveness}^{\infty}$. Now we prove $\mathsf{Liveness}^{\infty} \subseteq \mathfrak{D}_{\Sigma^{\infty}}$. Let $X \in \mathsf{Liveness}^{\infty}$. We have $\Sigma^{+} \subseteq \mathtt{prf}^{\uparrow}(X)$ and hence $\mathtt{prf}^{\uparrow}(X) = \Sigma^{+}$. The set $\{\bar{\sigma} \in \Sigma^{\omega} \mid \forall \bar{\sigma}' \in \Sigma^{+}.(\bar{\sigma}' \leq^{\mathrm{pf}} \bar{\sigma} \Rightarrow \bar{\sigma}' \in \mathtt{prf}^{\uparrow}(X)\}$ is equal to Σ^{ω} since for every $\bar{\sigma} \in \Sigma^{\omega}$ and $\bar{\sigma}' \in \Sigma^{+}$ we have that $\bar{\sigma}' \not\leq^{\mathrm{pf}} \bar{\sigma}$ or $\bar{\sigma}' \in \mathtt{prf}^{\uparrow}(X) = \Sigma^{+}$. So $\mathit{LimPrfCl}(X) = \Sigma^{+} \cup \Sigma^{\omega} = \Sigma^{\infty}$ and hence $X \in \mathfrak{D}_{\Sigma^{\infty}}$. □

▶ *Lemma 5*

Proof. Note that $\mathtt{slim}^{\omega}(\mathcal{X}) = \{Y \in \wp(\Sigma^\omega) \mid \mathtt{sprf}(Y) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})\}$. We have two cases.

Hypersafety case

First we prove $\mathfrak{C}_{\wp(\Sigma^\omega)} \subseteq \mathsf{HyperSafety}^{\omega}$. Let $\mathcal{X} \in \mathfrak{C}_{\wp(\Sigma^\omega)}$. For all $X \in \mathcal{X}$ we have $\mathtt{sprf}(X) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$ and hence $\mathcal{X} \in \mathsf{HyperSafety}^{\omega}$. For all $X \notin \mathcal{X}$ exists $Y \in \wp(\Sigma^+)$ such that $Y \trianglelefteq^{\mathrm{pf}} X \wedge Y \not\subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$, hence $\mathtt{sprf}(X) \not\subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$ and so $\mathcal{X} \notin \mathsf{HyperSafety}^{\omega}$. Now we prove $\mathsf{HyperSafety}^{\omega} \subseteq \mathfrak{C}_{\wp(\Sigma^\omega)}$. Let $\mathcal{X} \in \mathsf{HyperSafety}^{\omega}$. For all $X \in \mathcal{X}$ we have $\mathtt{sprf}(X) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$ which implies $\mathcal{X} \subseteq \{Y \in \wp(\Sigma^\omega) \mid \mathtt{sprf}(Y) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})\}$. For all $X \notin \mathcal{X}$ we have $\mathtt{sprf}(X) \not\subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$ which implies $X \notin \{Y \in \wp(\Sigma^\omega) \mid \mathtt{sprf}(Y) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})\}$. Hence $\mathcal{X} = \mathit{SlimCl}(\mathcal{X})$ and so $\mathcal{X} \in \mathfrak{C}_{\wp(\Sigma^\omega)}$.

Hyperliveness case

First we prove $\mathfrak{D}_{\wp(\Sigma^\omega)} \subseteq \mathsf{HyperLiveness}^{\omega}$. So let $\mathcal{X} \in \mathfrak{D}_{\wp(\Sigma^\omega)}$. From $\mathit{SlimCl}(\mathcal{X}) = \wp(\Sigma^\omega)$ it follows that $\{Y \in \wp(\Sigma^\omega) \mid \mathtt{sprf}(Y) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})\}$ is equal to $\wp(\Sigma^\omega)$. This means that $\mathtt{sprf}(\Sigma^\omega) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$, which implies that $\wp(\Sigma^+) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$. So $\mathcal{X} \in \mathsf{HyperLiveness}^{\omega}$. Now we prove $\mathsf{HyperLiveness}^{\omega} \subseteq \mathfrak{D}_{\wp(\Sigma^\omega)}$. Let $\mathcal{X} \in \mathsf{HyperLiveness}^{\omega}$, then $\wp(\Sigma^+) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$. This implies that $\forall Y \in \wp(\Sigma^\omega)$ we have $\mathtt{sprf}(Y) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$, namely $\{Y \in \wp(\Sigma^\omega) \mid \mathtt{sprf}(Y) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})\} = \wp(\Sigma^\omega)$. Hence $\mathit{SlimCl}(\mathcal{X}) = \wp(\Sigma^\omega)$ and $\mathcal{X} \in \mathfrak{C}_{\wp(\Sigma^\omega)}$. □

▶ *Lemma 6*

Proof. Note: $\mathit{SlimSprfCl}(\mathcal{X}) = \mathtt{sprf}^{\uparrow}(\mathcal{X}) \cup \{Y \in \wp(\Sigma^\infty) \mid \forall Y' \in \wp(\Sigma^+) \,.\, (Y' \trianglelefteq^{\mathrm{pf}} Y \Rightarrow Y' \in \mathtt{sprf}^{\uparrow}(\mathcal{X})\} = \{Y \in \wp(\Sigma^\infty) \mid \mathtt{sprf}(Y) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})\}$, since if X is in $\mathtt{sprf}^{\uparrow}(X)$ then all $Y \trianglelefteq^{\mathrm{pf}} X$ are in $\mathtt{sprf}^{\uparrow}(X)$ too. The proof is divided in two cases.

Hypersafety case

First we prove $\mathfrak{C}_{\wp(\Sigma^\infty)} \subseteq \mathsf{HyperSafety}^{\infty}$. Let $\mathcal{X} \in \mathfrak{C}_{\wp(\Sigma^\infty)}$. For all $X \in \mathcal{X}$ we have that $X \in \{Y \in \wp(\Sigma^\infty) \mid \mathtt{sprf}(Y) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})\}$ and hence $\mathtt{sprf}(X) \subseteq \mathcal{X}$. In fact, $X \in \{Y \in \wp(\Sigma^\infty) \mid \mathtt{sprf}(Y) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})\}$ implies $\mathtt{sprf}(X) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X}) \subseteq \mathcal{X}$. For all $X \notin \mathcal{X}$ we have that $X \notin \{Y \in \wp(\Sigma^\infty) \mid \mathtt{sprf}(Y) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})\}$. This implies that $\mathtt{sprf}(X) \not\subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X}) = \mathcal{X} \cap \wp(\Sigma^+)$, hence $\mathtt{sprf}(X) \not\subseteq \mathcal{X}$. Hence $\mathcal{X} \in \mathsf{HyperSafety}^{\infty}$. Now we prove $\mathsf{HyperSafety}^{\infty} \subseteq \mathfrak{C}_{\wp(\Sigma^\infty)}$. Let $\mathcal{X} \in \mathsf{HyperSafety}^{\infty}$. For all $X \in \mathcal{X}$ we have $\mathtt{sprf}(X) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$ and so $X \in \mathit{SlimSprfCl}(\mathcal{X})$. For all $X \notin \mathcal{X}$ we have $\mathtt{sprf}(X) \not\subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$ and so $X \notin \mathit{SlimSprfCl}(\mathcal{X})$. Hence $\mathcal{X} \in \mathfrak{C}_{\wp(\Sigma^\infty)}$.

Hyperliveness case

First we prove $\mathfrak{D}_{\wp(\Sigma^\infty)} \subseteq \mathsf{HyperLiveness}^{\infty}$. So let $\mathcal{X} \in \mathfrak{D}_{\wp(\Sigma^\infty)}$. From $\mathit{SlimSprfCl}(\mathcal{X}) = \wp(\Sigma^\infty)$ it follows that $\mathtt{sprf}^{\uparrow}(\mathcal{X}) = \wp(\Sigma^+)$. This implies $\wp(\Sigma^+) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$ and hence $\mathcal{X} \in \mathsf{HyperLiveness}^{\infty}$. Now we prove $\mathsf{HyperLiveness}^{\infty} \subseteq \mathfrak{D}_{\wp(\Sigma^\infty)}$. Let $\mathcal{X} \in \mathsf{HyperLiveness}^{\infty}$. Then we have $\wp(\Sigma^+) \subseteq \mathtt{sprf}^{\uparrow}(\mathcal{X})$ and hence $\mathtt{sprf}^{\uparrow}(\mathcal{X}) = \wp(\Sigma^+)$. Now we can note that the set $\{Y \in \wp(\Sigma^\infty) \mid \forall Y' \in \wp(\Sigma^+) \,.\, (Y' \trianglelefteq^{\mathrm{pf}} Y \Rightarrow Y' \in \mathtt{sprf}^{\uparrow}(\mathcal{X})\}$ is equal to $\wp(\Sigma^\infty)$ since for every $Y \in \wp(\Sigma^\infty)$ and $Y' \in \wp(\Sigma^+)$ we have that $Y' \not\trianglelefteq^{\mathrm{pf}} Y$ or $Y' \in \mathtt{sprf}^{\uparrow}(\mathcal{X}) \subseteq \wp(\Sigma^+)$. So $\mathit{SlimSprfCl}(\mathcal{X}) = \wp(\Sigma^\infty)$ and $\mathcal{X} \in \mathfrak{D}_{\wp(\Sigma^\infty)}$. □

Subsection 6.1.4

▶ *Extended version of Theorem 18*

Theorem 29 (Forward Kleenian Fixpoint Tranfser (Extended))**.** *Let* $\langle F, \mathbb{O}, \bot\rangle$*, with* $\mathbb{O} = \langle \mathcal{O}, \sqsubseteq, \sqcup\rangle$*, and* $\langle F^\sharp, \mathbb{O}^\sharp, \bot^\sharp\rangle$*, with* $\mathbb{O}^\sharp = \langle \mathcal{O}^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp\rangle$*, be concrete and abstract computational fixpoint definitions. Let* $\gamma \in \mathcal{O}^\sharp \to \mathcal{O}$ *be a strict Scott-continuous concretization function satisfying the* commutation condition $\gamma \circ F^\sharp = F \circ \gamma$ *(i.e.* forward completeness*) then:*

- *the respective iterates* F^δ *and* $F^{\sharp\delta}$*, of* F *and* $F^\sharp$ *from* $\bot$ *and* $\bot^\sharp$*, are such that* $F^\delta = \gamma(F^{\sharp\delta})$*, for every ordinal* δ*;*
- $\gamma(\mathsf{lfp}^{\sqsubseteq^\sharp}_{\bot^\sharp} F^\sharp) = \mathsf{lfp}^{\preccurlyeq}_{\bot} F$*;*
- *the iteration order of* $F^\sharp$ *is less than or equal to that of* F*.*

Proof. The proof is very similar to the one of the (backward) Kleenian fixpoint transfer (Theorem 3 of [Cousot, 2002]). Let F^δ and $F^{\sharp\delta}$, for an ordinal δ, be the respective ordinal-termed $\sqsubseteq$-increasing and $\sqsubseteq^\sharp$-increasing ultimately stationary chains of transfinite iterates of F and $F^\sharp$. We have $\gamma(F^{\sharp 0}) = \gamma(\bot^\sharp) = \bot = F^0$ by strictness of γ and definition of the iterates. Assume $\gamma(F^{\sharp\delta}) = F^\delta$ by induction hypothesis. By definition of the iterates, commutation condition and induction hypothesis, we have $\gamma(F^{\sharp\delta+1}) = \gamma(F^\sharp(F^{\sharp\delta})) = F(\gamma(F^{\sharp\delta})) = F(F^\delta) = F^{\delta+1}$. Given a limit ordinal ζ, assume $\gamma(F^{\sharp\delta}) = F^\delta$ for all $\delta < \zeta$. Then by definition of the iterates, continuity of γ and induction hypothesis, $\gamma(F^{\sharp\zeta}) = \gamma(\bigsqcup^\sharp_{\delta<\zeta} F^{\sharp\delta}) = \bigsqcup_{\delta<\zeta} \gamma(F^{\sharp\delta}) = \bigsqcup_{\delta<\zeta} F^\delta = F^\zeta$. By transfinite induction, we conclude that for every ordinal δ we have $\gamma(F^{\sharp\delta}) = F^\delta$. In particular $\gamma(\mathsf{lfp}^{\sqsubseteq^\sharp}_{\bot^\sharp} F^\sharp) = \gamma(F^{\sharp\epsilon}) = \gamma(F^{\sharp\max\{\epsilon,\epsilon'\}}) = F^{\max\{\epsilon,\epsilon'\}} = F^{\epsilon'} = \mathsf{lfp}^{\sqsubseteq}_{\bot} F$, where ϵ and ϵ' are the respective iteration orders. $F^{\sharp\epsilon}$ is a fixpoint of $F^\sharp$ so that by the correspondence between iterates and the commutation condition, we have $F(F^\epsilon) = F(\gamma(F^{\sharp\epsilon})) = \gamma(F^\sharp(F^{\sharp\epsilon})) = \gamma(F^{\sharp\epsilon}) = F^\epsilon$ proving that $\epsilon' \leq \epsilon$. □

Section 7.3

Theorem 30. *For every* $\mathsf{P} \in \mathsf{Imp}$ *and for every* $\mathcal{X} \subseteq \wp(\mathsf{Mem})$*:*

$$\{[\![\mathsf{P}]\!]^\curvearrowright X \mid X \in \mathcal{X}\} \subseteq (\!|\mathsf{P}|\!)^\curvearrowright_{\mathrm{I}} \mathcal{X} \text{ and } \{[\![\mathsf{P}]\!]^\curvearrowright X \mid X \in \mathcal{X}\} \subseteq (\!|\mathsf{P}|\!)^\curvearrowright_{\mathrm{M}} \mathcal{X}$$

Proof. Both hypersemantics $(\!|\mathsf{P}|\!)^\curvearrowright_{\mathrm{I}}$ and $(\!|\mathsf{P}|\!)^\curvearrowright_{\mathrm{M}}$ have the same definition for every command except for loop statements. Hence, the proof for the two hypersemantics differs only for the while construct (indeed, we will use, in the proof, the subscripts I, M only for loops). The proof is for structural induction on P.

Cases $^{\mathsf{i}}\mathsf{skip}^{\mathsf{f}}$, $^{\mathsf{i}}\mathsf{x} := \mathsf{a}^{\mathsf{f}}$ *and* $^{\mathsf{i}}\mathsf{if\ b\ then\ \{\ P_1\ \}\ else\ \{\ P_2\ \}}^{\mathsf{f}}$

We just have to apply the definition of the hypersemantics:

$$\{[\![^{\mathsf{i}}\mathsf{skip}^{\mathsf{f}}]\!]^\curvearrowright X \mid X \in \mathcal{X}\} \subseteq \{[\![^{\mathsf{i}}\mathsf{skip}^{\mathsf{f}}]\!]^\curvearrowright X \mid X \in \mathcal{X}\} = (\!|^{\mathsf{i}}\mathsf{skip}^{\mathsf{f}}|\!)^\curvearrowright \mathcal{X}$$

$$\{[\![^{\mathsf{i}}\mathsf{x} := \mathsf{a}^{\mathsf{f}}]\!]^\curvearrowright X \mid X \in \mathcal{X}\} \subseteq \{\{\mathsf{m}[\mathsf{x} \leftarrowtail n] \mid \mathsf{m} \in X \wedge \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \mathcal{X}\} = (\!|^{\mathsf{i}}\mathsf{x} := \mathsf{a}^{\mathsf{f}}|\!)^\curvearrowright \mathcal{X}$$

$\{[\![{}^{\mathsf{i}}\mathsf{if\ b\ then\ \{\,P_1\,\}\ else\ \{\,P_2\,\}}\,{}^{\mathsf{f}}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\} \subseteq \{[\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\} =$
$= (\!| {}^{\mathsf{i}}\mathsf{if\ b\ then\ \{\,P_1\,\}\ else\ \{\,P_2\,\}}\,{}^{\mathsf{f}} |\!)^{\curvearrowright}\mathcal{X}$

Case ${}^{\mathsf{i}}\mathsf{c}_1\,{}^{\mathsf{l}}\,.\,{}^{\mathsf{k}}\mathsf{c}_2\,{}^{\mathsf{f}}$

$\{[\![{}^{\mathsf{i}}\mathsf{c}_1\,{}^{\mathsf{l}}\,.\,{}^{\mathsf{k}}\mathsf{c}_2\,{}^{\mathsf{f}}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\}$
$=$ ‖ definition of $[\![\cdot]\!]^{\curvearrowright}$
$\{[\![{}^{\mathsf{k}}\mathsf{c}_2\,{}^{\mathsf{f}}]\!]^{\curvearrowright} \circ [\![{}^{\mathsf{i}}\mathsf{c}_1\,{}^{\mathsf{l}}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\}$
$\subseteq$ ‖ inductive hypothesis on c_2
$(\!| {}^{\mathsf{k}}\mathsf{c}_2\,{}^{\mathsf{f}} |\!)^{\curvearrowright}\{[\![{}^{\mathsf{i}}\mathsf{c}_1\,{}^{\mathsf{l}}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\}$
$\subseteq$ ‖ inductive hypothesis on c_1 and monotonicity of $(\!|\cdot|\!)^{\curvearrowright}$
$(\!| {}^{\mathsf{k}}\mathsf{c}_2\,{}^{\mathsf{f}} |\!)^{\curvearrowright} \circ (\!| {}^{\mathsf{i}}\mathsf{c}_1\,{}^{\mathsf{l}} |\!)^{\curvearrowright}\mathcal{X}$
$=$ ‖ definition of $(\!|\cdot|\!)^{\curvearrowright}$
$(\!| {}^{\mathsf{i}}\mathsf{c}_1\,{}^{\mathsf{l}}\,.\,{}^{\mathsf{k}}\mathsf{c}_2\,{}^{\mathsf{f}} |\!)^{\curvearrowright}\mathcal{X}$

Case ${}^{\mathsf{i}}\mathsf{while}\ {}^{\mathsf{k}}\mathsf{b}\ \{\,\mathsf{P}\,\}\,{}^{\mathsf{f}}$ *for* $(\!|\cdot|\!)^{\curvearrowright}_{\mathrm{M}}$

The semantics $[\![{}^{\mathsf{i}}\mathsf{while}\ {}^{\mathsf{k}}\mathsf{b}\ \{\,\mathsf{P}\,\}\,{}^{\mathsf{f}}]\!]^{\curvearrowright} X$ is defined as $[\![\neg\mathsf{b}]\!]^{\curvearrowright}(\mathsf{lfp}^{\subseteq}_{\varnothing}\,\mathsf{F}^{\curvearrowright})$ where $\mathsf{F}^{\curvearrowright}(T) \triangleq X \cup [\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} T$, namely $[\![\neg\mathsf{b}]\!]^{\curvearrowright}\bigcup_{n\in\mathbb{N}} \mathsf{F}^{\curvearrowright n}(\varnothing)$. The iterates of $\mathsf{F}^{\curvearrowright}$ are:

$$\mathsf{F}^{\curvearrowright 0}(\varnothing) = \varnothing \qquad \mathsf{F}^{\curvearrowright 1}(\varnothing) = X \qquad \mathsf{F}^{\curvearrowright 2}(\varnothing) = X \cup [\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X$$
$$\mathsf{F}^{\curvearrowright 3}(\varnothing) = X \cup [\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \qquad \ldots$$

Clearly, the semantics of while coincides with $[\![\neg\mathsf{b}]\!]^{\curvearrowright}\bigcup\{([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^n X \mid n \in \mathbb{N}\}$, where $([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^0 X \triangleq X$ and $([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{n+1} X \triangleq ([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^n X$. Then, by distributivity of the union, the semantics of while is equivalent to $\bigcup\{[\![\neg\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^n X \mid n \in \mathbb{N}\}$.

By definition, $(\!| {}^{\mathsf{i}}\mathsf{while}\ {}^{\mathsf{k}}\mathsf{b}\ \{\,\mathsf{P}\,\}\,{}^{\mathsf{f}} |\!)^{\curvearrowright}_{\mathrm{M}}\mathcal{X} = (\!|\neg\mathsf{b}|\!)^{\curvearrowright}(\mathsf{lfp}^{\subseteq}_{\varnothing}\,\mathsf{H}^{\curvearrowright}_{\mathrm{M}})$ where the semantic operator is $\mathsf{H}^{\curvearrowright}_{\mathrm{M}}(\mathcal{T}) \triangleq \mathcal{X} \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} T \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright} T \mid T \in \mathcal{T}\}$, namely $(\!|\neg\mathsf{b}|\!)^{\curvearrowright}\bigcup_{n\in\mathbb{N}} \mathsf{H}^{\curvearrowright n}_{\mathrm{M}}(\varnothing)$. The iterates of the semantic operator $\mathsf{H}^{\curvearrowright}_{\mathrm{M}}$ are:

$$\mathsf{H}^{\curvearrowright 0}_{\mathrm{M}}(\varnothing) = \varnothing \qquad \mathsf{H}^{\curvearrowright 1}_{\mathrm{M}}(\varnothing) = \mathcal{X}$$
$$\mathsf{H}^{\curvearrowright 2}_{\mathrm{M}}(\varnothing) = \mathcal{X} \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\}$$
$$\begin{aligned}\mathsf{H}^{\curvearrowright 3}_{\mathrm{M}}(\varnothing) &= \mathcal{X} \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\} \cup \\ &\quad \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\}\} = \\ &= \mathcal{X} \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\} \cup \\ &\quad \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright} X) \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright} X) \mid X \in \mathcal{X}\} = \\ &= \mathcal{X} \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\} \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\} \cup \\ &\quad \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\neg\mathsf{P}]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\} =\end{aligned}$$

$$= \mathcal{X} \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}X \mid X \in \mathcal{X}\} \cup$$
$$\cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}X \mid X \in \mathcal{X}\} \dots$$

The hypersemantics of while coincides with

$$(\!|\neg\mathsf{b}|\!)^{\curvearrowright}(\mathcal{X} \cup \{\textstyle\bigcup_{k\leq n}\{([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k+1}X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k}X \mid X \in \mathcal{X}\} \mid n \in \mathbb{N}\})$$

Then, by simple algebraic manipulations, we have that

$$\begin{array}{ll}
& (\!|\neg\mathsf{b}|\!)^{\curvearrowright}(\mathcal{X} \cup \{\textstyle\bigcup_{k\leq n}\{([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k+1}X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k}X \mid X \in \mathcal{X}\} \mid n \in \mathbb{N}\}) \\
= & \quad \| \text{ distributivity of } \cup \\
& (\!|\neg\mathsf{b}|\!)^{\curvearrowright}\mathcal{X} \cup (\!|\neg\mathsf{b}|\!)^{\curvearrowright}\{\textstyle\bigcup_{k\leq n}\{([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k+1}X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k}X \mid X \in \mathcal{X}\} \mid n \in \mathbb{N}\} \\
= & \quad \| \text{ definition of } (\!|\mathsf{b}|\!)^{\curvearrowright} \\
& \{[\![\neg\mathsf{b}]\!]^{\curvearrowright}X \mid X \in \mathcal{X}\} \cup \{[\![\neg\mathsf{b}]\!]^{\curvearrowright}\textstyle\bigcup_{k\leq n}\{([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k+1}X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k}X \mid X \in \mathcal{X}\} \mid n \in \mathbb{N}\} \\
= & \quad \| \text{ distributivity of } \cup \text{ and idempotence of } [\![\neg\mathsf{b}]\!]^{\curvearrowright} \\
& \{[\![\neg\mathsf{b}]\!]^{\curvearrowright}X \mid X \in \mathcal{X}\} \cup \{\textstyle\bigcup_{k\leq n}\{[\![\neg\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k+1}X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k}X \mid X \in \mathcal{X}\} \mid n \in \mathbb{N}\}
\end{array}$$

Finally, it is easy to note that for every possible $n \in \mathbb{N}$ and for every possible $X \in \mathcal{X}$, the set $\bigcup\{[\![\neg\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{n}X \mid n \in \mathbb{N}\}$ is in $\{[\![\neg\mathsf{b}]\!]^{\curvearrowright}X \mid X \in \mathcal{X}\} \cup \{\bigcup_{k\leq n}\{[\![\neg\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k+1}X \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^{k}X \mid X \in \mathcal{X}\} \mid n \in \mathbb{N}\}$. This proves that $\{[\![\overset{\mathsf{i}}{}\mathsf{while}\ \overset{\mathsf{k}}{}\mathsf{b}\ \{\ \mathsf{P}\ \}\ \overset{\mathsf{f}}{}]\!]^{\curvearrowright}X \mid X \in \mathcal{X}\}$ is contained in $(\!|\ \overset{\mathsf{i}}{}\mathsf{while}\ \overset{\mathsf{k}}{}\mathsf{b}\ \{\ \mathsf{P}\ \}\ \overset{\mathsf{f}}{}\ |\!)^{\curvearrowright}_{\mathrm{M}}\mathcal{X}$.

Case $\overset{\mathsf{i}}{}\mathsf{while}\ \overset{\mathsf{k}}{}\mathsf{b}\ \{\ \mathsf{P}\ \}\ \overset{\mathsf{f}}{}$ *for* $(\!|\cdot|\!)^{\curvearrowright}_{\mathrm{I}}$

In order to simplify the notation, we will often apply silently the following fact: $(\!|\mathsf{b}|\!)^{\curvearrowright}(\{\varnothing\} \cup \mathcal{X}) = (\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{X}$. The proof is straightforward, indeed, $(\!|\mathsf{b}|\!)^{\curvearrowright}(\{\varnothing\} \cup \mathcal{X}) = (\!|\mathsf{b}|\!)^{\curvearrowright}\{\varnothing\} \cup (\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{X} = \{[\![\mathsf{b}]\!]^{\curvearrowright}\varnothing\} \setminus \{\varnothing\} \cup (\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{X} = \varnothing \cup (\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{X} = (\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{X}$. By definition, $(\!|\ \overset{\mathsf{i}}{}\mathsf{while}\ \overset{\mathsf{k}}{}\mathsf{b}\ \{\ \mathsf{P}\ \}\ \overset{\mathsf{f}}{}\ |\!)^{\curvearrowright}_{\mathrm{I}}\mathcal{X} = (\!|\neg\mathsf{b}|\!)^{\curvearrowright}(\mathsf{lfp}^{\subseteq}_{\varnothing}\,\mathsf{H}^{\curvearrowright}_{\mathrm{I}})$ where the semantic operator is $\mathsf{H}^{\curvearrowright}_{\mathrm{I}}(\mathcal{T}) \triangleq \{\varnothing\} \cup (\mathcal{X} \boxplus (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{T})$, namely $(\!|\neg\mathsf{b}|\!)^{\curvearrowright}\bigcup_{n\in\mathbb{N}}\mathsf{H}^{\curvearrowright n}_{\mathrm{I}}(\varnothing)$. The iterates of the semantic operator $\mathsf{H}^{\curvearrowright}_{\mathrm{I}}$ are:

$$\mathsf{H}^{\curvearrowright 0}_{\mathrm{I}}(\varnothing) = \varnothing \qquad \mathsf{H}^{\curvearrowright 1}_{\mathrm{I}}(\varnothing) = \{\varnothing\} \qquad \mathsf{H}^{\curvearrowright 2}_{\mathrm{I}}(\varnothing) = \{\varnothing\} \cup \mathcal{X}$$
$$\mathsf{H}^{\curvearrowright 3}_{\mathrm{I}}(\varnothing) = \{\varnothing\} \cup (\mathcal{X} \boxplus (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{X}) = \{\varnothing\} \cup \{X \cup Y \mid X \in \mathcal{X} \wedge Y \in (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{X}\}$$
$$\begin{array}{l}
\mathsf{H}^{\curvearrowright 4}_{\mathrm{I}}(\varnothing) = \{\varnothing\} \cup (\mathcal{X} \boxplus (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright}\{X \cup Y \mid X \in \mathcal{X} \wedge Y \in (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{X}\}) = \\
\qquad = \{\varnothing\} \cup \{X \cup Y \mid X \in \mathcal{X} \wedge Y \in (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright}\{X' \cup Y' \mid X' \in \mathcal{X} \wedge Y' \in (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright}\mathcal{X}\}\} \\
\quad \dots
\end{array}$$

The hypersemantics of while coincides with $(\!|\neg\mathsf{b}|\!)^{\curvearrowright}(\{\varnothing\} \cup \bigcup_{n\in\mathbb{N}}(\mathcal{X} \boxplus (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright})^{n}\mathcal{X}$, where $(\mathcal{X} \boxplus (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright})^{0}\mathcal{X} \triangleq \mathcal{X}$ and $(\mathcal{X} \boxplus (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright})^{n+1} \triangleq (\mathcal{X} \boxplus (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright})(\mathcal{X} \boxplus (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright})^{n}\mathcal{X}$. Then, by simple algebraic manipulations, we have that

$$\begin{array}{ll}
& (\!|\neg\mathsf{b}|\!)^{\curvearrowright}(\{\varnothing\} \cup \textstyle\bigcup_{n\in\mathbb{N}}(\mathcal{X} \boxplus (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}}(\!|\mathsf{b}|\!)^{\curvearrowright})^{n}\mathcal{X} \\
= & \quad \| \text{ definition of } (\!|\mathsf{b}|\!)^{\curvearrowright}
\end{array}$$

$(\!|\neg \mathsf{b}|\!)^{\curvearrowright} \bigcup_{n\in\mathbb{N}} (\mathcal{X} \uplus (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathrm{I}} (\!|\mathsf{b}|\!)^{\curvearrowright})^n \mathcal{X}$
$\subseteq$ $\|$ inductive hypothesis on P and definition of $(\!|\mathsf{b}|\!)^{\curvearrowright}$
$(\!|\neg \mathsf{b}|\!)^{\curvearrowright} \bigcup_{n\in\mathbb{N}} (\lambda\mathcal{Y}\,.\,\mathcal{X} \uplus \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} Y \mid Y \in \mathcal{Y}\})^n \mathcal{X}$
$\subseteq$ $\|$ and definition of $\uplus$
$(\!|\neg \mathsf{b}|\!)^{\curvearrowright} \bigcup_{n\in\mathbb{N}} (\lambda\mathcal{Y}\,.\,\{X \cup [\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} Y \mid X \in \mathcal{X} \wedge Y \in \mathcal{Y}\})^n \mathcal{X}$

Remark. Note that this hypersemantics performs unions for every possible combination, at every recursive step. For example, with $n = 2$ we have:

$$\{X \cup [\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} Y \cup [\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} Z \mid X, Y, Z \in \mathcal{X}\}$$

Finally, it is easy to observe that for every possible $n \in \mathbb{N}$ and for every $X \in \mathcal{X}$, the set $\bigcup\{([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^n X \mid n \in \mathbb{N}\}$ is in $\bigcup_{n\in\mathbb{N}} (\lambda\mathcal{Y}\,.\,\{X \cup [\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} Y \mid X \in \mathcal{X} \wedge Y \in \mathcal{Y}\})^n \mathcal{X}$. Then, by definition of $(\!|\mathsf{b}|\!)^{\curvearrowright}$, we have that $\bigcup\{[\![\neg \mathsf{b}]\!]^{\curvearrowright}([\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright})^n X \mid n \in \mathbb{N}\}$ is in $(\!|\neg \mathsf{b}|\!)^{\curvearrowright} \bigcup_{n\in\mathbb{N}} (\lambda\mathcal{Y}\,.\,\{X \cup [\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} Y \mid X \in \mathcal{X} \wedge Y \in \mathcal{Y}\})^n \mathcal{X}$. This proves that $\{[\![{}^{\mathsf{i}}\mathsf{while}\ {}^{\mathsf{k}}\mathsf{b}\ \{\,\mathsf{P}\,\}^{\mathsf{f}}]\!]^{\curvearrowright} X \mid X \in \mathcal{X}\}$ is contained in $(\!|\, {}^{\mathsf{i}}\mathsf{while}\ {}^{\mathsf{k}}\mathsf{b}\ \{\,\mathsf{P}\,\}^{\mathsf{f}} |\!|^{\curvearrowright}_{\mathrm{I}} \mathcal{X}$.

□

► *Theorem 20*

Proof. Trivial application of Theorem 30, with $\mathcal{X} = \{X\}$. □

Subsection 8.1.3

Assumption 1. *The abstract arithmetic operations* $\oplus^\rho \in \mathtt{C}^\rho \times \mathtt{C}^\rho \to \mathtt{C}^\rho$, *with* $\oplus \in \{+, -, *\}$, *are sound:*

$$\{\{n \oplus m \mid n \in X \wedge m \in Y\} \mid X \in \gamma_{h^\rho}(c_1) \wedge Y \in \gamma_{h^\rho}(c_2)\} \subseteq \gamma_{h^\rho}(c_1 \oplus^\rho c_2)$$

Lemma 7. *The abstract hypersemantics for arithmetic expression* $(\!|\mathsf{a}|\!)^{\curvearrowright}_\rho$ *is sound:*

$$\{\{n \mid \exists \mathsf{m} \in X\,.\,\langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \subseteq \gamma_{h^\rho} (\!|\mathsf{a}|\!)^{\curvearrowright}_\rho \mathfrak{m}$$

Proof. The proof is for structural induction on a.

Case n

$\{\{n \mid \exists \mathsf{m} \in X\,.\,\langle \mathsf{n}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}$
$=$ $\|$ definition of $\Downarrow^{\mathbb{Z}}$
$\{\{n\}\}$
$\subseteq$ $\|$ extensivity of $\gamma_{h^\rho}\alpha_{h^\rho}$
$\gamma_{h^\rho}\alpha_{h^\rho}(\{\{n\}\})$
$=$ $\|$ definition of $(\!|\cdot|\!)^{\curvearrowright}_\rho$
$\gamma_{h^\rho} (\!|\mathsf{n}|\!)^{\curvearrowright}_\rho \mathfrak{m}$

Case x

$$
\begin{array}{ll}
& \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{x}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
= & \quad \| \text{ definition of } \Downarrow^{\mathbb{Z}} \\
& \{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
\subseteq & \quad \| \text{ extensivity of } \gamma_{h^\rho}\alpha_{h^\rho} \\
& \gamma_{h^\rho}\alpha_{h^\rho}(\{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}) \\
= & \quad \| \text{ definition of } \gamma_{\mathfrak{m}} \\
& \gamma_{h^\rho}\alpha_{h^\rho}(\gamma_{h^\rho}(\mathfrak{m}(\mathsf{x}))) \\
\subseteq & \quad \| \text{ reductivity of } \alpha_{h^\rho}\gamma_{h^\rho} \\
& \gamma_{h^\rho}(\mathfrak{m}(\mathsf{x})) \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright}_{\rho} \\
& \gamma_{h^\rho}(\!|\mathsf{x}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}
\end{array}
$$

Case (a)

$$
\begin{array}{ll}
& \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle (\mathsf{a}), \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
= & \quad \| \text{ definition of } \Downarrow^{\mathbb{Z}} \\
& \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
\subseteq & \quad \| \text{ inductive hypothesis} \\
& \gamma_{h^\rho}(\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}
\end{array}
$$

Case $\mathsf{a}_1 \oplus \mathsf{a}_2$*, with* $\oplus \in \{+, -, *\}$

$$
\begin{array}{ll}
& \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_1 \oplus \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
= & \quad \| \text{ definition of } \Downarrow^{\mathbb{Z}} \\
& \{\{n_1 \oplus n_2 \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
\subseteq & \quad \| \text{ set theory} \\
& \{\{n_1 \oplus n_2 \mid \exists \mathsf{m}, \mathsf{m}' \in X \,.\, \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m}'\rangle \Downarrow^{\mathbb{Z}} n_2\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
= & \quad \| \text{ set theory} \\
& \{\{n_1 \oplus n_2 \mid n_1 \in \{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \wedge n_2 \in \{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\}\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
\subseteq & \quad \| \text{ set theory} \\
& \left\{\{n_1 \oplus n_2 \mid n_1 \in X \wedge n_2 \in Y\} \;\middle|\; \begin{array}{l} X \in \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \wedge \\ Y \in \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \end{array}\right\} \\
\subseteq & \quad \| \text{ inductive hypothesis on } \mathsf{a}_1, \mathsf{a}_2 \\
& \{\{n_1 \oplus n_2 \mid n_1 \in X \wedge n_2 \in Y\} \mid X \in \gamma_{h^\rho}(\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \wedge Y \in \gamma_{h^\rho}(\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m}\} \\
\subseteq & \quad \| \text{ soundness of } \oplus^{\rho} \\
& \gamma_{h^\rho}((\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \oplus^{\rho} (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})
\end{array}
$$

$=$ $\quad \|$ definition of $(\!|\cdot|\!)^{\curvearrowright}_{\rho}$

$\gamma_{h^\rho}(\!|\mathsf{a}_1 \oplus \mathsf{a}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$

□

Assumption 2. *The abstract logical operations* $\bowtie^\rho \in \mathtt{C}^\rho \times \mathtt{C}^\rho \to \mathtt{C}^\rho \times \mathtt{C}^\rho$*, with* $\bowtie \in \{=, \neq, <, \leq\}$*, are sound:*

$$\begin{array}{l} \textit{let } \mathcal{X} = \{\{\langle n, m\rangle \mid n \in X \wedge m \in Y \wedge n \bowtie m\} \mid X \in \gamma_{h^\rho}(c_1) \wedge Y \in \gamma_{h^\rho}(c_2)\} \textit{ in} \\ \langle\{\{n \mid \langle n, m\rangle \in X\} \mid X \in \mathcal{X}\}, \{\{m \mid \langle n, m\rangle \in X\} \mid X \in \mathcal{X}\}\rangle \subseteq^2 \gamma^2_{h^\rho}(c_1 \bowtie^\rho c_2) \end{array}$$

Lemma 8. *The abstract hypersemantics for boolean expression* $(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}$ *is sound:*

$$(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) = \{\{\mathsf{m} \in X \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathfrak{m}}(\mathsf{m})\} \setminus \{\varnothing\} \subseteq \gamma_{\mathfrak{m}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$$

Proof. The proof is for structural induction on b.

Case $\mathtt{tt}$

$\{\{\mathsf{m} \in X \mid \langle \mathtt{tt}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathfrak{m}}(\mathsf{m})\} \setminus \{\varnothing\}$

$=$ $\quad \|$ definition of $\Downarrow^{\mathbb{B}}$

$\gamma_{\mathfrak{m}}(\mathfrak{m})$

$=$ $\quad \|$ definition of $(\!|\cdot|\!)^{\curvearrowright}_{\rho}$

$\gamma_{\mathfrak{m}}(\!|\mathtt{tt}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$

Case $\mathtt{ff}$

$\{\{\mathsf{m} \in X \mid \langle \mathtt{ff}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathfrak{m}}(\mathsf{m})\} \setminus \{\varnothing\}$

$=$ $\quad \|$ definition of $\Downarrow^{\mathbb{B}}$

$\{\varnothing\} \setminus \{\varnothing\}$

$=$ $\quad \|$ set theory

$\varnothing$

$=$ $\quad \|$ definition of $\gamma_{\mathfrak{m}}$

$\gamma_{\mathfrak{m}}(\mathfrak{m}_\bot)$

$=$ $\quad \|$ definition of $(\!|\cdot|\!)^{\curvearrowright}_{\rho}$

$\gamma_{\mathfrak{m}}(\!|\mathtt{ff}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$

Case (b)

$\{\{\mathsf{m} \in X \mid \langle (\mathsf{b}), \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathfrak{m}}(\mathsf{m})\} \setminus \{\varnothing\}$

$=$ $\quad \|$ definition of $\Downarrow^{\mathbb{B}}$

$\{\{\mathsf{m} \in X \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathfrak{m}}(\mathsf{m})\} \setminus \{\varnothing\}$

$\subseteq$ $\quad \|$ inductive hypothesis

$\gamma_{\mathfrak{m}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$

Case $\mathsf{b}_1 \wedge \mathsf{b}_2$

In order to prove this case we prove the equivalent formulation:

$$\forall \mathsf{x} \in \mathsf{Var}\,.\,(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq (\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$$

Take an arbitrary $\mathsf{x} \in \mathsf{Var}$. Then we have:

$(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x})$
$= \quad \|$ definition of $(\!|\mathsf{b}|\!)^{\curvearrowright}$
$(\alpha_{\mathfrak{m}}(\{\{[\![\mathsf{b}_1]\!]^{\curvearrowright}X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright}X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \setminus \{\varnothing\}))(\mathsf{x})$
$= \quad \|$ definition of $\alpha_{\mathfrak{m}}$
$(\alpha_{h^{\rho}}(\{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright}X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \setminus \{\varnothing\}))(\mathsf{x})$

The proof continues by cases, recalling that $(\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = (\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) \barwedge (\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$.

Case $(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \bot$

Since $\bot$ is the minimum, $\bot \trianglelefteq (\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$ trivially holds.

Case $(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \bar{\mathfrak{a}}$

$(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \bar{\mathfrak{a}}$ implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright}X\} \subseteq \mathfrak{a}$. Which, in turn, implies that $\exists X_1, X_2 \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X_2\} \neq \varnothing$ and $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright}X_2\} \neq \varnothing$. This excludes the possibility to have $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \bot$ or $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \bot$. In fact, by inductive hypothesis, $(\!|\mathsf{b}_1|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\mathfrak{m}}$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\mathfrak{m}}$. Thus, the only way to falsify the proof is that $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \bar{\mathfrak{a}}$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \bar{\mathfrak{b}}$, with $\mathfrak{a} \neq \mathfrak{b}$ (or the symmetric case). In fact, $\bar{\mathfrak{a}} \barwedge \bar{\mathfrak{b}} = \bot$ which is unsound. So, suppose to be in that case. This means that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\,.\,\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X\} \subseteq \mathfrak{a} \wedge \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X\} \subseteq \mathfrak{b}$. But this implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright}X\}$ is contained in $\mathfrak{a} \cup \mathfrak{b}$ (but not in $\mathfrak{a}$), which is absurd, since we have supposed that it is a subset of $\mathfrak{a}$. All this proves that $\bar{\mathfrak{a}} \trianglelefteq (\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$.

Case $(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \mathcal{A}^{\rho}$

$(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \mathcal{A}^{\rho}$ implies that $\exists X, Y \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright}X\} \subseteq \mathfrak{a}$ and $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}Y \cap [\![\mathsf{b}_2]\!]^{\curvearrowright}Y\} \subseteq \mathfrak{b}$, with $\mathfrak{a} \neq \mathfrak{b}$. As for the previous case, $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) \neq \bot$ or $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) \neq \bot$. The only way to falsify the proof is that $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \bar{\mathfrak{a}}$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) \trianglelefteq \mathcal{A}^{\rho}$ (or the symmetric case). In fact, $\bar{\mathfrak{a}} \barwedge \mathcal{A}^{\rho} = \bar{\mathfrak{a}}$, $\bar{\mathfrak{a}} \barwedge \bar{\mathfrak{a}} = \bar{\mathfrak{a}}$ and $\bar{\mathfrak{a}} \barwedge \bar{\mathfrak{b}} = \bot$ are unsound results. So, suppose $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \bar{\mathfrak{a}}$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \mathcal{A}^{\rho}$. This means that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\,.\,\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X\} \subseteq \mathfrak{a}$ and $\exists X_1, X_2 \in \gamma_{\mathfrak{m}}(\mathfrak{m})\,.\,\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright}X_1\} \subseteq \mathfrak{a} \wedge \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright}X_2\} \subseteq \mathfrak{b}$. But this implies that $\exists X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that the set $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright}X\}$ is contained

in $\mathfrak{a} \cup \mathfrak{b}$ (but not contained in $\mathfrak{a}$), which is absurd, since we have supposed that it is equal to $\mathcal{A}^\rho$. Now, suppose $(\!|b_1|\!)^{\curvearrowright}_\rho \mathfrak{m})(x) = \bar{\mathfrak{a}}$ and $(\!|b_2|\!)^{\curvearrowright}_\rho \mathfrak{m})(x) = \bar{\mathfrak{a}}$. This means that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m}) \,.\, \{m(x) \mid m \in [\![b_1]\!]^{\curvearrowright} X\} \subseteq \mathfrak{a} \supseteq \{m(x) \mid m \in [\![b_2]\!]^{\curvearrowright} X\}$. But this implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{m(x) \mid m \in [\![b_1]\!]^{\curvearrowright} X \cap [\![b_2]\!]^{\curvearrowright} X\}$ is contained in $\mathfrak{a}$, which is absurd, since we have supposed that it is equal to $\mathcal{A}^\rho$. Finally, suppose $(\!|b_1|\!)^{\curvearrowright}_\rho \mathfrak{m})(x) = \bar{\mathfrak{a}}$ and $(\!|b_2|\!)^{\curvearrowright}_\rho \mathfrak{m})(x) = \bar{\mathfrak{b}}$. This means that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m}) \,.\, \{m(x) \mid m \in [\![b_1]\!]^{\curvearrowright} X\} \subseteq \mathfrak{a} \wedge \{m(x) \mid m \in [\![b_2]\!]^{\curvearrowright} X\} \subseteq \mathfrak{b}$. But this implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{m(x) \mid m \in [\![b_1]\!]^{\curvearrowright} X \cap [\![b_2]\!]^{\curvearrowright} X\}$ is contained in $\mathfrak{a} \cup \mathfrak{b}$ (but not in $\mathfrak{a}$ or $\mathfrak{b}$), which is absurd, since we have supposed that it is equal to $\mathcal{A}^\rho$. All this proves that $\mathcal{A}^\rho \trianglelefteq (\!|b_1 \wedge b_2|\!)^{\curvearrowright}_\rho \mathfrak{m})(x)$.

Case $(\alpha_{\mathfrak{m}}(\!|b_1 \wedge b_2|\!)^{\curvearrowright} \gamma_{\mathfrak{m}}(\mathfrak{m}))(x) = \top$

$(\alpha_{\mathfrak{m}}(\!|b_1 \wedge b_2|\!)^{\curvearrowright} \gamma_{\mathfrak{m}}(\mathfrak{m}))(x) = \top$ implies that $\exists X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{m(x) \mid m \in [\![b_1]\!]^{\curvearrowright} X \cap [\![b_2]\!]^{\curvearrowright} X\}$ is contained in $\mathfrak{a} \cup \mathfrak{b}$ (but not in $\mathfrak{a}$ or $\mathfrak{b}$). This trivially implies that for the same X we have $\{m(x) \mid m \in [\![b_1]\!]^{\curvearrowright} X\}$ and $\{m(x) \mid m \in [\![b_2]\!]^{\curvearrowright} X\}$ are contained in $\mathfrak{a} \cup \mathfrak{b}$ (but not in $\mathfrak{a}$ of $\mathfrak{b}$). This is sufficient to state that $(\!|b_1|\!)^{\curvearrowright}_\rho \mathfrak{m})(x) = \top$ and $(\!|b_2|\!)^{\curvearrowright}_\rho \mathfrak{m})(x) = \top$. Hence $\top \trianglelefteq (\!|b_1 \wedge b_2|\!)^{\curvearrowright}_\rho \mathfrak{m})(x)$.

Since the variable x has been chosen arbitrarily, the proof holds for every variable. This terminates the case.

Case $b_1 \vee b_2$

In order to prove this case we prove the equivalent formulation:

$$\forall x \in \mathsf{Var} \,.\, (\alpha_{\mathfrak{m}}(\!|b_1 \vee b_2|\!)^{\curvearrowright} \gamma_{\mathfrak{m}}(\mathfrak{m}))(x) \trianglelefteq (\!|b_1 \vee b_2|\!)^{\curvearrowright}_\rho \mathfrak{m})(x)$$

Take an arbitrary $x \in \mathsf{Var}$. Then we have:

$$\begin{array}{ll} & (\alpha_{\mathfrak{m}}(\!|b_1 \vee b_2|\!)^{\curvearrowright} \gamma_{\mathfrak{m}}(\mathfrak{m}))(x) \\ = & \quad \| \text{ definition of } (\!|b|\!)^{\curvearrowright} \\ & (\alpha_{\mathfrak{m}}(\{\{[\![b_1]\!]^{\curvearrowright} X \cup [\![b_2]\!]^{\curvearrowright} X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \setminus \{\varnothing\}))(x) \\ = & \quad \| \text{ definition of } \alpha_{\mathfrak{m}} \\ & (\alpha_{h^\rho}(\{\{m(x) \mid m \in [\![b_1]\!]^{\curvearrowright} X \cup [\![b_2]\!]^{\curvearrowright} X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \setminus \{\varnothing\}))(x) \end{array}$$

The proof continues by cases, recalling that:

$$(\!|b_1 \vee b_2|\!)^{\curvearrowright}_\rho \mathfrak{m})(x) = \begin{cases} (\!|b_1|\!)^{\curvearrowright}_\rho \mathfrak{m})(x) \veebar (\!|b_2|\!)^{\curvearrowright}_\rho \mathfrak{m})(x) & \text{if } \mathfrak{m}(x) \neq \top \vee x \notin \mathsf{vars}(b_1) \cap \mathsf{vars}(b_1) \\ \top & \text{otherwise} \end{cases}$$

Case $(\alpha_{\mathfrak{m}}(\!|b_1 \vee b_2|\!)^{\curvearrowright} \gamma_{\mathfrak{m}}(\mathfrak{m}))(x) = \bot$

Since $\bot$ is the minimum, $\bot \trianglelefteq (\!|b_1 \vee b_2|\!)^{\curvearrowright}_\rho \mathfrak{m})(x)$ trivially holds.

Case $(\alpha_{\mathfrak{m}}(\!|b_1 \vee b_2|\!)^{\curvearrowright} \gamma_{\mathfrak{m}}(\mathfrak{m}))(x) = \bar{\mathfrak{a}}$

$(\alpha_{\mathfrak{m}}(\!|b_1 \vee b_2|\!)^{\curvearrowright} \gamma_{\mathfrak{m}}(\mathfrak{m}))(x) = \bar{\mathfrak{a}}$ implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{m(x) \mid m \in [\![b_1]\!]^{\curvearrowright} X \cup [\![b_2]\!]^{\curvearrowright} X\} \subseteq \mathfrak{a}$. Which, in turn, implies that $\exists X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{m(x) \mid m \in [\![b_1]\!]^{\curvearrowright} X\} \neq$

$\varnothing$ or $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright} X_2\} \neq \varnothing$, but not necessarily both. This excludes the possibility to have $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \bot$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \bot$. In fact, by inductive hypothesis, $(\!|\mathsf{b}_1|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\mathfrak{m}}$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\mathfrak{m}}$. This is not a problem, since if $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \bot$ then $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$ must be necessarily equal to $\bar{\mathfrak{a}}$ (or the symmetric case). This guarantees that the result $\bot \veebar \bar{\mathfrak{a}} = \bar{\mathfrak{a}}$ is sound. There are not other outcome leading to $\bot$ hence we have that $\bar{\mathfrak{a}} \trianglelefteq (\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$.

Case $(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \mathcal{A}^{\rho}$

$(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \mathcal{A}^{\rho}$ implies that $\exists X, Y \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright} X\} \subseteq \mathfrak{a}$ and $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} Y \cup [\![\mathsf{b}_2]\!]^{\curvearrowright} Y\} \subseteq \mathfrak{b}$, with $\mathfrak{a} \neq \mathfrak{b}$. As for the previous case, $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) \neq \bot$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) \neq \bot$ at the same time and if one of the two is $\bot$ then the other must be $\mathcal{A}^{\rho}$, so $\bot \veebar \mathcal{A}^{\rho} = \mathcal{A}^{\rho}$ is sound. The only way to falsify the proof is that $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \bar{\mathfrak{a}} = (\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$. In fact, $\bar{\mathfrak{a}} \veebar \bar{\mathfrak{a}} = \bar{\mathfrak{a}}$ which is unsound. So, suppose to be in that case. This means that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m}) \,.\, \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X\} \subseteq \mathfrak{a} \supseteq \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright} X\}$. But this implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright} X\} \subseteq \mathfrak{a}$, which is absurd, since we have supposed that it is equal to $\mathcal{A}^{\rho}$. All this proves that $\mathcal{A}^{\rho} \trianglelefteq (\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$.

Case $(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \top$

$(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \top$ implies that $\exists X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright} X\}$ is contained in $\mathfrak{a} \cup \mathfrak{b}$ (but not in $\mathfrak{a}$ or $\mathfrak{b}$). As for the previous cases, $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) \neq \bot$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) \neq \bot$ at the same time and if one of the two is $\bot$ then the other must be $\top$, so $\bot \veebar \top = \top$ is sound. Then, it is easy to note that there exists $X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright} X\}$ is contained in $\mathfrak{a} \cup \mathfrak{b}$ (but not in $\mathfrak{a}$ or $\mathfrak{b}$) if and only if $\mathfrak{m}(\mathsf{x}) = \top$ (indeed, $[\![\mathsf{b}_1]\!]^{\curvearrowright} X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright} X \subseteq X$). In this case, by definition, $(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \top$, hence it is sound.

Since the variable x has been chosen arbitrarily, the proof holds for every variable. This terminates the case.

Case $\mathsf{a}_1 \bowtie \mathsf{a}_2$, *with* $\bowtie \in \{=, \neq, <, \leq\}$

The proof for this case is divided in three parts. First, we have to prove the following preliminary result. Given $\mathfrak{m} \in \mathsf{Mem}^{\rho}$ we have that:

$$\alpha_{\mathfrak{m}}(\textstyle\bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})) \sqsubseteq \mathfrak{m} \tag{A.1}$$

If there exists a variable x such that $\mathfrak{m}(\mathsf{x}) = \bot$, then $\gamma_{\mathfrak{m}}(\mathfrak{m}) = \varnothing$ and so $\alpha_{\mathfrak{m}}(\bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})) = \mathfrak{m}_{\bot} \sqsubseteq \mathfrak{m}$. Otherwise, take an arbitrary variable x. If $\mathfrak{m}(\mathsf{x}) = \bar{\mathfrak{a}}$ then for every $X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ we have that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X\} \subseteq \mathfrak{a}$. But this implies that for every $X' \subseteq X$ we have $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X'\} \subseteq \mathfrak{a}$ as well. Thus, $(\alpha_{\mathfrak{m}}(\bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})))(\mathsf{x}) = \bar{\mathfrak{a}}$. Now suppose $\mathfrak{m}(\mathsf{x}) = \mathcal{A}^{\rho}$. This means that there exist $X, Y \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X\} \subseteq \mathfrak{a}$ and $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in Y\} \subseteq \mathfrak{a}$, with $\mathfrak{a} \neq \mathfrak{b}$. Now we can reason as before: for every $X' \subseteq X$ and $Y' \subseteq Y$ we have $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X'\} \subseteq \mathfrak{a}$ and $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in Y'\} \subseteq \mathfrak{b}$. Thus, $(\alpha_{\mathfrak{m}}(\bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})))(\mathsf{x}) = \mathcal{A}^{\rho}$. Finally, in the case of $\mathfrak{m}(\mathsf{x}) = \top$, $\alpha_{\mathfrak{m}}(\bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})))(\mathsf{x}) \trianglelefteq \mathfrak{m}(\mathsf{x})$ holds. Since the variable x has been chosen arbitrarily, the proof holds for every variable.

Now we retrieve a superset of $(\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^\curvearrowright\gamma_\mathfrak{m}(\mathfrak{m}) = \big\{\{\mathsf{m} \in X \mid \langle \mathsf{a}_1 \bowtie \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \;\big|\; X \in \gamma_\mathfrak{m}(\mathfrak{m})\big\} \setminus \{\varnothing\}$. In the following, we let $\mathbb{D} = \bigcup_{Z\in\gamma_\mathfrak{m}(\mathfrak{m})}(\wp(Z)\setminus\{\varnothing\})$.

$$
\begin{array}{l}
\big\{\{\mathsf{m} \in X \mid \langle \mathsf{a}_1 \bowtie \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \;\big|\; X \in \gamma_\mathfrak{m}(\mathfrak{m})\big\} \setminus \{\varnothing\} \\
= \quad \| \text{ definition of } \Downarrow^{\mathbb{B}} \\
\big\{\{\mathsf{m} \in X \mid \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \wedge n_1 \bowtie n_2\} \;\big|\; X \in \gamma_\mathfrak{m}(\mathfrak{m})\big\} \setminus \{\varnothing\} \\
= \quad \| \text{ set theory and definition of } [\![\mathsf{a}]\!]^\curvearrowright \\
\left\{\left\{\mathsf{m} \in X \;\middle|\; \exists n_1 \in [\![\mathsf{a}_1]\!]^\curvearrowright X\, \exists n_2 \in [\![\mathsf{a}_2]\!]^\curvearrowright X\,.\, \left(\begin{array}{l}\langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \\ \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \wedge \\ n_1 \bowtie n_2\end{array}\right)\right\} \;\middle|\; X \in \gamma_\mathfrak{m}(\mathfrak{m})\right\} \setminus \{\varnothing\} \\
\subseteq \quad \| \text{ set theory and definition of } \mathbb{D} \\
\left\{X \in \mathbb{D} \;\middle|\; \forall \mathsf{m} \in X\, \exists n_1 \in [\![\mathsf{a}_1]\!]^\curvearrowright X\, \exists n_2 \in [\![\mathsf{a}_2]\!]^\curvearrowright X\,.\, \left(\begin{array}{l}\langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \\ \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \wedge \\ n_1 \bowtie n_2\end{array}\right)\right\} \\
\subseteq \quad \| \text{ set theory and definition of } (\!|\mathsf{a}|\!)^\curvearrowright \\
\qquad \text{let } \mathbb{P} = \left\{\left\{\langle n_1, n_2\rangle \;\middle|\; \begin{array}{l} n_1 \in N_1 \wedge \\ n_2 \in N_2 \wedge \\ n_1 \bowtie n_2\end{array}\right\} \;\middle|\; N_1 \in (\!|\mathsf{a}_1|\!)^\curvearrowright\gamma_\mathfrak{m}(\mathfrak{m}) \wedge N_2 \in (\!|\mathsf{a}_2|\!)^\curvearrowright\gamma_\mathfrak{m}(\mathfrak{m})\right\} \text{ in} \\
\big\{X \subseteq \mathbb{D} \;\big|\; \exists P \in \mathbb{P}\, \forall \mathsf{m} \in X\, \exists \langle n_1, n_2\rangle \in P\,.\, (\langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2)\big\} \\
\subseteq \quad \| \text{ soundness of } (\!|\mathsf{a}|\!)^\curvearrowright_\rho \\
\qquad \text{let } \mathbb{P} = \left\{\left\{\langle n_1, n_2\rangle \;\middle|\; \begin{array}{l} n_1 \in N_1 \wedge \\ n_2 \in N_2 \wedge \\ n_1 \bowtie n_2\end{array}\right\} \;\middle|\; N_1 \in \gamma_{h^\rho}(\!|\mathsf{a}_1|\!)^\curvearrowright_\rho\mathfrak{m} \wedge N_2 \in \gamma_{h^\rho}(\!|\mathsf{a}_2|\!)^\curvearrowright_\rho\mathfrak{m}\right\} \text{ in} \\
\big\{X \subseteq \mathbb{D} \;\big|\; \exists P \in \mathbb{P}\, \forall \mathsf{m} \in X\, \exists \langle n_1, n_2\rangle \in P\,.\, (\langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2)\big\} \\
\subseteq \quad \| \text{ soundness of } \bowtie^{\text{cprp}} \text{ (i.e. } \mathbb{P}\subseteq\{\{\langle n,m\rangle \mid n\in X\wedge m\in Y\} \mid \langle\mathcal{X},\mathcal{Y}\rangle=\gamma^2_{h^\rho}((\!|\mathsf{a}_1|\!)^\curvearrowright_\rho\mathfrak{m}\bowtie^{\text{cprp}}(\!|\mathsf{a}_2|\!)^\curvearrowright_\rho\mathfrak{m})\wedge X\in\mathcal{X}\wedge Y\in\mathcal{Y}\}) \\
\qquad \text{let } \mathbb{P}' = \left\{\big\{\langle n, m\rangle \;\big|\; n \in X \wedge m \in Y\big\} \;\middle|\; \begin{array}{l}\langle\mathcal{X},\mathcal{Y}\rangle = \gamma^2_{h^\rho}((\!|\mathsf{a}_1|\!)^\curvearrowright_\rho\mathfrak{m} \bowtie^{\text{cprp}} (\!|\mathsf{a}_2|\!)^\curvearrowright_\rho\mathfrak{m}) \\ \wedge X \in \mathcal{X} \wedge Y \in \mathcal{Y}\end{array}\right\} \text{ in} \\
\big\{X \subseteq \mathbb{D} \;\big|\; \exists P \in \mathbb{P}'\, \forall \mathsf{m} \in X\, \exists \langle n_1, n_2\rangle \in P\,.\, (\langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2)\big\} \\
\subseteq \quad \| \text{ set theory and definition of } [\![\mathsf{a}]\!]^\curvearrowright \\
\qquad \text{let } \langle c_1, c_2\rangle = (\!|\mathsf{a}_1|\!)^\curvearrowright_\rho\mathfrak{m} \bowtie^{\text{cprp}} (\!|\mathsf{a}_2|\!)^\curvearrowright_\rho\mathfrak{m} \text{ in} \\
\big\{X \subseteq \mathbb{D} \;\big|\; [\![\mathsf{a}_1]\!]^\curvearrowright X \in \gamma_{h^\rho}(c_1) \wedge [\![\mathsf{a}_2]\!]^\curvearrowright X \in \gamma_{h^\rho}(c_2)\big\} \\
\subseteq \quad \| \text{ set theory and definition of } (\!|\mathsf{a}|\!)^\curvearrowright \\
\qquad \text{let } \langle c_1, c_2\rangle = (\!|\mathsf{a}_1|\!)^\curvearrowright_\rho\mathfrak{m} \bowtie^{\text{cprp}} (\!|\mathsf{a}_2|\!)^\curvearrowright_\rho\mathfrak{m} \text{ in} \\
\bigcup\big\{\mathcal{X} \subseteq \mathbb{D} \;\big|\; (\!|\mathsf{a}_1|\!)^\curvearrowright\mathcal{X} \subseteq \gamma_{h^\rho}(c_1) \wedge (\!|\mathsf{a}_2|\!)^\curvearrowright\mathcal{X} \subseteq \gamma_{h^\rho}(c_2)\big\} \\
\subseteq \quad \| \text{ set theory} \\
\bigcup\{\mathcal{X} \cap \mathcal{Y} \mid \mathcal{X}, \mathcal{Y} \subseteq \mathbb{D} \wedge (\!|\mathsf{a}_1|\!)^\curvearrowright\mathcal{X} \subseteq \gamma_{h^\rho}(c_1) \wedge (\!|\mathsf{a}_2|\!)^\curvearrowright\mathcal{Y} \subseteq \gamma_{h^\rho}(c_2)\} \\
= \quad \| \text{ distributivity of } \cap \\
\bigcup\{\mathcal{X} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_1|\!)^\curvearrowright\mathcal{X} \subseteq \gamma_{h^\rho}(c_1)\} \cap \bigcup\{\mathcal{Y} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_2|\!)^\curvearrowright\mathcal{Y} \subseteq \gamma_{h^\rho}(c_2)\}
\end{array}
$$

Hence $(\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\sqcap\!\rightarrow}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \bigcup\{\mathcal{X} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_1|\!)^{\sqcap\!\rightarrow}\mathcal{X} \subseteq \gamma_{h^\rho}(c_1)\} \cap \bigcup\{\mathcal{Y} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_2|\!)^{\sqcap\!\rightarrow}\mathcal{Y} \subseteq \gamma_{h^\rho}(c_2)\}$. Now, we can finally prove the soundness for this case.

$$
\begin{array}{ll}
& \gamma_{\mathfrak{m}}((\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\sqcap\!\rightarrow}_{\rho}\mathfrak{m}) \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\sqcap\!\rightarrow}_{\rho} \\
& \gamma_{\mathfrak{m}}(\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_1|\!)^{\sqcap\!\rightarrow}_{\rho}\mathfrak{n} \trianglelefteq c_1\} \sqcap \bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_2|\!)^{\sqcap\!\rightarrow}_{\rho}\mathfrak{n} \trianglelefteq c_2\}) \\
= & \quad \| \text{ co-aditivity of } \gamma_{\mathfrak{m}} \\
& \gamma_{\mathfrak{m}}(\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_1|\!)^{\sqcap\!\rightarrow}_{\rho}\mathfrak{n} \trianglelefteq c_1\}) \cap \gamma_{\mathfrak{m}}(\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_2|\!)^{\sqcap\!\rightarrow}_{\rho}\mathfrak{n} \trianglelefteq c_2\}) \\
\supseteq & \quad \| \text{ monotonicity of } \gamma_{\mathfrak{m}} \\
& \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \mid \mathfrak{n} \sqsubseteq \mathfrak{m} \wedge (\!|\mathsf{a}_1|\!)^{\sqcap\!\rightarrow}_{\rho}\mathfrak{n} \trianglelefteq c_1\} \cap \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \mid \mathfrak{n} \sqsubseteq \mathfrak{m} \wedge (\!|\mathsf{a}_2|\!)^{\sqcap\!\rightarrow}_{\rho}\mathfrak{n} \trianglelefteq c_2\} \\
\supseteq & \quad \| \text{ monotonicity of } \gamma_{h^\rho} \text{ and soundness of } (\!|\mathsf{a}|\!)^{\sqcap\!\rightarrow}_{\rho} \\
& \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \mid \mathfrak{n} \sqsubseteq \mathfrak{m} \wedge (\!|\mathsf{a}_1|\!)^{\sqcap\!\rightarrow}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^\rho}(c_1)\} \cap \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \mid \mathfrak{n} \sqsubseteq \mathfrak{m} \wedge (\!|\mathsf{a}_2|\!)^{\sqcap\!\rightarrow}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^\rho}(c_2)\} \\
\supseteq & \quad \| \text{ set theory and } \alpha_{\mathfrak{m}}(\mathbb{D}) \sqsubseteq \mathfrak{m} \\
& \bigcup\left\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \;\middle|\; \begin{array}{l} \mathfrak{n} \sqsubseteq \alpha_{\mathfrak{m}}(\mathbb{D}) \wedge \\ (\!|\mathsf{a}_1|\!)^{\sqcap\!\rightarrow}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^\rho}(c_1) \end{array}\right\} \cap \bigcup\left\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \;\middle|\; \begin{array}{l} \mathfrak{n} \sqsubseteq \alpha_{\mathfrak{m}}(\mathbb{D}) \wedge \\ (\!|\mathsf{a}_2|\!)^{\sqcap\!\rightarrow}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^\rho}(c_2) \end{array}\right\} \\
= & \quad \| \text{ Galois connection } \alpha_{\mathfrak{m}}, \gamma_{\mathfrak{m}} \\
& \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \mathbb{D} \mid (\!|\mathsf{a}_1|\!)^{\sqcap\!\rightarrow}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^\rho}(c_1)\} \cap \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \mathbb{D} \mid (\!|\mathsf{a}_2|\!)^{\sqcap\!\rightarrow}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^\rho}(c_2)\}) \\
= & \\
& \bigcup\{\mathcal{X} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_1|\!)^{\sqcap\!\rightarrow}\mathcal{X} \subseteq \gamma_{h^\rho}(c_1)\} \cap \bigcup\{\mathcal{X} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_2|\!)^{\sqcap\!\rightarrow}\mathcal{X} \subseteq \gamma_{h^\rho}(c_2)\})
\end{array}
$$

Hence we have that $\bigcup\{\mathcal{X} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_1|\!)^{\sqcap\!\rightarrow}\mathcal{X} \subseteq \gamma_{h^\rho}(c_1)\} \cap \bigcup\{\mathcal{Y} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_2|\!)^{\sqcap\!\rightarrow}\mathcal{Y} \subseteq \gamma_{h^\rho}(c_2)\} \subseteq \gamma_{\mathfrak{m}}((\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\sqcap\!\rightarrow}_{\rho}\mathfrak{m})$ which, in turn, implies $(\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\sqcap\!\rightarrow}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}((\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\sqcap\!\rightarrow}_{\rho}\mathfrak{m})$ as requested. □

▶ *Theorem 24*

Proof. We just have to prove that the abstract hypersemantics $(\!|\mathsf{P}|\!)^{\sqcap\!\rightarrow}_{\rho}$ approximates the best correct approximation of the concrete hypersemantics $(\!|\mathsf{P}|\!)^{\sqcap\!\rightarrow}$ in Mem^{ρ}, namely $\alpha_{\mathfrak{m}} \circ (\!|\mathsf{P}|\!)^{\sqcap\!\rightarrow} \circ \gamma_{\mathfrak{m}} \sqsubseteq (\!|\mathsf{P}|\!)^{\sqcap\!\rightarrow}_{\rho}$. The proof is for structural induction on P.

Case P *and* $\mathfrak{m} = \mathfrak{m}_{\perp}$

$$
\begin{array}{ll}
& \alpha_{\mathfrak{m}}(\!|\mathsf{P}|\!)^{\sqcap\!\rightarrow}\gamma_{\mathfrak{m}}(\mathfrak{m}_{\perp}) \\
= & \quad \| \text{ definition of } \gamma_{\mathfrak{m}} \\
& \alpha_{\mathfrak{m}}(\!|\mathsf{P}|\!)^{\sqcap\!\rightarrow}\varnothing \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\sqcap\!\rightarrow} \\
& \alpha_{\mathfrak{m}}(\varnothing) \\
= & \quad \| \text{ definition of } \alpha_{\mathfrak{m}} \\
& \mathfrak{m}_{\perp} \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\sqcap\!\rightarrow}_{\rho}
\end{array}
$$

$(\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}_{\perp}$

Case $\overset{\mathsf{i}}{}\mathsf{skip}^{\mathsf{f}}$

$\alpha_{\mathfrak{m}}(\!|\,\overset{\mathsf{i}}{}\mathsf{skip}^{\mathsf{f}}\,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m})$
$=$ $\quad\|$ definition of $(\!|\cdot|\!)^{\curvearrowright}$
$\alpha_{\mathfrak{m}}\gamma_{\mathfrak{m}}(\mathfrak{m})$
$=$ $\quad\|$ reductivity of $\alpha_{\mathfrak{m}}\gamma_{\mathfrak{m}}$
$\mathfrak{m}$
$=$ $\quad\|$ definition of $(\!|\cdot|\!)^{\curvearrowright}_{\rho}$
$(\!|\,\overset{\mathsf{i}}{}\mathsf{skip}^{\mathsf{f}}\,|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}$

Case $\overset{\mathsf{i}}{}\mathsf{x} := \mathsf{a}^{\mathsf{f}}$

$\alpha_{\mathfrak{m}}(\!|\,\overset{\mathsf{i}}{}\mathsf{x} := \mathsf{a}^{\mathsf{f}}\,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m})$
$=$ $\quad\|$ definition of $(\!|\cdot|\!)^{\curvearrowright}$
$\alpha_{\mathfrak{m}}(\{\{\mathsf{m}[\mathsf{x}\leftarrowtail n] \mid \mathsf{m}\in X \wedge \langle \mathsf{a},\mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\})$
$=$ $\quad\|\alpha_{\mathfrak{m}}=\alpha^{\cdot}_{h^{\rho}}\circ\alpha_{nnr}$ and definition of α_{nnr}
$\alpha^{\cdot}_{h^{\rho}} \circ (\lambda \mathsf{y}\,.\,\{\{\mathsf{m}(\mathsf{y}) \mid \mathsf{m}\in X\} \mid X \in \{\{\mathsf{m}[\mathsf{x}\leftarrowtail n] \mid \mathsf{m}\in X \wedge \langle \mathsf{a},\mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}\})$
$=$
$\alpha^{\cdot}_{h^{\rho}} \circ (\lambda \mathsf{y}\,.\,[\![\,\mathsf{y}=\mathsf{x}\,?\,\{\{n \mid \exists \mathsf{m}\in X\,.\,\langle \mathsf{a},\mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X\in\gamma_{\mathfrak{m}}(\mathfrak{m})\}\,:\,\{\{\mathsf{m}(\mathsf{y}) \mid \mathsf{m}\in X\} \mid X\in\gamma_{\mathfrak{m}}(\mathfrak{m})\}\,]\!])$
$\sqsubseteq$ $\quad\|$ soundness of $(\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}$
$\alpha^{\cdot}_{h^{\rho}} \circ (\lambda \mathsf{y}\,.\,[\![\,\mathsf{y}=\mathsf{x}\,?\,\gamma_{h^{\rho}}(\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}\,:\,\{\{\mathsf{m}(\mathsf{y}) \mid \mathsf{m}\in X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}\,]\!])$
$=$ $\quad\|$ definition of $\alpha^{\cdot}_{h^{\rho}}$
$\lambda \mathsf{y}\,.\,[\![\,\mathsf{y}=\mathsf{x}\,?\,\alpha_{h^{\rho}}\gamma_{h^{\rho}}(\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}\,:\,\alpha_{h^{\rho}}(\{\{\mathsf{m}(\mathsf{y}) \mid \mathsf{m}\in X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\})\,]\!]$
$=$ $\quad\|$ definition of $\gamma_{\mathfrak{m}}$
$\lambda \mathsf{y}\,.\,[\![\,\mathsf{y}=\mathsf{x}\,?\,\alpha_{h^{\rho}}\gamma_{h^{\rho}}(\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}\,:\,\alpha_{h^{\rho}}\gamma_{h^{\rho}}(\mathfrak{m}(\mathsf{y}))\,]\!]$
$\sqsubseteq$ $\quad\|$ reductivity of $\alpha_{h^{\rho}}\gamma_{h^{\rho}}$
$\lambda \mathsf{y}\,.\,[\![\,\mathsf{y}=\mathsf{x}\,?\,(\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}\,:\,\mathfrak{m}(\mathsf{y})\,]\!]$
$=$
$\mathfrak{m}[\mathsf{x}\leftarrowtail (\!|\mathsf{a}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}]$
$=$ $\quad\|$ definition of $(\!|\cdot|\!)^{\curvearrowright}_{\rho}$
$(\!|\,\overset{\mathsf{i}}{}\mathsf{x} := \mathsf{a}^{\mathsf{f}}\,|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$

Case $\overset{\mathsf{i}}{}\mathsf{c}^{\mathsf{l}} \,.\, \overset{\mathsf{k}}{}\mathsf{c}^{\mathsf{f}}$

$\alpha_{\mathfrak{m}}(\!|\,\overset{\mathsf{i}}{}\mathsf{c}_1^{\mathsf{l}} \,.\, \overset{\mathsf{k}}{}\mathsf{c}_2^{\mathsf{f}}\,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m})$

$$
\begin{array}{ll}
= & \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright} \\
\alpha_{\mathfrak{m}}(\!|\, {}^{\underline{k}}\mathsf{c}_2{}^{\underline{f}} \,|\!)^{\curvearrowright}(\!|\, {}^{\underline{i}}\mathsf{c}_1{}^{\underline{l}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) & \\
\sqsubseteq & \| \text{ extensivity of } \gamma_{\mathfrak{m}}\alpha_{\mathfrak{m}} \\
\alpha_{\mathfrak{m}}(\!|\, {}^{\underline{k}}\mathsf{c}_2{}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}\alpha_{\mathfrak{m}}(\!|\, {}^{\underline{i}}\mathsf{c}_1{}^{\underline{l}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) & \\
\sqsubseteq & \| \text{ inductive hypothesis} \\
(\!|\, {}^{\underline{k}}\mathsf{c}_2{}^{\underline{f}} \,|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\, {}^{\underline{i}}\mathsf{c}_1{}^{\underline{l}} \,|\!)^{\curvearrowright}_{\rho}\mathfrak{m} & \\
= & \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright}_{\rho} \\
(\!|\, {}^{\underline{i}}\mathsf{c}_1{}^{\underline{l}} \,.\, {}^{\underline{k}}\mathsf{c}_2{}^{\underline{f}} \,|\!)^{\curvearrowright}_{\rho}\mathfrak{m} &
\end{array}
$$

Case ${}^{\underline{i}}\mathsf{if\ b\ then\ \{\ P_1\ \}\ else\ \{\ P_2\ \}}^{\underline{f}}$ (♣)

In order to prove this case we prove the equivalent formulation:

$$\forall \mathsf{x} \in \mathsf{Var}\,.\quad (\alpha_{\mathfrak{m}}(\!|\, {}^{\underline{i}}\mathsf{if\ b\ then\ \{\ P_1\ \}\ else\ \{\ P_2\ \}}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq ((\!|\, {}^{\underline{i}}\mathsf{if\ b\ then\ \{\ P_1\ \}\ else\ \{\ P_2\ \}}^{\underline{f}} \,|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$$

Take an arbitrary $\mathsf{x} \in \mathsf{Var}$. Then we have:

$$
\begin{array}{ll}
(\alpha_{\mathfrak{m}}(\!|\, {}^{\underline{i}}\mathsf{if\ b\ then\ \{\ P_1\ \}\ else\ \{\ P_2\ \}}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) & \\
= & \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright} \\
(\alpha_{\mathfrak{m}}(\{[\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}X \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright}X \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}))(\mathsf{x}) & \\
= & \| \text{ definition of } \alpha_{\mathfrak{m}} \\
(\alpha_{h^{\rho}}(\{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}X \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright}X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}))(\mathsf{x}) &
\end{array}
$$

The proof continues by cases, recalling that $(\!|\, {}^{\underline{i}}\mathsf{if\ b\ then\ \{\ P_1\ \}\ else\ \{\ P_2\ \}}^{\underline{f}} \,|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$ is:

$$[\![\, \mathsf{x} \notin \mathsf{vars}^{:=}(\mathsf{P}_1) \cup \mathsf{vars}^{:=}(\mathsf{P}_2) \vee \mathfrak{n}(\mathsf{x}) \trianglelefteq \mathcal{A}^{\rho} \vee \mathsf{vars}^{\top}_{\mathfrak{m}}(\mathsf{b}) = \varnothing \;?\; \mathfrak{n}(\mathsf{x}) :\!: \top \,]\!]$$

where $\mathfrak{n} \triangleq (\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\rho}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m} \sqcup (\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\rho}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$ and $\trianglelefteq$ is the strict version of $\trianglelefteq$.

Case $\mathsf{x} \notin \mathsf{vars}^{:=}(\mathsf{P}_1) \cup \mathsf{vars}^{:=}(\mathsf{P}_2)$

We have that $(\alpha_{\mathfrak{m}}(\!|\, {}^{\underline{i}}\mathsf{if\ b\ then\ \{\ P_1\ \}\ else\ \{\ P_2\ \}}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x})$ is equal to $\mathfrak{m}(\mathsf{x})$, since x has not been modified ($\mathsf{vars}^{:=}$ is an over-approximation). For the same reason, we have that $(\alpha_{\mathfrak{m}}(\!|\mathsf{P}_1|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \mathfrak{m}(\mathsf{x})$ and $(\alpha_{\mathfrak{m}}(\!|\mathsf{P}_2|\!)^{\curvearrowright}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \mathfrak{m}(\mathsf{x})$. Then, by soundness of $(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}$ and by inductive hypothesis on $(\!|\mathsf{P}_1|\!)^{\curvearrowright}$ (analogous for $\neg\mathsf{b}$ and P_2), we have that $\alpha_{\mathfrak{m}}(\!|\mathsf{P}_1|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \sqsubseteq (\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\rho}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$ and $\alpha_{\mathfrak{m}}(\!|\mathsf{P}_2|\!)^{\curvearrowright}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \sqsubseteq (\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\rho}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m}$. So, $(\alpha_{\mathfrak{m}}(\!|\mathsf{P}_1|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq ((\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\rho}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$ and, similarly, $(\alpha_{\mathfrak{m}}(\!|\mathsf{P}_2|\!)^{\curvearrowright}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq ((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\rho}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$. This implies $\mathfrak{m}(\mathsf{x}) \trianglelefteq \mathfrak{n}(\mathsf{x}) = ((\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\rho}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) \veebar ((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\rho}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$, as we wanted.

Case $\mathfrak{n}(\mathsf{x}) = ((\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\rho}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) \veebar ((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\rho}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) \trianglelefteq \mathcal{A}^{\rho}$

Since b and $\neg\mathsf{b}$ cannot be false at the same time, we can have $((\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\rho}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x}) = \bot = ((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\rho}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\rho}\mathfrak{m})(\mathsf{x})$ if and only if $\mathfrak{m} = \mathfrak{m}_{\bot}$. So $\alpha_{\mathfrak{m}}(\!|\, {}^{\underline{i}}\mathsf{if\ b\ then\ \{\ P_1\ \}\ else\ \{\ P_2\ \}}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) =$

$\mathfrak{m}_\perp$ and $\mathfrak{n} = \mathfrak{m}_\perp$, hence $(\alpha_\mathfrak{m}(\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\underline{f}} \,|\!)^{\curvearrowright}_\rho \gamma_\mathfrak{m}(\mathfrak{m}))(\mathsf{x}) = \perp \trianglelefteq \perp = \mathfrak{n}(\mathsf{x})$ trivially holds. Otherwise, we have that $(\!|\mathsf{P}_1|\!)^{\curvearrowright}_\rho(\!|\mathsf{b}|\!)^{\curvearrowright}_\rho\mathfrak{m})(\mathsf{x}) = \bar{\mathfrak{a}}$ and $(\!|\mathsf{P}_2|\!)^{\curvearrowright}_\rho(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_\rho\mathfrak{m})(\mathsf{x}) \trianglelefteq \bar{\mathfrak{a}}$ (or the symmetric case). By soundness of $(\!|\mathsf{b}|\!)^{\curvearrowright}_\rho$ and inductive hypothesis on $(\!|\mathsf{P}_1|\!)^{\curvearrowright}_\rho$ we have that $\alpha_\mathfrak{m}(\!|\mathsf{P}_1|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_\mathfrak{m}(\mathfrak{m}) \sqsubseteq (\!|\mathsf{P}_1|\!)^{\curvearrowright}_\rho(\!|\mathsf{b}|\!)^{\curvearrowright}_\rho\mathfrak{m}$. Since $\{[\![\mathsf{P}_1]\!]^{\curvearrowright}X \mid X \in (\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_\mathfrak{m}(\mathfrak{m})\} \subseteq (\!|\mathsf{P}_1|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_\mathfrak{m}(\mathfrak{m})$ and, by definition, $(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_\mathfrak{m}(\mathfrak{m}) = \{[\![\mathsf{b}]\!]^{\curvearrowright}X \mid X \in \gamma_\mathfrak{m}(\mathfrak{m})\} \setminus \{\varnothing\}$, we have that $\{[\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}X \mid X \in \gamma_\mathfrak{m}(\mathfrak{m})\} \setminus \{\varnothing\} \subseteq (\!|\mathsf{P}|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_\mathfrak{m}(\mathfrak{m})$. By monotonicity of $\alpha_\mathfrak{m}$, we have that $\alpha(\{[\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}X \mid X \in \gamma_\mathfrak{m}(\mathfrak{m})\} \setminus \{\varnothing\}) \sqsubseteq (\!|\mathsf{P}_1|\!)^{\curvearrowright}_\rho(\!|\mathsf{b}|\!)^{\curvearrowright}_\rho\mathfrak{m}$. Analogous for $(\!|\mathsf{P}_2|\!)^{\curvearrowright}_\rho(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_\rho$. From $((\!|\mathsf{P}_1|\!)^{\curvearrowright}_\rho(\!|\mathsf{b}|\!)^{\curvearrowright}_\rho\mathfrak{m})(\mathsf{x}) = \bar{\mathfrak{a}}$ we can conclude that $\forall X \in \gamma_\mathfrak{m}(\mathfrak{m})$ we have $\{\mathfrak{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}X\} \subseteq \mathfrak{a}$. Similarly, $((\!|\mathsf{P}_2|\!)^{\curvearrowright}_\rho(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_\rho\mathfrak{m})(\mathsf{x}) = \perp$ implies that $\forall X \in \gamma_\mathfrak{m}(\mathfrak{m})$ we have $\{\mathfrak{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright}X\} = \varnothing$ and $((\!|\mathsf{P}_2|\!)^{\curvearrowright}_\rho(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_\rho\mathfrak{m})(\mathsf{x}) = \bar{\mathfrak{a}}$ implies that $\forall X \in \gamma_\mathfrak{m}(\mathfrak{m})$ we have $\{\mathfrak{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright}X\} \subseteq \mathfrak{a}$. Then, we have that $\{\{\mathfrak{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright}X\} \mid X \in \gamma_\mathfrak{m}(\mathfrak{m})\}$ is contained in $\mathfrak{a}$, in both cases. Thus, $\alpha_{h^\rho}(\{\{\mathfrak{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright}X\} \mid X \in \gamma_\mathfrak{m}(\mathfrak{m})\}) = \bar{\mathfrak{a}}$, which is approximated by $\mathcal{A}^\rho$, as required.

Case $\mathsf{vars}^\top_\mathfrak{m}(\mathsf{b}) = \varnothing$

We have that $\mathfrak{n}(\mathsf{x})$ is equal to $\mathcal{A}^\rho$ or $\top$, otherwise we fall into the previous cases. If $\mathfrak{n}(\mathsf{x}) = \top$, which is the maximum, then $(\alpha_\mathfrak{m}(\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\underline{f}} \,|\!)^{\curvearrowright}_\rho \gamma_\mathfrak{m}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq \top$ trivially holds. If $\mathfrak{n}(\mathsf{x}) = \mathcal{A}^\rho$, it could be that $((\!|\mathsf{P}_1|\!)^{\curvearrowright}_\rho(\!|\mathsf{b}|\!)^{\curvearrowright}_\rho\mathfrak{m})(\mathsf{x}) = \bar{\mathfrak{a}}$ and $((\!|\mathsf{P}_2|\!)^{\curvearrowright}_\rho(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_\rho\mathfrak{m})(\mathsf{x}) \in \{\bar{\mathfrak{b}}, \mathcal{A}^\rho\}$, with $\mathfrak{a} \neq \mathfrak{b}$ (or the symmetric case). The only way to falsify the proof is that $(\alpha_\mathfrak{m}(\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\underline{f}} \,|\!)^{\curvearrowright}_\rho \gamma_\mathfrak{m}(\mathfrak{m}))(\mathsf{x}) = \top$. This means that there exists $X \in \gamma_\mathfrak{m}(\mathfrak{m})$ such that $\{\mathfrak{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}X \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright}X\}$ is contained in $\mathfrak{a} \cup \mathfrak{b}$ (but not in $\mathfrak{a}$ or $\mathfrak{b}$). Since $\mathsf{vars}^\top_\mathfrak{m}(\mathsf{b}) = \varnothing$, this happens if and only if $\mathfrak{m}(\mathsf{x})$ is already equal to $\top$ and, hence, $\mathsf{x} \notin \mathsf{vars}(\mathsf{b})$. All these facts imply that $\mathfrak{n}(\mathsf{x})$ cannot be equal to $\mathcal{A}^\rho$ when $(\alpha_\mathfrak{m}(\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\underline{f}} \,|\!)^{\curvearrowright}_\rho \gamma_\mathfrak{m}(\mathfrak{m}))(\mathsf{x}) = \top$.

Case "otherwise"

Given that $(\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\underline{f}} \,|\!)^{\curvearrowright}_\rho\mathfrak{m})(\mathsf{x}) = \top$, which is the maximum, we have that $(\alpha_\mathfrak{m}(\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\underline{f}} \,|\!)^{\curvearrowright}_\rho \gamma_\mathfrak{m}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq \top$ trivially holds.

Since the variable x has been chosen arbitrarily, the proof holds for every variable. This terminates the case.

Case ${}^{\underline{i}}\mathsf{while}\ {}^{\underline{b}}\mathsf{k}\ \{\,\mathsf{P}\,\}^{\underline{f}}$

First, we can note that $\mathsf{H}^{\curvearrowright} = \lambda\mathcal{T}\,.\,\mathcal{X} \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}T \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}T \mid T \in \mathcal{T}\}$ coincides with the function $\lambda\mathcal{T}\,.\,\mathcal{X} \cup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}\mathcal{T}$, with $\underline{h}$ fresh label (it is indeed not necessary, since the post-conditions hypersemantics does not take into account labels). Now we need to derive a correct approximation of this latter:

$$
\begin{aligned}
&\lambda\mathfrak{n}\,.\,\alpha_\mathfrak{m}(\mathcal{X} \cup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_\mathfrak{m}(\mathfrak{n})) \\
&= \quad \| \text{ additivity of } \alpha_\mathfrak{m} \\
&= \lambda\mathfrak{n}\,.\,\alpha_\mathfrak{m}(\mathcal{X}) \sqcup \alpha_\mathfrak{m}((\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_\mathfrak{m}(\mathfrak{n})) \\
&\sqsubseteq \quad \| \text{ soundness of case } \clubsuit
\end{aligned}
$$

$\lambda \mathfrak{n} \,.\, \alpha_{\mathfrak{m}}(\mathcal{X}) \sqcup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\, \mathsf{P} \,\}\ \mathsf{else}\ \{\, {}^{\underline{h}}\mathsf{skip}^{\underline{h}} \,\}^{\underline{f}} \,|\!)^{\curvearrowright}_{\rho} \mathfrak{n}$

The function $\lambda \mathfrak{n} \,.\, \alpha_{\mathfrak{m}}(\mathcal{X}) \sqcup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\, \mathsf{P} \,\}\ \mathsf{else}\ \{\, {}^{\underline{h}}\mathsf{skip}^{\underline{h}} \,\}^{\underline{f}} \,|\!)^{\curvearrowright}_{\rho} \mathfrak{n}$ is sound, since it approximates the bca of $\mathsf{H}^{\curvearrowright}$. The bca is correct even at fixpoint, namely we have:

$$\alpha_{\mathfrak{m}}(\mathsf{lfp}^{\subseteq}_{\varnothing}\, \mathsf{H}^{\curvearrowright}) \sqsubseteq \mathsf{lfp}^{\sqsubseteq}_{\mathfrak{m}_{\perp}}\, \lambda \mathfrak{n} \,.\, \alpha_{\mathfrak{m}}(\mathcal{X}) \sqcup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\, \mathsf{P} \,\}\ \mathsf{else}\ \{\, {}^{\underline{h}}\mathsf{skip}^{\underline{h}} \,\}^{\underline{f}} \,|\!)^{\curvearrowright}_{\rho} \mathfrak{n} \tag{A.2}$$

Now we can continue the proof as follows.

$\alpha_{\mathfrak{m}}(\!|\, {}^{\underline{i}}\mathsf{while}\ {}^{\underline{k}}\mathsf{b}\ \{\, \mathsf{P} \,\}^{\underline{f}} \,|\!)^{\curvearrowright} \gamma_{\mathfrak{m}}(\mathfrak{m})$

$=$ $\quad \|$ definition of $(\!|\cdot|\!)^{\curvearrowright}$

$\alpha_{\mathfrak{m}}(\!|\neg \mathsf{b}|\!)^{\curvearrowright}(\mathsf{lfp}^{\subseteq}_{\varnothing}\, \lambda \mathcal{T} \,.\, \gamma_{\mathfrak{m}}(\mathfrak{m}) \cup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\, \mathsf{P} \,\}\ \mathsf{else}\ \{\, {}^{\underline{h}}\mathsf{skip}^{\underline{h}} \,\}^{\underline{f}} \,|\!)^{\curvearrowright} \mathcal{T})$

$\sqsubseteq$ $\quad \|$ soundness of $(\!|\mathsf{b}|\!)^{\curvearrowright}_{\rho}$

$(\!|\neg \mathsf{b}|\!)^{\curvearrowright}_{\rho} \alpha_{\mathfrak{m}}(\mathsf{lfp}^{\subseteq}_{\varnothing}\, \lambda \mathcal{T} \,.\, \gamma_{\mathfrak{m}}(\mathfrak{m}) \cup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\, \mathsf{P} \,\}\ \mathsf{else}\ \{\, {}^{\underline{h}}\mathsf{skip}^{\underline{h}} \,\}^{\underline{f}} \,|\!)^{\curvearrowright} \mathcal{T})$

$\sqsubseteq$ $\quad \|$ fixpoint approximation (equation A.2)

$(\!|\neg \mathsf{b}|\!)^{\curvearrowright}_{\rho}(\mathsf{lfp}^{\sqsubseteq}_{\mathfrak{m}_{\perp}}\, \lambda \mathfrak{n} \,.\, \alpha_{\mathfrak{m}} \gamma_{\mathfrak{m}}(\mathfrak{m}) \sqcup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\, \mathsf{P} \,\}\ \mathsf{else}\ \{\, {}^{\underline{h}}\mathsf{skip}^{\underline{h}} \,\}^{\underline{f}} \,|\!)^{\curvearrowright}_{\rho} \mathfrak{n})$

$=$ $\quad \|$ reductivity of $\alpha_{\mathfrak{m}} \gamma_{\mathfrak{m}}$

$(\!|\neg \mathsf{b}|\!)^{\curvearrowright}_{\rho}(\mathsf{lfp}^{\sqsubseteq}_{\mathfrak{m}_{\perp}}\, \lambda \mathfrak{n} \,.\, \mathfrak{m} \sqcup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\, \mathsf{P} \,\}\ \mathsf{else}\ \{\, {}^{\underline{h}}\mathsf{skip}^{\underline{h}} \,\}^{\underline{f}} \,|\!)^{\curvearrowright}_{\rho} \mathfrak{n})$

$=$ $\quad \|$ definition of $(\!|\cdot|\!)^{\curvearrowright}_{\rho}$

$(\!|\, {}^{\underline{i}}\mathsf{while}\ {}^{\underline{k}}\mathsf{b}\ \{\, \mathsf{P} \,\}^{\underline{f}} \,|\!)^{\curvearrowright}_{\rho} \mathfrak{m}$

□

Subsection 8.2.2

Lemma 9. *The abstract arithmetic operations* $\oplus^{\mathtt{Cprp}} \in \mathtt{C}^{\mathtt{Cprp}} \times \mathtt{C}^{\mathtt{Cprp}} \to \mathtt{C}^{\mathtt{Cprp}}$*, with* $\oplus \in \{+, -, *\}$*, are sound:*

$$\{\{n \oplus m \mid n \in X \wedge m \in Y\} \mid X \in \gamma_{h^{\mathtt{Cprp}}}(c_1) \wedge Y \in \gamma_{h^{\mathtt{Cprp}}}(c_2)\} \subseteq \gamma_{h^{\mathtt{Cprp}}}(c_1 \oplus^{\mathtt{C}} c_2)$$

Proof. The proof is by cases.

Case $c_1 = \bot, c_2 \in \mathtt{Cprp}$

$\gamma_{h^{\mathtt{Cprp}}}(\bot) = \varnothing$ and $\bot \oplus^{\mathtt{Cprp}} c_2 = \bot$ hence:
$\{\{n \oplus m \mid n \in X \wedge m \in Y\} \mid X \in \varnothing \wedge Y \in \gamma_{h^{\mathtt{Cprp}}}(c_2)\} = \varnothing \subseteq \varnothing = \gamma_{h^{\mathtt{Cprp}}}(\bot \oplus^{\mathtt{C}} c_2)$

Case $c_1 = \top, c_2 \in \mathtt{Cprp} \setminus \{\bot\}$

$\gamma_{h^{\mathtt{Cprp}}}(\top) = \mathtt{Cprp}(\wp(\mathbb{Z}))$ and $\top \oplus^{\mathtt{Cprp}} c_2 = \top$ hence:
$\{\{n \oplus m \mid n \in X \wedge m \in Y\} \mid X \in \mathtt{Cprp}(\wp(\mathbb{Z})) \wedge Y \in \gamma_{h^{\mathtt{Cprp}}}(c_2)\} = \mathtt{Cprp}(\wp(\mathbb{Z}))$
$\subseteq \mathtt{Cprp}(\wp(\mathbb{Z})) = \gamma_{h^{\mathtt{Cprp}}}(\bot \oplus^{\mathtt{C}} c_2)$

Case $c_1 = \bar{n}, c_2 = \bar{m}$

$\gamma_{h^{\mathsf{cprp}}}(\bar{n}) = \{\{n\}\}$ and $\bar{n} \oplus^{\mathsf{cprp}} \bar{m} = n \bar{\oplus} m$ hence:
$\{\{n \oplus m \mid n \in X \wedge m \in Y\} \mid X \in \{\{n\}\} \wedge Y \in \{\{m\}\}\} = \{\{n \oplus m\}\}$
$\subseteq \{\{n \oplus m\}\} = \gamma_{h^{\mathsf{cprp}}}(\bar{n} \oplus^{\mathsf{c}} \bar{m})$

Case $c_1 = \bar{n}, c_2 = \kappa$

$\gamma_{h^{\mathsf{cprp}}}(\bar{n}) = \{\{n\}\}$, $\gamma_{h^{\mathsf{cprp}}}\kappa = \{\{n\} \mid n \in \mathbb{Z}\}$ and $\bar{n} \oplus^{\mathsf{cprp}} \kappa = \kappa$ hence:
$\{\{n \oplus m \mid n \in X \wedge m \in Y\} \mid X \in \{\{n\}\} \wedge Y \in \{\{n\} \mid n \in \mathbb{Z}\}\} \subseteq \{\{n\} \mid n \in \mathbb{Z}\}$
$\subseteq \{\{n\} \mid n \in \mathbb{Z}\} = \gamma_{h^{\mathsf{cprp}}}(\bar{n} \oplus^{\mathsf{c}} \kappa)$

Case $c_1 = \kappa = c_2$

$\gamma_{h^{\mathsf{cprp}}}(\kappa) = \{\{n\} \mid n \in \mathbb{Z}\}$ and $\kappa \oplus^{\mathsf{cprp}} \kappa = \kappa$ hence:
$\{\{n \oplus m \mid n \in X \wedge m \in Y\} \mid X \in \{\{n\} \mid n \in \mathbb{Z}\} \wedge Y \in \{\{n\} \mid n \in \mathbb{Z}\}\} = \{\{n\} \mid n \in \mathbb{Z}\}$
$\subseteq \{\{n\} \mid n \in \mathbb{Z}\} = \gamma_{h^{\mathsf{cprp}}}(\kappa \oplus^{\mathsf{c}} \kappa)$

By symmetry, we cover all possible cases. □

Lemma 10. *The abstract hypersemantics for arithmetic expression* $(\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{c}}$ *is sound:*

$$\{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \subseteq \gamma_{h^{\mathsf{cprp}}}(\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{c}}\mathfrak{m}$$

Proof. The proof is for structural induction on a.

Case n

$\{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{n}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}$
$=$ ‖ definition of $\Downarrow^{\mathbb{Z}}$
$\{\{n\}\}$
$\subseteq$ ‖ extensivity of $\gamma_{h^{\mathsf{cprp}}}\alpha_{h^{\mathsf{cprp}}}$
$\gamma_{h^{\mathsf{cprp}}}\alpha_{h^{\mathsf{cprp}}}(\{\{n\}\})$
$=$ ‖ definition of $\alpha_{h^{\mathsf{cprp}}}$
$\gamma_{h^{\mathsf{cprp}}}(\bar{n})$
$=$ ‖ definition of $(\!|\cdot|\!)^{\curvearrowright}_{\mathsf{c}}$
$\gamma_{h^{\mathsf{cprp}}}(\!|\mathsf{n}|\!)^{\curvearrowright}_{\mathsf{c}}\mathfrak{m}$

Case x

$\{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{x}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}$
$=$ ‖ definition of $\Downarrow^{\mathbb{Z}}$

$$
\begin{array}{ll}
& \{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
\subseteq & \quad \| \text{ extensivity of } \gamma_{h^{\mathtt{cprp}}} \alpha_{h^{\mathtt{cprp}}} \\
& \gamma_{h^{\mathtt{cprp}}} \alpha_{h^{\mathtt{cprp}}} (\{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}) \\
= & \quad \| \text{ definition of } \gamma_{\mathfrak{m}} \\
& \gamma_{h^{\mathtt{cprp}}} \alpha_{h^{\mathtt{cprp}}} (\gamma_{h^{\mathtt{cprp}}} (\mathfrak{m}(\mathsf{x}))) \\
\subseteq & \quad \| \text{ reductivity of } \alpha_{h^{\mathtt{cprp}}} \gamma_{h^{\mathtt{cprp}}} \\
& \gamma_{h^{\mathtt{cprp}}} (\mathfrak{m}(\mathsf{x})) \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright}_{\mathsf{c}} \\
& \gamma_{h^{\mathtt{cprp}}} (\!|\mathsf{x}|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m}
\end{array}
$$

Case (a)

$$
\begin{array}{ll}
& \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle (\mathsf{a}), \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
= & \quad \| \text{ definition of } \Downarrow^{\mathbb{Z}} \\
& \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
\subseteq & \quad \| \text{ inductive hypothesis} \\
& \gamma_{h^{\mathtt{cprp}}} (\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m}
\end{array}
$$

Case $\mathsf{a}_1 \oplus \mathsf{a}_2$*, with* $\oplus \in \{+, -, *\}$

$$
\begin{array}{ll}
& \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_1 \oplus \mathsf{a}_2, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
= & \quad \| \text{ definition of } \Downarrow^{\mathbb{Z}} \\
& \{\{n_1 \oplus n_2 \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_1, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n_2\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
\subseteq & \quad \| \text{ set theory} \\
& \{\{n_1 \oplus n_2 \mid \exists \mathsf{m}, \mathsf{m}' \in X \,.\, \langle \mathsf{a}_1, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m}' \rangle \Downarrow^{\mathbb{Z}} n_2\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
= & \quad \| \text{ set theory} \\
& \{\{n_1 \oplus n_2 \mid n_1 \in \{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_1, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n\} \wedge n_2 \in \{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_2, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n\}\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \\
\subseteq & \quad \| \text{ set theory} \\
& \left\{ \{n_1 \oplus n_2 \mid n_1 \in X \wedge n_2 \in Y\} \;\middle|\; \begin{array}{l} X \in \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_1, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \wedge \\ Y \in \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}_2, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \end{array} \right\} \\
\subseteq & \quad \| \text{ inductive hypothesis on } \mathsf{a}_1, \mathsf{a}_2 \\
& \{\{n_1 \oplus n_2 \mid n_1 \in X \wedge n_2 \in Y\} \mid X \in \gamma_{h^{\mathtt{cprp}}} (\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m} \wedge Y \in \gamma_{h^{\mathtt{cprp}}} (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m}\} \\
\subseteq & \quad \| \text{ soundness of } \oplus^{\mathtt{cprp}} \\
& \gamma_{h^{\mathtt{cprp}}} ((\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m} \oplus^{\mathtt{cprp}} (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m}) \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright}_{\mathsf{c}} \\
& \gamma_{h^{\mathtt{cprp}}} (\!|\mathsf{a}_1 \oplus \mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m}
\end{array}
$$

□

Lemma 11. *The abstract logical operations* $\bowtie^{\mathtt{c}} \in \mathtt{C}^{\mathtt{Cprp}} \times \mathtt{C}^{\mathtt{Cprp}} \to \mathtt{C}^{\mathtt{Cprp}} \times \mathtt{C}^{\mathtt{Cprp}}$*, with* $\bowtie \in \{=, \neq, <, \leq\}$*, are sound:*

$$\begin{array}{l}\textit{let } \mathcal{X} = \{\{\langle n, m\rangle \mid n \in X \wedge m \in Y \wedge n \bowtie m\} \mid X \in \gamma_{h^{\mathtt{Cprp}}}(c_1) \wedge Y \in \gamma_{h^{\mathtt{Cprp}}}(c_2)\} \textit{ in} \\ \langle\{\{n \mid \langle n, m\rangle \in X\} \mid X \in \mathcal{X}\}, \{\{m \mid \langle n, m\rangle \in X\} \mid X \in \mathcal{X}\}\rangle \subseteq^2 \gamma^2_{h^{\mathtt{Cprp}}}(c_1 \bowtie^{\mathtt{c}} c_2)\end{array}$$

Proof. We recall that, given a pair $\langle X, Y\rangle$, we denote with $\langle X, Y\rangle_{\vdash} = X$ its projection on the first element of the pair and with $\langle X, Y\rangle_{\dashv} = Y$ its projection on the second element. The proof is by cases.

Case $c_1 = \bot, c_2 \in \mathtt{Cprp}$

$\gamma_{h^{\mathtt{Cprp}}}(\bot) = \varnothing$ and $\bot \bowtie^{\mathtt{c}} c_2 = \langle \bot, \bot\rangle$ hence:
$\mathcal{X} = \{\{\langle n, m\rangle \mid n \in X \wedge m \in Y \wedge n \bowtie m\} \mid X \in \varnothing \wedge Y \in \gamma_{h^{\mathtt{Cprp}}}(c_2)\} = \varnothing$ and
$\{\{n \mid \langle n, m\rangle \in X\} \mid X \in \varnothing\} = \varnothing \subseteq \varnothing = (\gamma^2_{h^{\mathtt{Cprp}}}(\bot \bowtie^{\mathtt{c}} c_2))_{\vdash}$
$\{\{m \mid \langle n, m\rangle \in X\} \mid X \in \varnothing\} = \varnothing \subseteq \varnothing = (\gamma^2_{h^{\mathtt{Cprp}}}(\bot \bowtie^{\mathtt{c}} c_2))_{\dashv}$

Case $c_1 = \bar{n} = c_2$ *and* $\bowtie \in \{=, \leq\}$

$\gamma_{h^{\mathtt{Cprp}}}(\bar{n}) = \{\{n\}\}$ and $\bar{n} \bowtie^{\mathtt{c}} \bar{n} = \langle \bar{n}, \bar{n}\rangle$ hence:
$\mathcal{X} = \{\{\langle n, m\rangle \mid n \in X \wedge m \in Y \wedge n \bowtie m\} \mid X \in \{\{n\}\} \wedge Y \in \{\{n\}\}\} = \{\{\langle n, n\rangle\}\}$ and
$\{\{n \mid \langle n, m\rangle \in X\} \mid X \in \{\{\langle n, n\rangle\}\}\} = \{\{n\}\} \subseteq \{\{n\}\} = (\gamma^2_{h^{\mathtt{Cprp}}}(\bar{n} \bowtie^{\mathtt{c}} \bar{n}))_{\vdash}$
$\{\{m \mid \langle n, m\rangle \in X\} \mid X \in \{\{\langle n, n\rangle\}\}\} = \{\{n\}\} \subseteq \{\{n\}\} = (\gamma^2_{h^{\mathtt{Cprp}}}(\bar{n} \bowtie^{\mathtt{c}} \bar{n}))_{\dashv}$

Case $c_1 = \bar{n}, c_2 = \bar{m}$*,* $n < m$ *and* $\bowtie \in \{<, \leq, \neq\}$

$\gamma_{h^{\mathtt{Cprp}}}(\bar{n}) = \{\{n\}\}$ and $\bar{n} \bowtie^{\mathtt{c}} \bar{m} = \langle \bar{n}, \bar{m}\rangle$ hence:
$\mathcal{X} = \{\{\langle n, m\rangle \mid n \in X \wedge m \in Y \wedge n \bowtie m\} \mid X \in \{\{n\}\} \wedge Y \in \{\{m\}\}\} = \{\{\langle n, m\rangle\}\}$ and
$\{\{n \mid \langle n, m\rangle \in X\} \mid X \in \{\{\langle n, m\rangle\}\}\} = \{\{n\}\} \subseteq \{\{n\}\} = (\gamma^2_{h^{\mathtt{Cprp}}}(\bar{n} \bowtie^{\mathtt{c}} \bar{m}))_{\vdash}$
$\{\{m \mid \langle n, m\rangle \in X\} \mid X \in \{\{\langle n, m\rangle\}\}\} = \{\{m\}\} \subseteq \{\{m\}\} = (\gamma^2_{h^{\mathtt{Cprp}}}(\bar{n} \bowtie^{\mathtt{c}} \bar{m}))_{\dashv}$

Case $c_1 = \bar{n}, c_2 = \bar{m}$*,* $n > m$ *and* $\bowtie \in \{\neq\}$

$\gamma_{h^{\mathtt{Cprp}}}(\bar{n}) = \{\{n\}\}$ and $\bar{n} \bowtie^{\mathtt{c}} \bar{m} = \langle \bar{n}, \bar{m}\rangle$ hence:
$\mathcal{X} = \{\{\langle n, m\rangle \mid n \in X \wedge m \in Y \wedge n \bowtie m\} \mid X \in \{\{n\}\} \wedge Y \in \{\{m\}\}\} = \{\{\langle n, m\rangle\}\}$ and
$\{\{n \mid \langle n, m\rangle \in X\} \mid X \in \{\{\langle n, m\rangle\}\}\} = \{\{n\}\} \subseteq \{\{n\}\} = (\gamma^2_{h^{\mathtt{Cprp}}}(\bar{n} \bowtie^{\mathtt{c}} \bar{m}))_{\vdash}$
$\{\{m \mid \langle n, m\rangle \in X\} \mid X \in \{\{\langle n, m\rangle\}\}\} = \{\{m\}\} \subseteq \{\{m\}\} = (\gamma^2_{h^{\mathtt{Cprp}}}(\bar{n} \bowtie^{\mathtt{c}} \bar{m}))_{\dashv}$

For all other cases we have $c_1 \oplus^{\mathtt{Cprp}} c_2 = \langle \top, \top\rangle$ hence the inclusions trivially hold. □

Lemma 12. *The abstract hypersemantics for boolean expression* $(\!|\mathsf{b}|\!)^{\mathtt{C}}_{\mathtt{C}}$ *is sound:*

$$(\!|\mathsf{b}|\!)^{\mathtt{C}}\gamma_{\mathtt{m}}(\mathtt{m}) = \{\{\mathsf{m} \in X \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathtt{m}}(\mathtt{m})\} \setminus \{\varnothing\} \subseteq \gamma_{\mathtt{m}}(\!|\mathsf{b}|\!)^{\mathtt{C}}_{\mathtt{C}}\mathtt{m}$$

Proof. The proof is for structural induction on b.

Case $\mathtt{tt}$

$$
\begin{array}{ll}
& \{\{\mathsf{m} \in X \mid \langle \mathtt{tt}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathfrak{m}}(\mathsf{m})\} \setminus \{\varnothing\} \\
= & \quad \| \text{ definition of } \Downarrow^{\mathbb{B}} \\
& \gamma_{\mathfrak{m}}(\mathfrak{m}) \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright}_{\mathsf{c}} \\
& \gamma_{\mathfrak{m}}(\!|\mathtt{tt}|\!)^{\curvearrowright}_{\mathsf{c}}\mathfrak{m}
\end{array}
$$

Case $\mathtt{ff}$

$$
\begin{array}{ll}
& \{\{\mathsf{m} \in X \mid \langle \mathtt{ff}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathfrak{m}}(\mathsf{m})\} \setminus \{\varnothing\} \\
= & \quad \| \text{ definition of } \Downarrow^{\mathbb{B}} \\
& \{\varnothing\} \setminus \{\varnothing\} \\
= & \quad \| \text{ set theory} \\
& \varnothing \\
= & \quad \| \text{ definition of } \gamma_{\mathfrak{m}} \\
& \gamma_{\mathfrak{m}}(\mathfrak{m}_{\perp}) \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright}_{\mathsf{c}} \\
& \gamma_{\mathfrak{m}}(\!|\mathtt{ff}|\!)^{\curvearrowright}_{\mathsf{c}}\mathfrak{m}
\end{array}
$$

Case (b)

$$
\begin{array}{ll}
& \{\{\mathsf{m} \in X \mid \langle (\mathsf{b}), \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathfrak{m}}(\mathsf{m})\} \setminus \{\varnothing\} \\
= & \quad \| \text{ definition of } \Downarrow^{\mathbb{B}} \\
& \{\{\mathsf{m} \in X \mid \langle \mathsf{b}, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \mid X \in \gamma_{\mathfrak{m}}(\mathsf{m})\} \setminus \{\varnothing\} \\
\subseteq & \quad \| \text{ inductive hypothesis} \\
& \gamma_{\mathfrak{m}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{c}}\mathfrak{m}
\end{array}
$$

Case $\mathsf{b}_1 \wedge \mathsf{b}_2$

In order to prove this case we prove the equivalent formulation:

$$\forall \mathsf{x} \in \mathsf{Var}\,.\, (\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq (\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{c}}\mathfrak{m})(\mathsf{x})$$

Take an arbitrary $\mathsf{x} \in \mathsf{Var}$. Then we have:

$$
\begin{array}{ll}
& (\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \\
= & \quad \| \text{ definition of } (\!|\mathsf{b}|\!)^{\curvearrowright} \\
& (\alpha_{\mathfrak{m}}(\{\{[\![\mathsf{b}_1]\!]^{\curvearrowright}X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright}X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \setminus \{\varnothing\}))(\mathsf{x})
\end{array}
$$

$$= \quad \| \text{ definition of } \alpha_{\mathfrak{m}}$$
$$(\alpha_{h^{\text{cprp}}}(\{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright} X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \setminus \{\varnothing\}))(\mathsf{x})$$

The proof continues by cases, recalling that $(\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = (\!(\mathsf{b}_1)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \mathbin{\overline{\wedge}} (\!(\mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$.

Case $(\alpha_{\mathfrak{m}}(\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \bot$

Since $\bot$ is the minimum, $\bot \trianglelefteq (\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$ trivially holds.

Case $(\alpha_{\mathfrak{m}}(\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \bar{n}$

$(\alpha_{\mathfrak{m}}(\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \bar{n}$ implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright} X\}$ is $\{n\}$. Which, in turn, implies that $\exists X_1, X_2 \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X_2\} \neq \varnothing$ and $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright} X_2\} \neq \varnothing$. This excludes the possibility to have $(\!(\mathsf{b}_1)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bot$ or $(\!(\mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bot$. In fact, by inductive hypothesis, $(\!(\mathsf{b}_1)\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}(\!(\mathsf{b}_1)\!)^{\curvearrowright}_{\mathfrak{m}}$ and $(\!(\mathsf{b}_2)\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}(\!(\mathsf{b}_2)\!)^{\curvearrowright}_{\mathfrak{m}}$. Thus, the only way to falsify the proof is that $(\!(\mathsf{b}_1)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{n}$ and $(\!(\mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{m}$, with $n \neq m$ (or the symmetric case). In fact, $\bar{n} \mathbin{\overline{\wedge}} \bar{m} = \bot$ which is unsound. So, suppose to be in that case. This means that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m}) \,.\, \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X\} = \{n\} \wedge \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X\} = \{m\}$. But this implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright} X\}$ is $\{n, m\}$, which is absurd, since we have supposed that it is equal to $\{n\}$. All this proves that $\bar{n} \trianglelefteq (\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$.

Case $(\alpha_{\mathfrak{m}}(\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \kappa$

$(\alpha_{\mathfrak{m}}(\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \kappa$ implies that $\exists X, Y \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright} X\} = \{n\}$ and $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} Y \cap [\![\mathsf{b}_2]\!]^{\curvearrowright} Y\} = \{m\}$, with $n \neq m$. As for the previous case, $(\!(\mathsf{b}_1)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \neq \bot$ or $(\!(\mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \neq \bot$. The only way to falsify the proof is that $(\!(\mathsf{b}_1)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{n}$ and $(\!(\mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \trianglelefteq \kappa$ (or the symmetric case). In fact, $\bar{n} \mathbin{\overline{\wedge}} \kappa = \bar{n}$, $\bar{n} \mathbin{\overline{\wedge}} \bar{n} = \bar{n}$ and $\bar{n} \mathbin{\overline{\wedge}} \bar{m} = \bot$ are unsound results. So, suppose $(\!(\mathsf{b}_1)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{n}$ and $(\!(\mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \kappa$. This means that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m}) \,.\, \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X\} = \{n\}$ and $\exists X_1, X_2 \in \gamma_{\mathfrak{m}}(\mathfrak{m}) \,.\, \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright} X_1\} = \{n\} \wedge \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright} X_2\} = \{m\}$. But this implies that $\exists X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright} X\}$ is $\{n, m\}$, which is absurd, since we have supposed that it is equal to κ. Now, suppose $(\!(\mathsf{b}_1)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{n}$ and $(\!(\mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{n}$. This means that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m}) \,.\, \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X\} = \{n\} = \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright} X\}$. But this implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright} X\}$ is $\{n\}$, which is absurd, since we have supposed that it is equal to κ. Finally, suppose $(\!(\mathsf{b}_1)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{n}$ and $(\!(\mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{m}$. This means that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m}) \,.\, \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X\} = \{n\} \wedge \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright} X\} = \{m\}$. But this implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright} X\}$ is $\{n, m\}$, which is absurd, since we have supposed that it is equal to κ. All this proves that $\kappa \trianglelefteq (\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$.

Case $(\alpha_{\mathfrak{m}}(\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \top$

$(\alpha_{\mathfrak{m}}(\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \top$ implies that $\exists X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cap [\![\mathsf{b}_2]\!]^{\curvearrowright} X\} \supseteq \{n, m\}$. This trivially implies that for the same X we have $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X\} \supseteq \{n, m\}$ and $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright} X\} \supseteq \{n, m\}$. This is sufficient to state that $(\!(\mathsf{b}_1)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \top$ and $(\!(\mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \top$. Hence $\top \trianglelefteq (\!(\mathsf{b}_1 \wedge \mathsf{b}_2)\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$.

Since the variable x has been chosen arbitrarily, the proof holds for every variable. This terminates the case.

Case $\mathsf{b}_1 \vee \mathsf{b}_2$

In order to prove this case we prove the equivalent formulation:

$$\forall \mathsf{x} \in \mathsf{Var}\,.\,(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq (\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$$

Take an arbitrary $\mathsf{x} \in \mathsf{Var}$. Then we have:

$$\begin{array}{ll}
& (\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \\
= & \quad \| \text{ definition of } (\!|\mathsf{b}|\!)^{\curvearrowright} \\
& (\alpha_{\mathfrak{m}}(\{\{[\![\mathsf{b}_1]\!]^{\curvearrowright}X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright}X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \setminus \{\varnothing\}))(\mathsf{x}) \\
= & \quad \| \text{ definition of } \alpha_{\mathfrak{m}} \\
& (\alpha_{h^{\mathsf{cprp}}}(\{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright}X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \setminus \{\varnothing\}))(\mathsf{x})
\end{array}$$

The proof continues by cases, recalling that:

$$(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \begin{cases} (\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \veebar (\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) & \text{if } \mathfrak{m}(\mathsf{x}) \neq \top \vee \mathsf{x} \notin \mathsf{vars}(\mathsf{b}_1) \cap \mathsf{vars}(\mathsf{b}_1) \\ \top & \text{otherwise} \end{cases}$$

Case $(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \bot$

Since $\bot$ is the minimum, $\bot \trianglelefteq (\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$ trivially holds.

Case $(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \bar{n}$

$(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \bar{n}$ implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright}X\}$ is $\{n\}$. Which, in turn, implies that $\exists X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X\} \neq \varnothing$ or $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright}X_2\} \neq \varnothing$, but not necessarily both. This excludes the possibility to have $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bot$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bot$. In fact, by inductive hypothesis, $(\!|\mathsf{b}_1|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\mathfrak{m}}$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\mathfrak{m}}$. This is not a problem, since if $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bot$ then $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$ must be necessarily equal to $\bar{n}$ (or the symmetric case). This guarantees that the result $\bot \veebar \bar{n} = \bar{h}$ is sound. There are not other outcome leading to $\bot$ hence we have that $\bar{n} \trianglelefteq (\!|\mathsf{b}_1 \wedge \mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$.

Case $(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \kappa$

$(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \kappa$ implies that $\exists X, Y \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright}X\} = \{n\}$ and $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}Y \cup [\![\mathsf{b}_2]\!]^{\curvearrowright}Y\} = \{m\}$, with $n \neq m$. As for the previous case, $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \neq \bot$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \neq \bot$ at the same time and if one of the two is $\bot$ then the other must be κ, so $\bot \veebar \kappa = \kappa$ is sound. The only way to falsify the proof is that $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{n} = (\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$. In fact, $\bar{n} \veebar \bar{n} = \bar{n}$ which is unsound. So, suppose to be in that case. This means that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\,.\,\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X\} = \{n\} = \{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_2]\!]^{\curvearrowright}X\}$. But this implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ the set $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright}X\}$ is $\{n\}$, which is absurd, since we have supposed that it is equal to κ. All this proves that $\kappa \trianglelefteq (\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$.

Case $(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \top$

$(\alpha_{\mathfrak{m}}(\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \top$ implies that $\exists X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright}X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright}X\} \supseteq \{n, m\}$. As for the previous cases, $(\!|\mathsf{b}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \neq \bot$ and $(\!|\mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \neq \bot$

at the same time and if one of the two is $\bot$ then the other must be $\top$, so $\bot \veebar \top = \top$ is sound. Then, it is easy to note that there exists $X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{b}_1]\!]^{\curvearrowright} X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright} X\} \supseteq \{n, m\}$ if and only if $\mathfrak{m}(\mathsf{x}) = \top$ (indeed, $[\![\mathsf{b}_1]\!]^{\curvearrowright} X \cup [\![\mathsf{b}_2]\!]^{\curvearrowright} X \subseteq X$). In this case, by definition, $((\!|\mathsf{b}_1 \vee \mathsf{b}_2|\!)^{\curvearrowright}_{\mathsf{C}} \mathfrak{m})(\mathsf{x}) = \top$, hence it is sound.

Since the variable x has been chosen arbitrarily, the proof holds for every variable. This terminates the case.

Case $\mathsf{a}_1 \bowtie \mathsf{a}_2$, *with* $\bowtie \in \{=, \neq, <, \leq\}$

The proof for this case is divided in three parts. First, we have to prove the following preliminary result. Given $\mathfrak{m} \in \mathsf{Mem}^{\mathsf{cprp}}$ we have that:

$$\alpha_{\mathfrak{m}}(\textstyle\bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})) \sqsubseteq \mathfrak{m} \tag{A.3}$$

If there exists a variable x such that $\mathfrak{m}(\mathsf{x}) = \bot$, then $\gamma_{\mathfrak{m}}(\mathfrak{m}) = \varnothing$ and so $\alpha_{\mathfrak{m}}(\bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})) = \mathfrak{m}_{\bot} \sqsubseteq \mathfrak{m}$. Otherwise, take an arbitrary variable x. If $\mathfrak{m}(\mathsf{x}) = \bar{n}$ then for every $X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ we have that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X\}\{n\}$. But this implies that for every $X' \subseteq X$ we have $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X'\} = \{n\}$ as well. Thus, $(\alpha_{\mathfrak{m}}(\bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})))(\mathsf{x}) = \bar{n}$. Now suppose $\mathfrak{m}(\mathsf{x}) = \kappa$. This means that there exist $X, Y \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X\} = \{n\}$ and $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in Y\} = \{m\}$, with $n \neq m$. Now we can reason as before: for every $X' \subseteq X$ and $Y' \subseteq Y$ we have $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in X'\}\{n\}$ and $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in Y'\} \subseteq \{m\}$. Thus, $(\alpha_{\mathfrak{m}}(\bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})))(\mathsf{x}) = \kappa$. Finally, in the case of $\mathfrak{m}(\mathsf{x}) = \top$, $\alpha_{\mathfrak{m}}(\bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})))(\mathsf{x}) \trianglelefteq \mathfrak{m}(\mathsf{x})$ holds. Since the variable x has been chosen arbitrarily, the proof holds for every variable.

In any case we have that $(\alpha_{\mathfrak{m}}(\bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})))(\mathsf{x}) \trianglelefteq \mathfrak{m}(\mathsf{x})$, as requested by A.3. Now we retrieve an superset of $(\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) = \left\{\{\mathsf{m} \in X \mid \langle \mathsf{a}_1 \bowtie \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \;\middle|\; X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\right\} \setminus \{\varnothing\}$. In the following, we let $\mathbb{D} = \bigcup_{Z \in \gamma_{\mathfrak{m}}(\mathfrak{m})}(\wp(Z) \setminus \{\varnothing\})$.

$$
\begin{array}{ll}
& \left\{\{\mathsf{m} \in X \mid \langle \mathsf{a}_1 \bowtie \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{B}} \mathbb{tt}\} \;\middle|\; X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\right\} \setminus \{\varnothing\} \\
= & \quad \| \text{ definition of } \Downarrow^{\mathbb{B}} \\
& \left\{\{\mathsf{m} \in X \mid \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \wedge n_1 \bowtie n_2\} \;\middle|\; X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\right\} \setminus \{\varnothing\} \\
= & \quad \| \text{ set theory and definition of } [\![\mathsf{a}]\!]^{\curvearrowright} \\
& \left\{\left\{\mathsf{m} \in X \;\middle|\; \exists n_1 \in [\![\mathsf{a}_1]\!]^{\curvearrowright} X \, \exists n_2 \in [\![\mathsf{a}_2]\!]^{\curvearrowright} X \,.\, \left(\begin{array}{l} \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \\ \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \wedge \\ n_1 \bowtie n_2 \end{array}\right)\right\} \;\middle|\; X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\right\} \setminus \{\varnothing\} \\
\subseteq & \quad \| \text{ set theory and definition of } \mathbb{D} \\
& \left\{X \in \mathbb{D} \;\middle|\; \forall \mathsf{m} \in X \, \exists n_1 \in [\![\mathsf{a}_1]\!]^{\curvearrowright} X \, \exists n_2 \in [\![\mathsf{a}_2]\!]^{\curvearrowright} X \,.\, \left(\begin{array}{l} \langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \\ \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2 \wedge \\ n_1 \bowtie n_2 \end{array}\right)\right\} \\
\subseteq & \quad \| \text{ set theory and definition of } (\!|\mathsf{a}|\!)^{\curvearrowright} \\
& \quad \text{let } \mathbb{P} = \left\{\left\{\langle n_1, n_2\rangle \;\middle|\; \begin{array}{l} n_1 \in N_1 \wedge \\ n_2 \in N_2 \wedge \\ n_1 \bowtie n_2 \end{array}\right\} \;\middle|\; N_1 \in (\!|\mathsf{a}_1|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \wedge N_2 \in (\!|\mathsf{a}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m})\right\} \text{ in} \\
& \left\{X \subseteq \mathbb{D} \;\middle|\; \exists P \in \mathbb{P} \forall \mathsf{m} \in X \, \exists \langle n_1, n_2\rangle \in P \,.\, (\langle \mathsf{a}_1, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n_2)\right\}
\end{array}
$$

$$
\begin{array}{ll}
\subseteq & \| \text{ soundness of } (\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{C}} \\
& \text{let } \mathbb{P} = \left\{ \left\{ \langle n_1, n_2 \rangle \;\middle|\; \begin{array}{l} n_1 \in N_1 \wedge \\ n_2 \in N_2 \wedge \\ n_1 \bowtie n_2 \end{array} \right\} \;\middle|\; N_1 \in \gamma_{h^{\mathsf{Cprp}}}((\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}) \wedge N_2 \in \gamma_{h^{\mathsf{Cprp}}}((\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}) \right\} \text{ in} \\
& \left\{ X \subseteq \mathbb{D} \;\middle|\; \exists P \in \mathbb{P}\, \forall \mathsf{m} \in X\, \exists \langle n_1, n_2 \rangle \in P \,.\, (\langle \mathsf{a}_1, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n_2) \right\} \\
\subseteq & \| \text{ soundness of } \bowtie^{\mathsf{Cprp}} \text{ (i.e. } \mathbb{P} \subseteq \{\{\langle n,m \rangle \mid n \in X \wedge m \in Y\} \mid \langle \mathcal{X}, \mathcal{Y} \rangle = \gamma^2_{h^{\mathsf{Cprp}}}((\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m} \bowtie^{\mathsf{Cprp}} (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}) \wedge X \in \mathcal{X} \wedge Y \in \mathcal{Y}\}) \\
& \text{let } \mathbb{P}' = \left\{ \left\{ \langle n, m \rangle \;\middle|\; n \in X \wedge m \in Y \right\} \;\middle|\; \begin{array}{l} \langle \mathcal{X}, \mathcal{Y} \rangle = \gamma^2_{h^{\mathsf{Cprp}}}((\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m} \bowtie^{\mathsf{Cprp}} (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}) \\ \wedge X \in \mathcal{X} \wedge Y \in \mathcal{Y} \end{array} \right\} \text{ in} \\
& \left\{ X \subseteq \mathbb{D} \;\middle|\; \exists P \in \mathbb{P}'\, \forall \mathsf{m} \in X\, \exists \langle n_1, n_2 \rangle \in P \,.\, (\langle \mathsf{a}_1, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n_1 \wedge \langle \mathsf{a}_2, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n_2) \right\} \\
\subseteq & \| \text{ set theory and definition of } [\![\mathsf{a}]\!]^{\curvearrowright} \\
& \text{let } \langle c_1, c_2 \rangle = (\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m} \bowtie^{\mathsf{Cprp}} (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m} \text{ in} \\
& \left\{ X \subseteq \mathbb{D} \;\middle|\; [\![\mathsf{a}_1]\!]^{\curvearrowright} X \in \gamma_{h^{\mathsf{Cprp}}}(c_1) \wedge [\![\mathsf{a}_2]\!]^{\curvearrowright} X \in \gamma_{h^{\mathsf{Cprp}}}(c_2) \right\} \\
\subseteq & \| \text{ set theory and definition of } (\!|\mathsf{a}|\!)^{\curvearrowright} \\
& \text{let } \langle c_1, c_2 \rangle = (\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m} \bowtie^{\mathsf{Cprp}} (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m} \text{ in} \\
& \bigcup \left\{ \mathcal{X} \subseteq \mathbb{D} \;\middle|\; (\!|\mathsf{a}_1|\!)^{\curvearrowright} \mathcal{X} \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_1) \wedge (\!|\mathsf{a}_2|\!)^{\curvearrowright} \mathcal{X} \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_2) \right\} \\
\subseteq & \| \text{ set theory} \\
& \bigcup \{ \mathcal{X} \cap \mathcal{Y} \mid \mathcal{X}, \mathcal{Y} \subseteq \mathbb{D} \wedge (\!|\mathsf{a}_1|\!)^{\curvearrowright} \mathcal{X} \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_1) \wedge (\!|\mathsf{a}_2|\!)^{\curvearrowright} \mathcal{Y} \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_2) \} \\
= & \| \text{ distributivity of } \cap \\
& \bigcup \{ \mathcal{X} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_1|\!)^{\curvearrowright} \mathcal{X} \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_1) \} \cap \bigcup \{ \mathcal{Y} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_2|\!)^{\curvearrowright} \mathcal{Y} \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_2) \}
\end{array}
$$

Hence $(\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \bigcup\{\mathcal{X} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_1|\!)^{\curvearrowright}\mathcal{X} \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_1)\} \cap \bigcup\{\mathcal{Y} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_2|\!)^{\curvearrowright}\mathcal{Y} \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_2)\}$. Now, we can finally prove the soundness for this case.

$$
\begin{array}{ll}
& \gamma_{\mathfrak{m}}((\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}) \\
= & \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright}_{\mathsf{C}} \\
& \gamma_{\mathfrak{m}}(\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{n} \trianglelefteq c_1\} \sqcap \bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{n} \trianglelefteq c_2\}) \\
= & \| \text{ co-aditivity of } \gamma_{\mathfrak{m}} \\
& \gamma_{\mathfrak{m}}(\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{n} \trianglelefteq c_1\}) \cap \gamma_{\mathfrak{m}}(\bigsqcup\{\mathfrak{n} \sqsubseteq \mathfrak{m} \mid (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{n} \trianglelefteq c_2\}) \\
\supseteq & \| \text{ monotonicity of } \gamma_{\mathfrak{m}} \\
& \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \mid \mathfrak{n} \sqsubseteq \mathfrak{m} \wedge (\!|\mathsf{a}_1|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{n} \trianglelefteq c_1\} \cap \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \mid \mathfrak{n} \sqsubseteq \mathfrak{m} \wedge (\!|\mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{n} \trianglelefteq c_2\} \\
\supseteq & \| \text{ monotonicity of } \gamma_{h^{\mathsf{Cprp}}} \text{ and soundness of } (\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{C}} \\
& \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \mid \mathfrak{n} \sqsubseteq \mathfrak{m} \wedge (\!|\mathsf{a}_1|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_1)\} \cap \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \mid \mathfrak{n} \sqsubseteq \mathfrak{m} \wedge (\!|\mathsf{a}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_2)\} \\
\supseteq & \| \text{ set theory and } \alpha_{\mathfrak{m}}(\mathbb{D}) \sqsubseteq \mathfrak{m} \\
& \bigcup \left\{ \gamma_{\mathfrak{m}}(\mathfrak{n}) \;\middle|\; \begin{array}{l} \mathfrak{n} \sqsubseteq \alpha_{\mathfrak{m}}(\mathbb{D}) \wedge \\ (\!|\mathsf{a}_1|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_1) \end{array} \right\} \cap \bigcup \left\{ \gamma_{\mathfrak{m}}(\mathfrak{n}) \;\middle|\; \begin{array}{l} \mathfrak{n} \sqsubseteq \alpha_{\mathfrak{m}}(\mathbb{D}) \wedge \\ (\!|\mathsf{a}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_2) \end{array} \right\} \\
= & \| \text{ Galois connection } \alpha_{\mathfrak{m}}, \gamma_{\mathfrak{m}} \\
& \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \mathbb{D} \mid (\!|\mathsf{a}_1|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_1)\} \cap \bigcup\{\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \mathbb{D} \mid (\!|\mathsf{a}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{n}) \subseteq \gamma_{h^{\mathsf{Cprp}}}(c_2)\}) \\
= &
\end{array}
$$

$$\bigcup\{\mathcal{X} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_1|\!)^{\curvearrowright}\mathcal{X} \subseteq \gamma_{h^{\mathsf{cprp}}}(c_1)\} \cap \bigcup\{\mathcal{X} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_2|\!)^{\curvearrowright}\mathcal{X} \subseteq \gamma_{h^{\mathsf{cprp}}}(c_2)\})$$

Hence we have that $\bigcup\{\mathcal{X} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_1|\!)^{\curvearrowright}\mathcal{X} \subseteq \gamma_{h^{\mathsf{cprp}}}(c_1)\} \cap \bigcup\{\mathcal{Y} \subseteq \mathbb{D} \mid (\!|\mathsf{a}_2|\!)^{\curvearrowright}\mathcal{Y} \subseteq \gamma_{h^{\mathsf{cprp}}}(c_2)\} \subseteq \gamma_{\mathfrak{m}}((\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})$ which, in turn, implies $(\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \subseteq \gamma_{\mathfrak{m}}((\!|\mathsf{a}_1 \bowtie \mathsf{a}_2|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})$ as requested. □

▶ *Theorem 27*

Proof. We just have to prove that the abstract hypersemantics $(\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}$ approximates the best correct approximation of the concrete hypersemantics $(\!|\mathsf{P}|\!)^{\curvearrowright}$ in $\mathsf{Mem}^{\mathsf{cprp}}$, namely $\alpha_{\mathfrak{m}} \circ (\!|\mathsf{P}|\!)^{\curvearrowright} \circ \gamma_{\mathfrak{m}} \sqsubseteq (\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}$. The proof is for structural induction on P.

Case P *and* $\mathfrak{m} = \mathfrak{m}_\bot$

$\alpha_{\mathfrak{m}}(\!|\mathsf{P}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}_\bot)$
$=$ ‖ definition of $\gamma_{\mathfrak{m}}$
$\alpha_{\mathfrak{m}}(\!|\mathsf{P}|\!)^{\curvearrowright}\varnothing$
$=$ ‖ definition of $(\!|\cdot|\!)^{\curvearrowright}$
$\alpha_{\mathfrak{m}}(\varnothing)$
$=$ ‖ definition of $\alpha_{\mathfrak{m}}$
$\mathfrak{m}_\bot$
$=$ ‖ definition of $(\!|\cdot|\!)^{\curvearrowright}_{\mathsf{C}}$
$(\!|\mathsf{P}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}_\bot$

Case $^{\mathsf{i}}\mathsf{skip}^{\mathsf{f}}$

$\alpha_{\mathfrak{m}}(\!|\,^{\mathsf{i}}\mathsf{skip}^{\mathsf{f}}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m})$
$=$ ‖ definition of $(\!|\cdot|\!)^{\curvearrowright}$
$\alpha_{\mathfrak{m}}\gamma_{\mathfrak{m}}(\mathfrak{m})$
$=$ ‖ reductivity of $\alpha_{\mathfrak{m}}\gamma_{\mathfrak{m}}$
$\mathfrak{m}$
$=$ ‖ definition of $(\!|\cdot|\!)^{\curvearrowright}_{\mathsf{C}}$
$(\!|\,^{\mathsf{i}}\mathsf{skip}^{\mathsf{f}}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}$

Case $^{\mathsf{i}}\mathsf{x} := \mathsf{a}^{\mathsf{f}}$

$\alpha_{\mathfrak{m}}(\!|\,^{\mathsf{i}}\mathsf{x} := \mathsf{a}^{\mathsf{f}}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m})$
$=$ ‖ definition of $(\!|\cdot|\!)^{\curvearrowright}$
$\alpha_{\mathfrak{m}}(\{\{\mathsf{m}[\mathsf{x} \hookleftarrow n] \mid \mathsf{m} \in X \wedge \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\})$
$=$ ‖$\alpha_{\mathfrak{m}} = \dot{\alpha}_{h^{\mathsf{cprp}}} \circ \alpha_{nnr}$ and definition of α_{nnr}
$\dot{\alpha}_{h^{\mathsf{cprp}}} \circ (\lambda \mathsf{y} \,.\, \{\{\mathsf{m}(\mathsf{y}) \mid \mathsf{m} \in X\} \mid X \in \{\{\mathsf{m}[\mathsf{x} \hookleftarrow n] \mid \mathsf{m} \in X \wedge \langle \mathsf{a}, \mathsf{m}\rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}\})$

$$
\begin{array}{ll}
& = \\
& \dot{\alpha}_{h^{\mathsf{cprp}}} \circ (\lambda \mathsf{y} \,.\, [\![\mathsf{y} = \mathsf{x} \mathbin{?} \{\{n \mid \exists \mathsf{m} \in X \,.\, \langle \mathsf{a}, \mathsf{m} \rangle \Downarrow^{\mathbb{Z}} n\} \mid X \in \gamma_{\mathsf{m}}(\mathfrak{m})\} \mathbin{:} \{\{\mathfrak{m}(\mathsf{y}) \mid \mathsf{m} \in X\} \mid X \in \gamma_{\mathsf{m}}(\mathfrak{m})\}]\!]) \\
\sqsubseteq & \quad \| \text{ soundness of } (\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{c}} \\
& \dot{\alpha}_{h^{\mathsf{cprp}}} \circ (\lambda \mathsf{y} \,.\, [\![\mathsf{y} = \mathsf{x} \mathbin{?} \gamma_{h^{\mathsf{cprp}}}(\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m} \mathbin{:} \{\{\mathfrak{m}(\mathsf{y}) \mid \mathsf{m} \in X\} \mid X \in \gamma_{\mathsf{m}}(\mathfrak{m})\}]\!]) \\
= & \quad \| \text{ definition of } \dot{\alpha}_{h^{\mathsf{cprp}}} \\
& \lambda \mathsf{y} \,.\, [\![\mathsf{y} = \mathsf{x} \mathbin{?} \alpha_{h^{\mathsf{cprp}}} \gamma_{h^{\mathsf{cprp}}}(\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m} \mathbin{:} \alpha_{h^{\mathsf{cprp}}}(\{\{\mathfrak{m}(\mathsf{y}) \mid \mathsf{m} \in X\} \mid X \in \gamma_{\mathsf{m}}(\mathfrak{m})\})]\!] \\
= & \quad \| \text{ definition of } \gamma_{\mathsf{m}} \\
& \lambda \mathsf{y} \,.\, [\![\mathsf{y} = \mathsf{x} \mathbin{?} \alpha_{h^{\mathsf{cprp}}} \gamma_{h^{\mathsf{cprp}}}(\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m} \mathbin{:} \alpha_{h^{\mathsf{cprp}}} \gamma_{h^{\mathsf{cprp}}}(\mathfrak{m}(\mathsf{y}))]\!] \\
\sqsubseteq & \quad \| \text{ reductivity of } \alpha_{h^{\mathsf{cprp}}} \gamma_{h^{\mathsf{cprp}}} \\
& \lambda \mathsf{y} \,.\, [\![\mathsf{y} = \mathsf{x} \mathbin{?} (\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m} \mathbin{:} \mathfrak{m}(\mathsf{y})]\!] \\
= & \\
& \mathfrak{m}[\mathsf{x} \leftarrowtail (\!|\mathsf{a}|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m}] \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright}_{\mathsf{c}} \\
& (\!|\, {}^{\mathsf{i}}\mathsf{x} := \mathsf{a}\, {}^{\mathsf{f}} \,|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m}
\end{array}
$$

Case ${}^{\mathsf{i}}\mathsf{c}\, {}^{\mathsf{l}} \,.\, {}^{\mathsf{k}}\mathsf{c}\, {}^{\mathsf{f}}$

$$
\begin{array}{ll}
& \alpha_{\mathsf{m}}(\!|\, {}^{\mathsf{i}}\mathsf{c}_1\, {}^{\mathsf{l}} \,.\, {}^{\mathsf{k}}\mathsf{c}_2\, {}^{\mathsf{f}} \,|\!)^{\curvearrowright} \gamma_{\mathsf{m}}(\mathfrak{m}) \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright} \\
& \alpha_{\mathsf{m}}(\!|\, {}^{\mathsf{k}}\mathsf{c}_2\, {}^{\mathsf{f}} \,|\!)^{\curvearrowright}(\!|\, {}^{\mathsf{i}}\mathsf{c}_1\, {}^{\mathsf{l}} \,|\!)^{\curvearrowright} \gamma_{\mathsf{m}}(\mathfrak{m}) \\
\sqsubseteq & \quad \| \text{ extensivity of } \gamma_{\mathsf{m}} \alpha_{\mathsf{m}} \\
& \alpha_{\mathsf{m}}(\!|\, {}^{\mathsf{k}}\mathsf{c}_2\, {}^{\mathsf{f}} \,|\!)^{\curvearrowright} \gamma_{\mathsf{m}} \alpha_{\mathsf{m}}(\!|\, {}^{\mathsf{i}}\mathsf{c}_1\, {}^{\mathsf{l}} \,|\!)^{\curvearrowright} \gamma_{\mathsf{m}}(\mathfrak{m}) \\
\sqsubseteq & \quad \| \text{ inductive hypothesis} \\
& (\!|\, {}^{\mathsf{k}}\mathsf{c}_2\, {}^{\mathsf{f}} \,|\!)^{\curvearrowright}_{\mathsf{c}}(\!|\, {}^{\mathsf{i}}\mathsf{c}_1\, {}^{\mathsf{l}} \,|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m} \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright}_{\mathsf{c}} \\
& (\!|\, {}^{\mathsf{i}}\mathsf{c}_1\, {}^{\mathsf{l}} \,.\, {}^{\mathsf{k}}\mathsf{c}_2\, {}^{\mathsf{f}} \,|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m}
\end{array}
$$

Case ${}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\, \mathsf{P}_1 \,\}\ \mathsf{else}\ \{\, \mathsf{P}_2 \,\}\, {}^{\mathsf{f}}$ (♣)

In order to prove this case we prove the equivalent formulation:

$$
\forall \mathsf{x} \in \mathsf{Var} \,.\, (\alpha_{\mathsf{m}}(\!|\, {}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\, \mathsf{P}_1 \,\}\ \mathsf{else}\ \{\, \mathsf{P}_2 \,\}\, {}^{\mathsf{f}} \,|\!)^{\curvearrowright} \gamma_{\mathsf{m}}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq (\!|\, {}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\, \mathsf{P}_1 \,\}\ \mathsf{else}\ \{\, \mathsf{P}_2 \,\}\, {}^{\mathsf{f}} \,|\!)^{\curvearrowright}_{\mathsf{c}} \mathfrak{m})(\mathsf{x})
$$

Take an arbitrary $\mathsf{x} \in \mathsf{Var}$. Then we have:

$$
\begin{array}{ll}
& (\alpha_{\mathsf{m}}(\!|\, {}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\, \mathsf{P}_1 \,\}\ \mathsf{else}\ \{\, \mathsf{P}_2 \,\}\, {}^{\mathsf{f}} \,|\!)^{\curvearrowright} \gamma_{\mathsf{m}}(\mathfrak{m}))(\mathsf{x}) \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright} \\
& (\alpha_{\mathsf{m}}(\{[\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg \mathsf{b}]\!]^{\curvearrowright} X \mid X \in \gamma_{\mathsf{m}}(\mathfrak{m})\}))(\mathsf{x})
\end{array}
$$

$= \quad \|$ definition of $\alpha_{\mathfrak{m}}$

$(\alpha_{h^{\text{cprp}}}(\{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright} X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}))(\mathsf{x})$

The proof continues by cases, recalling that $(\!|\, {}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\mathsf{f}}\,|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$ is:

$$(\!|\, \mathsf{x} \notin \mathsf{vars}^{:=}(\mathsf{P}_1) \cup \mathsf{vars}^{:=}(\mathsf{P}_2) \vee \mathfrak{n}(\mathsf{x}) \trianglelefteq \kappa \vee \mathsf{vars}^{\top}_{\mathfrak{m}}(\mathsf{b}) = \varnothing \mathbin{?} \mathfrak{n}(\mathsf{x}) : \top \,|\!)$$

where $\mathfrak{n} \triangleq (\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m} \sqcup (\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}$ and $\trianglelefteq$ is the strict version of $\trianglelefteq$.

Case $\mathsf{x} \notin \mathsf{vars}^{:=}(\mathsf{P}_1) \cup \mathsf{vars}^{:=}(\mathsf{P}_2)$

We have that $(\alpha_{\mathfrak{m}}(\!|\, {}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\mathsf{f}}\,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x})$ is equal to $\mathfrak{m}(\mathsf{x})$, since x has not been modified ($\mathsf{vars}^{:=}$ is an over-approximation). For the same reason, we have that $(\alpha_{\mathfrak{m}}(\!|\mathsf{P}_1|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \mathfrak{m}(\mathsf{x})$ and $(\alpha_{\mathfrak{m}}(\!|\mathsf{P}_2|\!)^{\curvearrowright}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \mathfrak{m}(\mathsf{x})$. Then, by soundness of $(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}$ and by inductive hypothesis on $(\!|\mathsf{P}_1|\!)^{\curvearrowright}$ (analogous for $\neg\mathsf{b}$ and P_2), we have that $\alpha_{\mathfrak{m}}(\!|\mathsf{P}_1|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \sqsubseteq (\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}$ and $\alpha_{\mathfrak{m}}(\!|\mathsf{P}_2|\!)^{\curvearrowright}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \sqsubseteq (\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}$. So, $(\alpha_{\mathfrak{m}}(\!|\mathsf{P}_1|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq ((\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$ and, similarly, $(\alpha_{\mathfrak{m}}(\!|\mathsf{P}_2|\!)^{\curvearrowright}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq ((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$. This implies $\mathfrak{m}(\mathsf{x}) \trianglelefteq \mathfrak{n}(\mathsf{x}) = ((\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \veebar ((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$, as we wanted.

Case $\mathfrak{n}(\mathsf{x}) = ((\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \veebar ((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \trianglelefteq \kappa$

Since b and $\neg\mathsf{b}$ cannot be false at the same time, we can have $((\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bot = ((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x})$ if and only if $\mathfrak{m} = \mathfrak{m}_{\bot}$. So $\alpha_{\mathfrak{m}}(\!|\, {}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\mathsf{f}}\,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) = \mathfrak{m}_{\bot}$ and $\mathfrak{n} = \mathfrak{m}_{\bot}$, hence $(\alpha_{\mathfrak{m}}(\!|\, {}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\mathsf{f}}\,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \bot \trianglelefteq \bot = \mathfrak{n}(\mathsf{x})$ trivially holds. Otherwise, we have that $((\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{n}$ and $((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \trianglelefteq \bar{n}$ (or the symmetric case). By soundness of $(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}$ and inductive hypothesis on $(\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}$ we have that $\alpha_{\mathfrak{m}}(\!|\mathsf{P}_1|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \sqsubseteq (\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}$. Since $\{[\![\mathsf{P}_1]\!]^{\curvearrowright} X \mid X \in (\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m})\} \subseteq (\!|\mathsf{P}_1|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m})$ and, by definition, $(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) = \{[\![\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \setminus \{\varnothing\}$, we have that $\{[\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \setminus \{\varnothing\} \subseteq (\!|\mathsf{P}|\!)^{\curvearrowright}(\!|\mathsf{b}|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m})$. By monotonicity of $\alpha_{\mathfrak{m}}$, we have that $\alpha(\{[\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\} \setminus \{\varnothing\}) \sqsubseteq (\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}$. Analogous for $(\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}$. From $((\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{n}$ we can conclude that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ we have $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X\} = \{n\}$. Similarly, $((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bot$ implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ we have $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright} X\} = \varnothing$ and $((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{n}$ implies that $\forall X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ we have $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright} X\} = \{n\}$. Then, we have that $\{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright} X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}$ is equal to $\{\{n\}\}$ in both cases. Thus, $\alpha_{h^{\text{cprp}}}(\{\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright} X\} \mid X \in \gamma_{\mathfrak{m}}(\mathfrak{m})\}) = \bar{n}$, which is approximated by κ, as required.

Case $\mathsf{vars}^{\top}_{\mathfrak{m}}(\mathsf{b}) = \varnothing$

We have that $\mathfrak{n}(\mathsf{x})$ is equal to κ or $\top$, otherwise we fall into the previous cases. If $\mathfrak{n}(\mathsf{x}) = \top$, which is the maximum, then $(\alpha_{\mathfrak{m}}(\!|\, {}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\mathsf{f}}\,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq \top$ trivially holds. If $\mathfrak{n}(\mathsf{x}) = \kappa$, it could be that $((\!|\mathsf{P}_1|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \bar{n}$ and $((\!|\mathsf{P}_2|\!)^{\curvearrowright}_{\mathsf{C}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) \in \{\bar{m}, \kappa\}$, with $n \neq m$ (or the symmetric case). The only way to falsify the proof is that $(\alpha_{\mathfrak{m}}(\!|\, {}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\mathsf{f}}\,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \top$. This means that there exists $X \in \gamma_{\mathfrak{m}}(\mathfrak{m})$ such that $\{\mathsf{m}(\mathsf{x}) \mid \mathsf{m} \in [\![\mathsf{P}_1]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright} X \cup [\![\mathsf{P}_2]\!]^{\curvearrowright}[\![\neg\mathsf{b}]\!]^{\curvearrowright} X\} \supseteq \{n, m\}$. Since $\mathsf{vars}^{\top}_{\mathfrak{m}}(\mathsf{b}) = \varnothing$, this happens if and only if $\mathfrak{m}(\mathsf{x})$ is already equal to $\top$ and, so $\mathsf{x} \notin \mathsf{vars}(\mathsf{b})$. All these facts imply that $\mathfrak{n}(\mathsf{x})$ cannot be κ when $(\alpha_{\mathfrak{m}}(\!|\, {}^{\mathsf{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\mathsf{f}}\,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) = \top$.

Case "otherwise"

Given that $(\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\underline{f}} \,|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m})(\mathsf{x}) = \top$, which is the maximum, we have that $(\alpha_{\mathfrak{m}}(\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}_1\,\}\ \mathsf{else}\ \{\,\mathsf{P}_2\,\}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}))(\mathsf{x}) \trianglelefteq \top$ trivially holds.

Since the variable x has been chosen arbitrarily, the proof holds for every variable. This terminates the case.

Case ${}^{\underline{i}}\mathsf{while}\ {}^{\underline{b}}\mathsf{k}\ \{\,\mathsf{P}\,\}^{\underline{f}}$

First, we can note that $\mathsf{H}^{\curvearrowright} = \lambda\mathcal{T}\,.\,\mathcal{X} \cup \{[\![\mathsf{P}]\!]^{\curvearrowright}[\![\mathsf{b}]\!]^{\curvearrowright}T \cup [\![\neg\mathsf{b}]\!]^{\curvearrowright}T \mid T \in \mathcal{T}\}$ coincides with the function $\lambda\mathcal{T}\,.\,\mathcal{X} \cup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}\mathcal{T}$, with $\underline{h}$ fresh label (it is indeed not necessary, since the post-conditions hypersemantics does not take into account labels). Now we need to derive a correct approximation of this latter:

$$
\begin{array}{ll}
& \lambda\mathfrak{n}\,.\,\alpha_{\mathfrak{m}}(\mathcal{X} \cup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{n})) \\
= & \quad \| \text{ additivity of } \alpha_{\mathfrak{m}} \\
= & \lambda\mathfrak{n}\,.\,\alpha_{\mathfrak{m}}(\mathcal{X}) \sqcup \alpha_{\mathfrak{m}}((\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{n})) \\
\sqsubseteq & \quad \| \text{ soundness of case } \clubsuit \\
& \lambda\mathfrak{n}\,.\,\alpha_{\mathfrak{m}}(\mathcal{X}) \sqcup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{n}
\end{array}
$$

The function $\lambda\mathfrak{n}\,.\,\alpha_{\mathfrak{m}}(\mathcal{X}) \sqcup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{n}$ is sound, since it approximates the bca of $\mathsf{H}^{\curvearrowright}$. The bca is correct even at fixpoint, namely we have:

$$\alpha_{\mathfrak{m}}(\mathsf{lfp}^{\subseteq}_{\varnothing}\,\mathsf{H}^{\curvearrowright}) \sqsubseteq \mathsf{lfp}^{\sqsubseteq}_{\mathfrak{m}_{\bot}}\,\lambda\mathfrak{n}\,.\,\alpha_{\mathfrak{m}}(\mathcal{X}) \sqcup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{n} \tag{A.4}$$

Now we can continue the proof as follows.

$$
\begin{array}{ll}
& \alpha_{\mathfrak{m}}(\!|\, {}^{\underline{i}}\mathsf{while}\ {}^{\underline{k}}\mathsf{b}\ \{\,\mathsf{P}\,\}^{\underline{f}} \,|\!)^{\curvearrowright}\gamma_{\mathfrak{m}}(\mathfrak{m}) \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright} \\
& \alpha_{\mathfrak{m}}(\!|\neg\mathsf{b}|\!)^{\curvearrowright}(\mathsf{lfp}^{\subseteq}_{\varnothing}\,\lambda\mathcal{T}\,.\,\gamma_{\mathfrak{m}}(\mathfrak{m}) \cup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}\mathcal{T}) \\
\sqsubseteq & \quad \| \text{ soundness of } (\!|\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}} \\
& (\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}\alpha_{\mathfrak{m}}(\mathsf{lfp}^{\subseteq}_{\varnothing}\,\lambda\mathcal{T}\,.\,\gamma_{\mathfrak{m}}(\mathfrak{m}) \cup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}\mathcal{T}) \\
\sqsubseteq & \quad \| \text{ fixpoint approximation (equation A.4)} \\
& (\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}(\mathsf{lfp}^{\sqsubseteq}_{\mathfrak{m}_{\bot}}\,\lambda\mathfrak{n}\,.\,\alpha_{\mathfrak{m}}\gamma_{\mathfrak{m}}(\mathfrak{m}) \sqcup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{n}) \\
= & \quad \| \text{ reductivity of } \alpha_{\mathfrak{m}}\gamma_{\mathfrak{m}} \\
& (\!|\neg\mathsf{b}|\!)^{\curvearrowright}_{\mathsf{C}}(\mathsf{lfp}^{\sqsubseteq}_{\mathfrak{m}_{\bot}}\,\lambda\mathfrak{n}\,.\,\mathfrak{m} \sqcup (\!|\, {}^{\underline{i}}\mathsf{if}\ \mathsf{b}\ \mathsf{then}\ \{\,\mathsf{P}\,\}\ \mathsf{else}\ \{\ {}^{\underline{h}}\mathsf{skip}^{\underline{h}}\ \}^{\underline{f}} \,|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{n}) \\
= & \quad \| \text{ definition of } (\!|\cdot|\!)^{\curvearrowright}_{\mathsf{C}} \\
& (\!|\, {}^{\underline{i}}\mathsf{while}\ {}^{\underline{k}}\mathsf{b}\ \{\,\mathsf{P}\,\}^{\underline{f}} \,|\!)^{\curvearrowright}_{\mathsf{C}}\mathfrak{m}
\end{array}
$$

□

Bibliography

Adi, Kamel and Mourad Debbabi (2003). "Abstract Interpretation for Proving Secrecy Properties in Security Protocols". In: *Electronic Notes in Theoretical Computer Science* 55.1, pp. 25–50. ISSN: 1571-0661. DOI: http://dx.doi.org/10.1016/S1571-0661(04)00243-9.

Agrawal, S. and B. Bonakdarpour (2016). "Runtime Verification of k-Safety Hyperproperties in HyperLTL". In: *Proc. of CSF*, pp. 239–252.

Alpern, Bowen and Fred B. Schneider (1985). "Defining liveness". In: *Information Processing Letters* 21.4, pp. 181–185. ISSN: 0020-0190. DOI: http://dx.doi.org/10.1016/0020-0190(85)90056-0. URL: http://www.sciencedirect.com/science/article/pii/0020019085900560.

Antonopoulos, T. et al. (2017). "Decomposition instead of self-composition for proving the absence of timing channels". In: *Proc. of PLDI*, pp. 362–375.

Assaf, Mounir et al. (2017). "Hypercollecting semantics and its application to static analysis of information flow". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pp. 874–887. URL: http://dl.acm.org/citation.cfm?id=3009889.

Balliu, M.; M. Dam and G. Le Guernic (2011). "Epistemic Temporal Logic for Information Flow Security". In: *Proc. of PLAS*, pp. 1–12.

Barthe, G.; P. R. D'Argenio and T. Rezk (2004). "Secure information flow by self-composition". In: *Proc. of CSF*, pp. 100–114.

Chang, Edward; Zohar Manna and Amir Pnueli (1992). *The safety-progress classification*. Tech. rep. Stanford University, Dept. of Computer Science.

Clarke, Edmund M.; Orna Grumberg and David E. Long (1994). "Model Checking and Abstraction". In: *ACM Trans. Program. Lang. Syst.* 16.5, pp. 1512–1542. ISSN: 0164-0925. DOI: 10.1145/186025.186051.

Clarkson, M. R. et al. (2014). "Temporal Logics for Hyperproperties". In: *Proc. of POST*.

Clarkson, Michael R. and Fred B. Schneider (2010). "Hyperproperties". In: *J. Comput. Secur.* 18.6, pp. 1157–1210. ISSN: 0926-227X. URL: http://dl.acm.org/citation.cfm?id=1891823.1891830.

Cohen, Ellis (1977). "Information Transmission in Computational Systems". In: *SIGOPS Oper. Syst. Rev.* 11.5, pp. 133–139. ISSN: 0163-5980. DOI: 10.1145/1067625.806556. URL: http://doi.acm.org/10.1145/1067625.806556.

Comini, Marco; Giorgio Levi and Maria Chiara Meo (2001). "A Theory of Observables for Logic Programs". In: *Information and Computation* 169.1, pp. 23–80. ISSN: 0890-5401. DOI: http://dx.doi.org/10.1006/inco.2000.3024.

Cousot, Patrick (1997). "Types as Abstract Interpretation". In: *Conference Record of POPL 97: The 24TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Paris, France*. ACM, pp. 316–331.

— (2002). "Constructive design of a hierarchy of semantics of a transition system by abstract interpretation". In: *Theor. Comput. Sci.* 277.1-2, pp. 47–103.

Cousot, Patrick and Radhia Cousot (1976). "Static determination of dynamic properties of programs". In: *Proceedings of the Second International Symposium on Programming*. Dunod, Paris, France, pp. 106–130.

— (1977). "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints". In: *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. POPL '77. Los Angeles, California: ACM, pp. 238–252. DOI: 10.1145/512950.512973. URL: http://doi.acm.org/10.1145/512950.512973.

— (1979a). "Constructive versions of Tarski's fixed point theorems." In: *Pacific Journal of Mathematics* 82.1, pp. 43–57.

— (1979b). "Systematic Design of Program Analysis Frameworks". In: *Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. POPL '79. San Antonio, Texas: ACM, pp. 269–282. DOI: 10.1145/567752.567778. URL: http://doi.acm.org/10.1145/567752.567778.

— (1992). "Abstract Interpretation Frameworks". In: *J. Log. Comput.* 2.4, pp. 511–547. DOI: 10.1093/logcom/2.4.511. URL: http://dx.doi.org/10.1093/logcom/2.4.511.

— (1993). "Galois Connection Based Abstract Interpretations for Strictness Analysis (invited paper)". In: *Proceedings of the International Conference on Formal Methods in Programming and their Applications, Academgorodok, Novosibirsk, Russia*. Ed. by D. Bjørner; M. Broy and I.V. Pottosin. Lecture Notes in Computer Science 735. Springer-Verlag, Berlin, Germany, pp. 98–127. ISBN: 978-3-540-57316-6. DOI: 10.1007/BFb0039703.

— (2001). "A Case Study in Abstract Interpretation Based Program Transformation". In: *Electronic Notes in Theoretical Computer Science* 45, pp. 41–64. ISSN: 1571-0661. DOI: http://dx.doi.org/10.1016/S1571-0661(04)80954-X. URL: http://www.sciencedirect.com/science/article/pii/S157106610480954X.

— (2002). "Systematic Design of Program Transformation Frameworks by Abstract Interpretation". In: *Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '02. Portland, Oregon: ACM, pp. 178–190. ISBN: 1-58113-450-9. DOI: 10.1145/503272.503290.

— (2012). "An Abstract Interpretation Framework for Termination". In: *Conference Record of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. Philadelphia, PA: ACM Press, New York, pp. 245–258.

Cousot, Patrick; Roberto Giacobazzi and Francesco Ranzato (2019). "A^2I: Abstract^2 Interpretation". In: *Proc. ACM Program. Lang.* 3.POPL, 42:1–42:31. ISSN: 2475-1421. DOI: 10.1145/3290355.

Dalla Preda, Mila and Michele Pasqua (2016). "Software Watermarking: A Semantics-based Approach". In: *6th Workshop on Numerical and Symbolic Abstract Domains (NSAD 2016), Edinburgh, Scotland, September 11, 2016, Proceedings*, pp. 71–85. DOI: 10.1007/978-3-319-66706-5_12. URL: https://doi.org/10.1007/978-3-319-66706-5_12.

— (2018). "Semantics-based software watermarking by abstract interpretation". In: *Mathematical Structures in Computer Science*, pp. 1–50. DOI: 10.1017/S0960129518000038.

Dimitrova, R. et al. (2012). "Model Checking Information Flow in Reactive Systems". In: *Proc. of VMCAI*, pp. 169–185.

Eilenberg, Samuel (1974). *Automata, Languages, and Machines*. Orlando, FL, USA: Academic Press, Inc. ISBN: 0122340019.

Emerson, Ernest A. (1983). "Alternative semantics for temporal logics". In: *Theoretical Computer Science* 26.1, pp. 121–130. ISSN: 0304-3975. DOI: http://dx.doi.org/10.1016/0304-3975(83)90082-8. URL: http://www.sciencedirect.com/science/article/pii/0304397583900828.

Finkbeiner, B.; M. N. Rabe and C. Sánchez (2015). "Algorithms for Model Checking HyperLTL and HyperCTL". In: *Proc. of CAV*, pp. 30–48.

Giacobazzi, Roberto; Francesco Logozzo and Francesco Ranzato (2015). "Analyzing Program Analyses". In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pp. 261–273.

Giacobazzi, Roberto and Isabella Mastroeni (2004). "Abstract Non-interference: Parameterizing Non-interference by Abstract Interpretation". In: *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '04. Venice, Italy: ACM, pp. 186–197. ISBN: 1-58113-729-X. DOI: 10.1145/964001.964017. URL: http://doi.acm.org/10.1145/964001.964017.

— (2018). "Abstract Non-Interference: A Unifying Framework for Weakening Information-flow". In: *ACM Trans. Priv. Secur.* 21.2, pp. 1–31.

Giacobazzi, Roberto; Francesco Ranzato and Francesca Scozzari (2000). "Making Abstract Interpretations Complete". In: *J. ACM* 47.2, pp. 361–416. ISSN: 0004-5411. DOI: 10.1145/333979.333989. URL: http://doi.acm.org/10.1145/333979.333989.

Goguen, Joseph A. and José Meseguer (1982). "Security Policies and Security Models." In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, pp. 11–20. ISBN: 0-8186-0410-7. DOI: http://doi.ieeecomputersociety.org/10.1109/SP.1982.10014.

Helm, Richard; Kim Marriott and Martin Odersky (1995). "Spatial Query Optimization: From Boolean Constraints to Range Queries". In: *Journal of Computer and System Sciences* 51.2, pp. 197–210. ISSN: 0022-0000. DOI: http://dx.doi.org/10.1006/jcss.1995.1061.

Hodován, Renáta and Akos Kiss (2018). *Grammarinator - ANTLRv4 grammar-based test generator*. URL: https://github.com/renatahodovan/grammarinator.

Hunt, Sebastian and Isabella Mastroeni (2005). "The PER Model of Abstract Non-interference". In: *Static Analysis, 12th International Symposium, SAS 2005, London, UK, September 7-9, 2005, Proceedings*, pp. 171–185.

Kuratowski, Kazimierz (1967). "Topology: Volume I". In: *ZAMM - Journal of Applied Mathematics and Mechanics* 47.8, pp. 560–560. ISSN: 1521-4001. DOI: 10.1002/zamm.19670470839. URL: http://dx.doi.org/10.1002/zamm.19670470839.

Manna, Zohar and Amir Pnueli (1995). *Temporal Verification of Reactive Systems: Safety*. New York, NY, USA: Springer-Verlag New York, Inc. ISBN: 0-387-94459-1.

Markowsky, George (1976). "Chain-complete posets and directed sets with applications." In: *Algebra Univers.* 6, pp. 53–68. DOI: 10.1007/BF02485815. URL: http://dx.doi.org/10.1007/BF02485815.

Mastroeni, Isabella and Michele Pasqua (2017). "Hyperhierarchy of Semantics - A Formal Framework for Hyperproperties Verification". In: *Static Analysis - 24th International Symposium, (SAS 2017), New York City, USA, August 30 – September 1, 2017, Proceedings*, pp. 232–252. DOI: 10.1007/978-3-319-66706-5_12. URL: https://doi.org/10.1007/978-3-319-66706-5_12.

Mastroeni, Isabella and Michele Pasqua (2018). "Verifying Bounded Subset-Closed Hyperproperties". In: *Static Analysis*. Ed. by Andreas Podelski. Cham: Springer International Publishing, pp. 263–283. ISBN: 978-3-319-99725-4.

— (2019). "Statically Analyzing Information Flows – An Abstract Interpretation-based Hyperanalysis for Non-Interference". In: *Proceedings of the 34rd Annual ACM Symposium on Applied Computing*. SAC '19. Limassol, Cyprus: ACM, pp. 2215–2223. ISBN: 978-1-4503-5933-7. DOI: 10.1145/3297280.3297498.

McCullough, Daryl (1987). "Specifications for Multi-Level Security and a Hook-Up". In: *Security and Privacy, 1987 IEEE Symposium on*, pp. 161–161. DOI: 10.1109/SP.1987.10009.

McLean, John (1996). "A general theory of composition for a class of 'possibilistic' properties". In: *IEEE Transactions on Software Engineering* 22.1, pp. 53–67. ISSN: 0098-5589. DOI: 10.1109/32.481534.

Pasqua, Michele and Isabella Mastroeni (2017). "On topologies for (hyper)properties". In: *18th Italian Conference on Theoretical Computer Science (ICTCS 2017), Naples, Italy, September 26 – 28, 2017, Proceedings*, pp. 1–12. URL: http://ceur-ws.org/Vol-1949/ICTCSpaper13.pdf.

Perdrix, Simon (2008). "Quantum Entanglement Analysis Based on Abstract Interpretation". In: *Static Analysis: 15th International Symposium (SAS 2008). Proceedings*. Springer Berlin Heidelberg, pp. 270–282. ISBN: 978-3-540-69166-2. DOI: 10.1007/978-3-540-69166-2_18.

Plaisted, David A. (1981). "Theorem proving with abstraction". In: *Artificial Intelligence* 16.1, pp. 47–108. ISSN: 0004-3702. DOI: http://dx.doi.org/10.1016/0004-3702(81)90015-1.

Plotkin, Gordon D. (1976). "A powerdomain construction". In: *SIAM J. OF COMPUTING*.

Ranzato, Francesco and Francesco Tapparo (2004). "Strong Preservation as Completeness in Abstract Interpretation". In: *ESOP, Proceedings*. Ed. by David Schmidt, pp. 18–32.

Roşu, Grigore (2012). "On Safety Properties and Their Monitoring". In: *Scientific Annals of Computer Science* 22.2, pp. 327–365. DOI: http://dx.doi.org/10.7561/SACS.2012.2.327.

Sousa, M. and I. Dillig (2016). "Cartesian Hoare Logic for Verifying K-safety Properties". In: *Proc. of PLDI*, pp. 57–69.

Stallings, William and Lawrie Brown (2007). *Computer Security: Principles and Practice*. 1st. Upper Saddle River, NJ, USA: Prentice Hall Press. ISBN: 0136004245, 9780136004240.

Sutherland, David (1986). "A Model of Information". In: *Proc. 9th National Security Conference*. Gaithersburg, Md., pp. 175–183.

Terauchi, T. and A. Aiken (2005). "Secure Information Flow As a Safety Problem". In: *Proc. of SAS*, pp. 352–367.

Urban, Caterina and Peter Müller (2018). "An Abstract Interpretation Framework for Input Data Usage". In: *Programming Languages and Systems*. Ed. by Amal Ahmed. Cham: Springer International Publishing, pp. 683–710.

Volpano, D.; C. Irvine and G. Smith (1996). "A Sound Type System for Secure Flow Analysis". In: *J. Comput. Secur.* 4, pp. 167–187.

Zdancewic, Steve and Andrew C. Myers (2003). "Observational determinism for concurrent program security". In: *Proc. of IEEE Computer Security Foundations Workshop*. Pacific Grove, CA, pp. 29–43.