Verifying Bounded Subset-Closed Hyperproperties

Isabella Mastroeni and Michele Pasqua

University of Verona - Dipartimento di Informatica
Strada le Grazie 15, 37134, Verona, Italy
(isabella.mastroeni|michele.pasqua)@univr.it

Abstract. Hyperproperties are quickly becoming very popular in the context of systems security, due to their expressive power. They differ from classic trace properties since they are represented by sets of sets of executions instead of sets of executions. This allows us, for instance, to capture information flow security specifications, which cannot be expressed as trace properties, namely as predicates over single executions. In this work, we reason about how it is possible to move standard abstract interpretation-based static analysis methods, designed for trace properties, towards the verification of hyperproperties. In particular, we focus on the verification of *bounded subset-closed hyperproperties* which are easier to verify than generic hyperproperties. It turns out that a lot of interesting specifications (e.g., Non-Interference) lie in this category.

1 Introduction

When reasoning about systems executions, a key point is the degree of approximation given by the choice of the semantics used to represent computations. Since its origin in 1977, abstract interpretation [12] has been widely used to describe and formalize approximate systems computations in many different areas of computer science and, in particular, in program verification. In this direction, comparative semantics consists in comparing semantics at different levels of abstraction, always by abstract interpretation [11, 17]. 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 verification expressiveness, but also because not all the semantics are suitable for proving any possible specification of interest. In other words, the semantics must describe at least the program features involved by the specification of interest. For instance, in the security context, there are specifications that can be expressed as trace properties, like Access Control, and others which cannot, like Non-Interference[1]. In this latter case, it is necessary to specify it as an *hyperproperty*. Intuitively, a trace property is defined exclusively in terms of individual executions and, in general, do not specify any relation between different executions of a system. Instead,

[1] Access Control is defined over systems (reachable) states. Non-Interference, instead, is defined over systems input/output (I/O) traces.

an hyperproperty specifies the set of sets of system executions allowed by the specification, therefore expressing relations between executions. In [9] it is stated that hyperproperties are able to define every possible specification concerning systems modeled as sets of traces (of states).

Unfortunately, hyperproperties are not, in general, precisely verifiable with standard methods, e.g., with standard abstract interpretation-based static analyses. In [25] we face the problem of formally verifying hyperproperties from a very general point of view, by providing several ingredients necessary for tackling the problem of verifying hyperproperties. We introduce a classification of hyperproperties distinguishing between those that can be "precisely" analyzed with standard program analysis (*trace* hyperproperties), those that technically could be analyzed with standard methods (with potentially unsatisfactory results) but for which an analysis at hyperlevel could gain precision (*subset-closed* hyperproperties) and those for which standard static analyses cannot work properly (all other hyperproperties). Then we formally describe the hyperlevel of semantics by integrating the hyperlevel in the hierarchy of semantics [11], providing a formal framework for reasoning about hyperproperties of systems.

Contribution. In the present work, program verification of hyperproperties, which was the main motivation of [25], becomes the central focus. First of all, 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 flow) but their verification is made easier. In particular, 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 analysis has to move to the hyperlevel for reducing the loss of precision, which, at the standard level, could make the analysis useless (even if it is still possible).
At this point, we wonder how we can lift, not the whole concrete semantics (as in [25]), but the interpreter computing the collecting semantics. We propose a general technique for lifting collecting semantics and we observe that the semantics proposed in [9] is a particular instance of our general approach. 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.
Finally, as it happens in standard analysis where the collecting semantics is approximated in a domain of observations, we aim at defining hyper abstract domains, in order to approximate the collecting hypersemantics. With this aim in mind, we propose a methodology for lifting abstract domains to the hyperlevel.

Structure of the paper. In Sect. 2, we briefly recall the concept of hyperproperty and the issue of its verification. Then we introduce the new notion of bounded subset-closed hyperproperty. In Sect. 3, we deal with the problem of lifting the collecting semantics of a given static analysis at the level of sets of sets. In Sect. 4, we describe general patterns for building (hyper) domains, suitable for the verification of hyperproperties. In Sect. 5, we show how to instantiate the methodologies introduced, in order to obtain sound and complete static analyses for bounded subset-closed hyperproperties. Finally, in the last two sections, we have related works, future research directions and conclusions.

2 Concerning Hyperproperties Verification

Let $\mathbb{D}\mathbb{E}\mathbb{N}$ be the set of all possible denotations for systems executions (e.g., reachable states, pairs of input and output states, finite sequences of states, etc.). We recall that while a trace property $\mathfrak{P}$, i.e., a property whose satisfaction depends on single executions, is modeled as the set of all executions satisfying it (hence $\mathfrak{P} \in \wp(\mathbb{D}\mathbb{E}\mathbb{N})$), an hyperproperty $\mathfrak{H}\mathfrak{p}$, verifiable on sets of executions, is modeled as the set of all sets of executions satisfying it (hence $\mathfrak{H}\mathfrak{p} \in \wp(\wp(\mathbb{D}\mathbb{E}\mathbb{N}))$).

2.1 Bounded Subset-Closed Hyperproperties

In [25], we define the following hyperproperties classification:

$$\mathtt{TRC}^{\mathtt{H}} \triangleq \{\mathfrak{H}\mathfrak{p} \in \wp(\wp(\mathbb{D}\mathbb{E}\mathbb{N})) \mid \wp(\textstyle\bigcup \mathfrak{H}\mathfrak{p}) = \mathfrak{H}\mathfrak{p}\}$$
$$\mathtt{SSC}^{\mathtt{H}} \triangleq \{\mathfrak{H}\mathfrak{p} \in \wp(\wp(\mathbb{D}\mathbb{E}\mathbb{N})) \mid X \in \mathfrak{H}\mathfrak{p} \Rightarrow (\forall Y \subseteq X \,.\, Y \in \mathfrak{H}\mathfrak{p})\}$$

The first are called *trace hyperproperties* and the second *subset-closed hyperproperties*. Trace hyperproperties are isomorphic to trace properties, namely they corresponds to all and only the hyperproperties verifiable on single executions, i.e., they do not need the comparison of different executions. Subset-closed hyperproperties are those hyperproperties that can be refuted just by showing an arbitrary subset of the semantics that does not satisfies the hyperproperty (witness of refutation).

In this paper, we introduce a stronger notion of subset-closed hyperproperty, allowing us to further restrict the search space for possible refuting witnesses.

Definition 1 (k-Bounded Subset-Closed Hyperproperty).

$$\mathtt{SSC}^{\mathtt{H}}_{k} \triangleq \{\mathfrak{H}\mathfrak{p} \in \wp(\wp(\mathbb{D}\mathbb{E}\mathbb{N})) \mid X \notin \mathfrak{H}\mathfrak{p} \Leftrightarrow (\exists T_k \subseteq X \,.\, (|T_k| \leq k \wedge T_k \notin \mathfrak{H}\mathfrak{p}))\}$$

The set T_k is the *witness of refutation*, namely a set of traces of cardinality at most $k \in \mathbb{N}$ violating the property. 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 $\mathfrak{H}\mathfrak{p} \in \mathtt{SSC}^{\mathtt{H}}_{k}$, if we find $\{\mathfrak{d}^1, \mathfrak{d}^2, \dots \mathfrak{d}^k\} \subseteq X$ such that $\{\mathfrak{d}^1, \mathfrak{d}^2, \dots \mathfrak{d}^k\} \notin \mathfrak{H}\mathfrak{p}$, then we can imply that $X \notin \mathfrak{H}\mathfrak{p}$. Hence $X \models \mathfrak{H}\mathfrak{p}$, meaning X satisfies $\mathfrak{H}\mathfrak{p}$, iff $\{\{\mathfrak{d}^1, \mathfrak{d}^2, \dots \mathfrak{d}^k\} \mid \mathfrak{d}^1, \mathfrak{d}^2, \dots \mathfrak{d}^k \in X\} \subseteq \mathfrak{H}\mathfrak{p}$. Clearly, it turns out that a trace hyperproperty is 1-bounded, namely $\mathtt{TRC}^{\mathtt{H}} = \mathtt{SSC}^{\mathtt{H}}_{1}$.

It is also clear that the union of all the k-bounded subset-closed hyperproperties and the unbounded subset-closed hyperproperties (i.e., those with $k = \omega$) is precisely the set of all the subset-closed hyperproperties.

Proposition 1. *It holds that* $\mathtt{SSC}^{\mathtt{H}} = \bigcup_{k \leq \omega} \mathtt{SSC}^{\mathtt{H}}_{k}$.

For every $\mathfrak{Hp} \in \texttt{SSC}^{\texttt{H}}$ we can define a refuting set $R_{\mathfrak{Hp}}$, namely a set of sets of traces representing the witnesses for refuting the hyperproperty. These sets are inspired by the prefixes representing the "bad thing" in safety properties. It is possible to define different refuting sets for a given hyperproperty, since when a set $X \notin \mathfrak{Hp}$ then we have that $X \cup Y \notin \mathfrak{Hp}$, by subset-closure. A $\texttt{SSC}^{\texttt{H}}$ hyperproperty $\mathfrak{Hp}$ is violated iff the given set of traces is a superset of an element in $R_{\mathfrak{Hp}}$. So $\mathfrak{Hp}$ can be characterized as:

$$\forall X \in \wp(\mathbb{DEN}) \,.\, (\exists T_r \in R_{\mathfrak{Hp}} \,.\, T_r \subseteq X) \Leftrightarrow X \notin \mathfrak{Hp} \tag{1}$$

If $\mathfrak{Hp} \in \texttt{SSC}^{\texttt{H}}_k$ (i.e., it is bounded) then we can define the *minimal* refuting set $R^{\min}_{\mathfrak{Hp}}$ (i.e., the one containing the sets with minimal cardinality) characterizing the hyperproperty. This means that for every set violating the hyperproperty, $R^{\min}_{\mathfrak{Hp}}$ contains only its minimal representative (w.r.t. $\subseteq$). In particular, every element in $R^{\min}_{\mathfrak{Hp}}$ has cardinality k.

Example 1. Let $\mathsf{St} = \mathsf{Var} \to \mathbb{Z}$ and $\mathbb{DEN} = \mathsf{St} \times \mathsf{St}$. Non-Interference [10, 21], parametric on a security variables typing $\Gamma \in \mathsf{Var} \to \{\mathsf{L}, \mathsf{H}\}$, is:

$$\texttt{NI} \triangleq \{X \in \wp(\mathbb{DEN}) \mid \forall \mathfrak{d}, \mathfrak{d}' \in X \,.\, (\mathfrak{d}_{\vdash} =_{\mathsf{L}} \mathfrak{d}'_{\vdash} \Rightarrow \mathfrak{d}_{\dashv} =_{\mathsf{L}} \mathfrak{d}'_{\dashv})\}$$

where $\mathfrak{d}_{\vdash}$ and $\mathfrak{d}_{\dashv}$ are the projections on the first and last element of the pair $\mathfrak{d}$, respectively. The equivalence $=_{\mathsf{L}}$ holds for memories agreeing on the values of public (L) variables. $\texttt{NI}$ is in $\texttt{SSC}^{\texttt{H}}_2$, namely $X \models \texttt{NI}$ iff $\{\{\mathfrak{d}, \mathfrak{d}'\} \mid \mathfrak{d}, \mathfrak{d}' \in X\} \subseteq \texttt{NI}$. Hence, if we find a pair of interfering executions, i.e., $\{\mathfrak{d}, \mathfrak{d}'\} \notin \texttt{NI}$, then we prove that $X \not\models \texttt{NI}$. Indeed, the minimal refuting set for Non-Interference is:

$$R^{\min}_{\texttt{NI}} \triangleq \left\{\{\mathfrak{d}, \mathfrak{d}'\} \in \wp(\mathbb{DEN}) \;\middle|\; \mathfrak{d}_{\vdash} =_{\mathsf{L}} \mathfrak{d}'_{\vdash} \wedge \mathfrak{d}_{\dashv} \neq_{\mathsf{L}} \mathfrak{d}'_{\dashv}\right\}$$

End example.

Note that substituting $\subseteq$ with the prefix-set relation $\leqslant$[2] in (1) we obtain the minimal refuting set for an hypersafety.

2.2 The Safety/Liveness Dichotomy

In the context of trace properties, a particular kind of properties are the *safety* ones [2], 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 [2]. 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 for satisfying the property. Dually, liveness (trace) properties express the fact that "something good eventually happens", namely the systems satisfying a

[2] Here $X \leqslant Y$ iff for every $d \in X$ exists $d' \in Y$ such that d is a prefix of d'[9].

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 [9, 28].

Another particular class of hyperproperties are the *k-safety hyperproperties* (or k-hypersafety). They are safety hyperproperties in which the "bad thing" never involves more than k executions [9]. This means that it is possible to check the violation of a k-hypersafety just observing a set of k executions (note that 1-hypersafeties 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 [9].

It turns out that all hypersafety are subset-closed [9]. But 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 also other hyperliveness which are bounded, as we can see in the following example.

Example 2. Suppose now that executions denotations are infinite sequences of states, namely $\mathbb{D}\mathbb{E}\mathbb{N} = \mathsf{St}^{\omega}$. 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:

$$\mathtt{SyncR} \triangleq \left\{ X \subseteq \mathsf{St}^{\omega} \;\middle|\; \forall \mathfrak{d}, \mathfrak{d}' \in X \, \forall i \in \mathbb{N} \, . \begin{array}{l} (\mathrm{Req}(\mathfrak{d}, i) \wedge \mathrm{Req}(\mathfrak{d}', i)) \Rightarrow \\ \exists j \in \mathbb{N} \, . \, (\mathrm{Resp}(\mathfrak{d}, i, j) \wedge \mathrm{Resp}(\mathfrak{d}', i, j)) \end{array} \right\}$$

It is easy to note that $\mathtt{SyncR}$ 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. *End example.*

Example 2 proves that there are hyperproperties which are not k-hypersafety but are k-bounded subset-closed (other than the trivial liveness properties). In Fig. 1 we have a graphical representation of how we can classify hyperproperties, w.r.t. the safety/liveness dichotomy and subset-closure.

2.3 Exploring the Hyperproperties Verification Issue

Let us now consider as systems the programs P written in a given imperative deterministic programming language, with assignments, conditionals and while

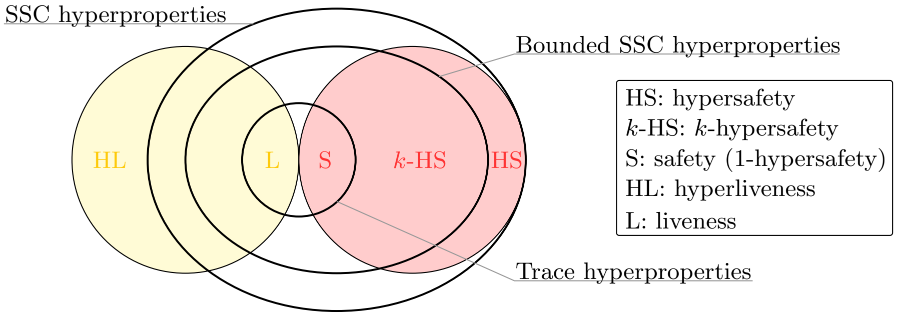


Fig. 1. Classification of Hyperproperties.

loops. Let $\mathbb{D}\mathbb{EN}$ be the domain of denotations for program behaviors, then $\mathcal{S}[\![\mathsf{P}]\!] \in \wp(\mathbb{D}\mathbb{EN})$ denotes the semantics of P, intended as the strongest trace property of P. In this case properties of P are those in $\wp(\mathbb{D}\mathbb{EN})$, while hyperproperties of P are those in $\wp(\wp(\mathbb{D}\mathbb{EN}))$. For instance, if $\mathbb{D}\mathbb{EN} = \mathsf{St} \triangleq \mathsf{Var} \to \mathbb{Z}$, i.e., states are represented as mappings from variables to values, we cannot express Non-Interference (comparing traces of executions sharing the same low inputs) but we can express Access Control (checking whether in a program point an access has been granted or not). For defining Non-Interference we need, at least, denotations representing the programs input/output (I/O) relation, e.g., $\mathbb{D}\mathbb{EN} = \mathsf{St} \times \mathsf{St}$.

In the context of program verification of (trace) properties, the satisfaction is given by set inclusion, i.e., a program P satisfies a property $\mathfrak{P} \in \wp(\mathbb{D}\mathbb{EN})$, written $\mathsf{P} \models \mathfrak{P}$, iff $\mathcal{S}[\![\mathsf{P}]\!] \subseteq \mathfrak{P}$. For hyperproperties, $\mathsf{P} \models \mathfrak{Hp}$ iff $\mathcal{S}[\![\mathsf{P}]\!] \in \mathfrak{Hp}$ iff $\{\mathcal{S}[\![\mathsf{P}]\!]\} \subseteq \mathfrak{Hp}$. In particular, $\{\mathcal{S}[\![\mathsf{P}]\!]\} \in \wp(\wp(\mathbb{D}\mathbb{EN}))$ is the strongest hyperproperty of P [25].

In general, the semantics of a program is not computable, hence practical verification methods rely on approximations. In standard trace properties verification, we compute an over-approximation, e.g., by abstract interpretation, $O \supseteq \mathcal{S}[\![\mathsf{P}]\!]$ which is such that, if $O \subseteq \mathfrak{P}$, then we can soundly imply $\mathsf{P} \models \mathfrak{P}$. Unfortunately, over-approximations on $\wp(\mathbb{D}\mathbb{EN})$ do not always work properly with hyperproperties. In particular, it formally does work for $\mathfrak{Hp} \in \mathtt{SSC}^{\mathtt{H}}$, in fact if we prove that $O \supseteq \mathcal{S}[\![\mathsf{P}]\!]$ and $O \in \mathfrak{Hp}$, then by subset-closure of $\mathfrak{Hp}$ we also have that $\mathcal{S}[\![\mathsf{P}]\!] \in \mathfrak{Hp}$. Hence, we can conclude that standard approaches for semantic approximation may work also for hyperproperties, clearly taking into account the imprecision due to the semantics approximation. For instance, suppose $\mathbb{D}\mathbb{EN} = \mathsf{St}$, and suppose to be interested in verifying the hyperproperty

$$\mathtt{PP} \triangleq \{X \in \wp(\mathbb{Z}) \mid \forall \mathfrak{d}_1, \mathfrak{d}_2 \in X \,.\, \mathtt{Par}(\mathfrak{d}_1) = \mathtt{Par}(\mathfrak{d}_2) \Rightarrow \mathtt{Pos}(\mathfrak{d}_1) = \mathtt{Pos}(\mathfrak{d}_2)\}$$

where $\mathtt{Par}$ is the parity while $\mathtt{Pos}$ is the sign of numerical values, respectively.

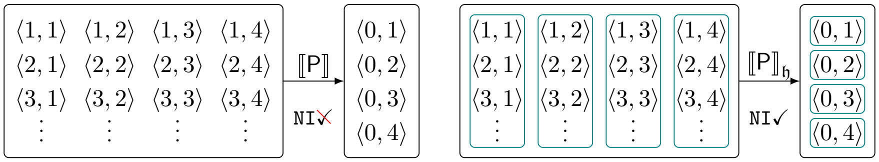
Then, suppose $\mathcal{S}[\![\mathsf{P}_1]\!] = \{1, 3, 4\}$[3], in this case it is clear that $\mathsf{P}_1 \models \mathtt{PP}$, but also the abstract computation of P_1 computing the sign of the set (in this case

[3] For the sake of simplicity, we suppose the programs P_i have only one variable and the state is denoted by the set of its possible values.

positive) would allow to verify the hyperproperty for P_1 (if all computed values have the same sign, $\mathtt{PP}$ is trivially verified). It is anyway clear that, as usual in abstraction, we lose precision since, for example, the program P_2 such that $\mathcal{S}[\mathsf{P}_2] = \{-1, -3, 4\}$ satisfies $\mathtt{PP}$, but the sign abstraction of the semantics would return $\top$, not allowing to verify $\mathtt{PP}$.

Moreover, real problems of precision arise, also for $\mathtt{SSC}^{\mathtt{H}}$, when, due to the approximation, we move verification on domains less expressive than $\mathbb{D}\mathrm{EN}$. For instance, when $\mathbb{D}\mathrm{EN}$ is defined on traces of states (e.g., I/O traces St^2 or partial traces St^*) and the verification method deals with states only. Indeed, if the abstract computation could approximate sets of traces as sets of traces, then still we could reason as before, but sets of traces are usually approximated as a trace of sets, computing the trace of reachable states. This approximation completely loses the trace information necessary for verifying a hyperproperty defined on a trace domain of denotations. In Fig. 2 it is graphically provided the intuition that, by approximating the collecting semantics at the hyperlevel, we obtain a more precise approximation, since we can keep distinctions among reachable states allowing us to verify hyperproperties, with sufficient precision, even in presence of approximation.

Example 3. Consider, for instance, Non-Interference of Example 1, where states in St are denoted as tuples of values, namely a state $[\mathsf{h}/_h, \mathsf{l}/_l]$ is denoted as $\langle h, l\rangle$. Let $\mathsf{P} \triangleq \mathsf{h} := \mathtt{0}\ ;\ \mathsf{l} := 2\mathsf{l}$ and $\Gamma(\mathsf{h}) \triangleq \mathsf{H}, \Gamma(\mathsf{l}) \triangleq \mathsf{L}$. Now consider $\mathfrak{I} \triangleq \{\langle h, l\rangle \mid l \in \{1,2,3,4\}, h \in \mathbb{Z}\}$, then the resulting semantics of the program, starting from $\mathfrak{I}$, is $\{\langle h, l\rangle\langle 0, 2l\rangle \mid l \in \{1,2,3,4\}, h \in \mathbb{Z}\}$. Any over-approximation of this set in $\wp(\mathbb{D}\mathrm{EN})$ allows us to soundly verify $\mathtt{NI}$, e.g., if we abstractly compute, in output, the set $\{\langle h, l\rangle\langle 0, 2l\rangle \mid h, l \in \mathbb{Z}\}$ then we can still soundly verify $\mathtt{NI}$. But, any approximation on traces of sets (i.e., on $\wp(\mathsf{St})^2$), e.g., the trace of sets $\{\langle h, l\rangle \mid l \in \{1,2,3,4\}, h \in \mathbb{Z}\}\{\langle 0, l\rangle \mid l \in \{2,4,6,8\}\}$, losing the I/O relation of traces, becomes useless for $\mathtt{NI}$ verification. In this case, we need to move towards the hyperlevel of semantics, in order to not lose too much information, necessary to verify the hyperproperty. In the example, the possibility to compute the trace of (hyper)sets $\{\{\langle h, l\rangle \mid h \in \mathbb{Z}\} \mid l \in \{1,2,3,4\}\}\{\{\langle 0, l\rangle\} \mid l \in \{2,4,6,8\}\}$ would allow us to verify $\mathtt{NI}$ observing that, independently from a fixed low input and from any high input, the low output is always a constant, being the output of the resulting trace a set of sets of states sharing the same low value. Graphically:



End example.

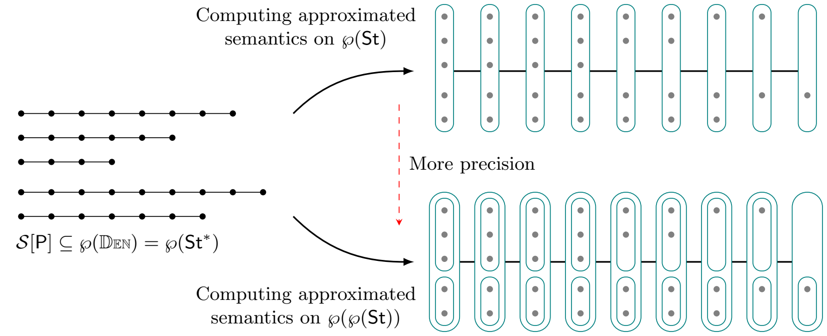

Computing approximated semantics on $\wp(\mathsf{St})$
More precision
$\mathcal{S}[\![\mathsf{P}]\!] \subseteq \wp(\mathbb{D}\mathbb{EN}) = \wp(\mathsf{St}^*)$
Computing approximated semantics on $\wp(\wp(\mathsf{St}))$

Fig. 2. The intuition: Why computing approximation on $\wp(\wp(\mathsf{St}))$ is more precise.

3 Lifting the Collecting Semantics

In this section, we describe how we can move the computation of a semantics into the hyperlevel, in order to be able to approximate the verification of hyperproperties, still keeping as much precision as possible, together with analysis feasibility. We provide the lifting framework parametric on the domain of denotations of the collecting semantics to lift, namely we consider a collecting semantics defined in $\wp(\mathbb{D}\mathbb{EN})$ and we show how to lift it on $\wp(\wp(\mathbb{D}\mathbb{EN}))$. Independently from the domain of the hyperproperty to verify, it is the verification and approximation process that fixes the relation between denotations domains, as shown in Fig. 2. In the figure, the semantics and the hyperproperty to verify are defined on $\wp(\mathsf{St}^*)$, while we lift into the hyperlevel a collecting semantics computed on $\wp(\mathsf{St})$, moving the computation at the hyperlevel, i.e., on $\wp(\wp(\mathsf{St}))$.

As we have observed, in order to verify hyperproperties, we may need to move program semantics into the hyperlevel. In [25], we describe the links between standard and hypersemantics of a transition system. In this section, we show how to *lift* a given collecting semantics[4], defined on sets, in order to obtain a corresponding *collecting hypersemantics*, defined on sets of sets, suitable for hyperproperties verification. In this work, we consider big-step semantics, but the whole framework can be generalized to other types of semantics.

Let $\mathfrak{L}$ be a deterministic imperative language whose set of statements is $Stm_{\mathfrak{L}}$ (single statements without composition). Given the domain of denotations $\mathbb{D}\mathbb{EN}$, the semantic computation is defined by a semantic operator inductively defined on the syntax of $\mathfrak{L}$, i.e., $f^{\mathfrak{L}} \in Stm_{\mathfrak{L}} \times \mathbb{D}\mathbb{EN} \to \mathbb{D}\mathbb{EN}$. Let $\mathsf{P} \in \mathfrak{L}$ be a program written in $\mathfrak{L}$, its concrete (big-step) semantics is a function $\llparenthesis\, \mathsf{P} \,\rrparenthesis \in \mathbb{D}\mathbb{EN} \to \mathbb{D}\mathbb{EN}$ defined compositionally on the statements of P, i.e., it is computed by composing the application of $f^{\mathfrak{L}}$ to the program statements. For instance, let $\mathsf{P} = \mathsf{h} := \mathsf{0}\ ;\ \mathsf{l} := \mathsf{2l}$, the concrete semantics is $\llparenthesis\, \mathsf{P} \,\rrparenthesis \mathfrak{d} = f^{\mathfrak{L}}(\mathsf{l} := \mathsf{2l}, f^{\mathfrak{L}}(\mathsf{h} := \mathsf{0}, \mathfrak{d}))$. In particular, $\llparenthesis\, \mathsf{P} \,\rrparenthesis$ is defined also in terms of the semantics of arithmetic expressions, denoted $\llparenthesis\, \mathsf{a} \,\rrparenthesis \in \mathbb{D}\mathbb{EN} \to \mathbb{Z}$, and of boolean expressions, denoted $\llparenthesis\, \mathsf{b} \,\rrparenthesis \in \mathbb{D}\mathbb{EN} \to \mathbb{B}$.

[4] Namely a semantics function, defined on a program P syntax, computing $\mathcal{S}[\![\mathsf{P}]\!]$.

3.1 Lifting the (collecting) interpreter

The collecting semantics $[\![\mathsf{P}]\!] \in \wp(\mathbb{D}\mathrm{en}) \rightarrow \wp(\mathbb{D}\mathrm{en})$ is the additive lift (i.e., the set of the direct images of the elements in input) to sets of denotations, namely $[\![\mathsf{P}]\!]X = \{\langle\!|\,\mathsf{P}\,|\!\rangle\mathfrak{d} \mid \mathfrak{d} \in X\}$. As far as expression semantics is concerned, for boolean expressions $[\![\mathsf{b}]\!] \in \wp(\mathbb{D}\mathrm{en}) \rightarrow \wp(\mathbb{D}\mathrm{en})$ is a filtering function, namely $[\![\mathsf{b}]\!]X \triangleq \{\mathfrak{d} \in X \mid \langle\!|\,\mathsf{b}\,|\!\rangle\mathfrak{d} = \mathtt{tt}\}$, while for arithmetic expressions it is the additive lift of the concrete semantics. The collecting semantics is computed by composing a new operator $F^{\mathfrak{L}} \in \mathit{Stm}_{\mathfrak{L}} \times \wp(\mathbb{D}\mathrm{en}) \rightarrow \wp(\mathbb{D}\mathrm{en})$, which is the additive lift of $f^{\mathfrak{L}}$. For example, the semantics for assignments is $[\![\mathsf{x} := \mathsf{a}]\!]X = F^{\mathfrak{L}}(\mathsf{x} := \mathsf{a}, X) \triangleq \{f^{\mathfrak{L}}(\mathsf{x} := \mathsf{a}, \mathfrak{d}) \mid \mathfrak{d} \in X\}$. The while statement operator is defined as $F^{\mathfrak{L}}(\mathsf{while}\ \mathsf{b}\ \{\,\mathsf{P}\,\}, X) \triangleq [\![\neg\mathsf{b}]\!](\mathit{lfp}^{\subseteq}_{\varnothing}\,\mathcal{W})$, where $\mathcal{W} \triangleq \lambda T\,.\,X \cup [\![\mathsf{P}]\!][\![\mathsf{b}]\!]T$. It can be shown that $\mathcal{W}$ is a monotone function over the complete lattice $\langle\wp(\mathbb{D}\mathrm{en}), \subseteq, \cup, \cap, \mathbb{D}\mathrm{en}, \varnothing\rangle$ hence its least fixpoint exists and it can be computed as $\bigcup_{n\geq 0}\mathcal{W}^n(\varnothing)$, with $\mathcal{W}^0 \triangleq \lambda X\,.\,\varnothing$ and $\mathcal{W}^{n+1} \triangleq \lambda X\,.\,\mathcal{W}\circ\mathcal{W}^n(X)$. In this case, this least fixpoint is precisely the additive lift of $f^{\mathfrak{L}}$, namely $F^{\mathfrak{L}}(\mathsf{while}\ \mathsf{b}\ \{\,\mathsf{P}\,\}, X) = \{f^{\mathfrak{L}}(\mathsf{while}\ \mathsf{b}\ \{\,\mathsf{P}\,\}, \mathfrak{d}) \mid \mathfrak{d} \in X\}$. Note that, if $\mathfrak{I} \subseteq \mathbb{D}\mathrm{en}$ is the set of *all* possible inputs of the program, the collecting semantics $[\![\mathsf{P}]\!]$ from $\mathfrak{I}$ computes the strongest program property $\mathcal{S}[\mathsf{P}] \in \wp(\mathbb{D}\mathrm{en})$, i.e., $\mathcal{S}[\mathsf{P}] = [\![\mathsf{P}]\!]\mathfrak{I}$.

At this point, we have to move semantics towards the hyperlevel, namely on $\wp(\wp(\mathbb{D}\mathrm{en}))$, since, when we are interested in hyperproperties, we may need to define a *collecting hypersemantics* $[\![\mathsf{P}]\!]_{\mathfrak{h}} \in \wp(\wp(\mathbb{D}\mathrm{en})) \rightarrow \wp(\wp(\mathbb{D}\mathrm{en}))$. In this case, we need to lift the semantic operator $F^{\mathfrak{L}}$, and we can show several ways for doing it. Suppose to have the filtering function $[\![\mathsf{b}]\!]_{\mathfrak{h}} \in \wp(\wp(\mathbb{D}\mathrm{en})) \rightarrow \wp(\wp(\mathbb{D}\mathrm{en}))$ for boolean expressions, defined as $[\![\mathsf{b}]\!]_{\mathfrak{h}}\mathcal{X} \triangleq \{[\![\mathsf{b}]\!]X \mid X \in \mathcal{X}\}$. The definition of the collecting hypersemantics is just the additive lift (to sets of sets) of $F^{\mathfrak{L}}$ for every statement, except for the while case. Indeed, we can observe that, at hyperlevel, the semantic operator $F^{\mathfrak{L}}_{\mathfrak{h}}$ for the while statements does not coincide with the additive lift of $F^{\mathfrak{L}}$, which would be $F^{\mathfrak{L}}_{\mathfrak{h}}(\mathsf{while}\ \mathsf{b}\ \{\,\mathsf{P}\,\}, \mathcal{X}) \triangleq [\![\neg\mathsf{b}]\!]_{\mathfrak{h}}(\mathit{lfp}^{\subseteq}_{\varnothing}\,\mathcal{W}_{\mathfrak{h}})$ with $\mathcal{W}_{\mathfrak{h}} \triangleq \lambda\mathcal{T}\,.\,\mathcal{X} \cup [\![\mathsf{P}]\!]_{\mathfrak{h}}[\![\mathsf{b}]\!]_{\mathfrak{h}}\mathcal{T}$. Unfortunately, this semantics is not sound being such that $[\![\mathsf{P}]\!]X \notin [\![\mathsf{P}]\!]_{\mathfrak{h}}\{X\}$. This is a problem, since when $[\![\mathsf{P}]\!]\mathfrak{I} \notin [\![\mathsf{P}]\!]_{\mathfrak{h}}\{\mathfrak{I}\}$, from $[\![\mathsf{P}]\!]_{\mathfrak{h}}\{\mathfrak{I}\} \subseteq \mathfrak{H}\mathfrak{p}$ we cannot infer anything about the property validation.

Example 4. Let $\mathbb{D}\mathrm{en} = \mathsf{Var} \rightarrow \mathbb{Z}$ and $\mathsf{P} = \mathsf{while}\ (\mathsf{x} < 4)\ \{\,\mathsf{x} := \mathsf{x} + 1\,\}$. Since P has only one variable, we simplify the notation by denoting $[\mathsf{x}/_{v}]$ just by v and the set of functions $\{[\mathsf{x}/_{v_1}], \ldots [\mathsf{x}/_{v_n}]\}$ by $\{v_1, \ldots v_n\}$. The collecting semantics, from $\mathfrak{I} = \{2, 5\}$, is $[\![\mathsf{P}]\!]\{2, 5\} = \{4, 5\}$, computed as $\{2, 5\} \xrightarrow{\mathcal{W}} \{4, 5\}$ where

$$\mathcal{W}^0 = \varnothing;\ \mathcal{W}^1 = \{2, 5\};\ \mathcal{W}^2 = \{2, 3, 5\};\ \mathcal{W}^3 = \{2, 3, 4, 5\}$$

The trivial additive lift of the while collecting semantics would be $[\![\mathsf{P}]\!]_{\mathfrak{h}}\{\{2, 5\}\} = \{\varnothing, \{4\}, \{5\}\}$, computed as $\{\{2, 5\}\} \xrightarrow{\mathcal{W}_{\mathfrak{h}}} \{\varnothing, \{4\}, \{5\}\}$ where

$$\mathcal{W}_{\mathfrak{h}}{}^0 = \varnothing;\ \mathcal{W}_{\mathfrak{h}}{}^1 = \{\{2, 5\}\};\ \mathcal{W}_{\mathfrak{h}}{}^2 = \{\{3\}, \{2, 5\}\};\ \mathcal{W}_{\mathfrak{h}}{}^3 = \{\{3\}, \{4\}, \{2, 5\}\};$$
$$\mathcal{W}_{\mathfrak{h}}{}^4 = \{\varnothing, \{3\}, \{4\}, \{2, 5\}\}$$

From the iterates of $\mathcal{W}^i$ and $\mathcal{W}\mathfrak{h}^i$ we can observe the monotonicity (and the extensivity) of $\mathcal{W}$ and $\mathcal{W}\mathfrak{h}$, but the hypersemantics is not sound, because $[\![\mathsf{P}]\!]\{2,5\} = \{4,5\} \notin \{\varnothing, \{4\}, \{5\}\} = [\![\mathsf{P}]\!]_{\mathfrak{h}}\{\{2,5\}\}$. *End example.*

In order to lift the while semantics, we propose the following three possibilities. We define the collecting hypersemantics operator for while statements as $F^{\mathscr{E}}_{\mathfrak{h}}(\textsf{while}\ \mathsf{b}\ \{\ \mathsf{P}\ \}, \mathcal{X}) \triangleq [\![\neg\mathsf{b}]\!]_{\mathfrak{h}}(\mathit{lfp}^{\subseteq}_{\varnothing}\, \mathcal{W}\mathfrak{h})$ where:

1. (*Bcc lift*) $\quad \mathcal{W}\mathfrak{h} \triangleq \lambda\mathcal{T}\,.\,\wp(\bigcup\mathcal{X} \cup [\![\mathsf{P}]\!][\![\mathsf{b}]\!]\bigcup\mathcal{T})$
2. (*Inner lift*) $\quad \mathcal{W}\mathfrak{h} \triangleq \lambda\mathcal{T}\,.\,\{\varnothing\} \cup (\mathcal{X} \Cup [\![\mathsf{P}]\!]_{\mathfrak{h}}[\![\mathsf{b}]\!]_{\mathfrak{h}}\mathcal{T})$
3. (*Mixed lift*) $\quad \mathcal{W}\mathfrak{h} \triangleq \lambda\mathcal{T}\,.\,\mathcal{X} \cup \{[\![\mathsf{P}]\!][\![\mathsf{b}]\!]T \cup [\![\neg\mathsf{b}]\!]T \mid T \in \mathcal{T}\}$

The *Bcc lift* defines the collecting hypersemantics as the best complete concretization [25] of the while semantics. The *Inner lift* combines by union, at each step of computation, all the possible results. In particular, the binary operator $\Cup \in \wp(\wp(\mathbb{D}\mathbb{EN})) \times \wp(\wp(\mathbb{D}\mathbb{EN})) \to \wp(\wp(\mathbb{D}\mathbb{EN}))$, defined as $\mathcal{X} \Cup \mathcal{Y} \triangleq \{X \cup Y \mid X \in \mathcal{X} \wedge Y \in \mathcal{Y}\}$, is a slight modification of $\uplus$, introduced in [25] and it is an instance of the construction presented in [14] (Page 4, example 1). Moreover, the resulting semantics corresponds to the one proposed in [20] for analyzing analyses. Finally, the *Mixed lift* is the instantiation of the hypercollecting semantics of [4] to a generic trace denotations domain $\mathbb{D}\mathbb{EN}$. Each while operator $\mathcal{W}\mathfrak{h}$ is a monotone function over the complete lattice $\langle \wp(\wp(\mathbb{D}\mathbb{EN})), \subseteq, \cup, \cap, \wp(\mathbb{D}\mathbb{EN}), \varnothing\rangle$, hence its least fixpoint exists and it can be computed as shown before.

Unfortunately, none of the previous definitions computes the additive lift of $F^{\mathscr{E}}$, namely $\{F^{\mathscr{E}}(\textsf{while}\ \mathsf{b}\ \{\ \mathsf{P}\ \}, X) \mid X \in \mathcal{X}\} \neq F^{\mathscr{E}}_{\mathfrak{h}}(\textsf{while}\ \mathsf{b}\ \{\ \mathsf{P}\ \}, \mathcal{X})$, as we can observe in the next example.

Example 5. Consider P of Example 4. The *Bcc lift* collecting hypersemantics is $[\![\mathsf{P}]\!]_{\mathfrak{h}}\{\{2,5\}\} = \wp(\{4,5\})$, computed as $\{\{2,5\}\} \xrightarrow{\mathcal{W}\mathfrak{h}} \wp(\{4,5\})\}$ where

$$\mathcal{W}\mathfrak{h}^0 = \ \varnothing;\ \mathcal{W}\mathfrak{h}^1 = \wp(\{2,5\});\ \mathcal{W}\mathfrak{h}^2 = \wp(\{2,3,5\});\ \mathcal{W}\mathfrak{h}^3 = \wp(\{2,3,4,5\})$$

The *Inner lift* collecting hypersemantics is $[\![\mathsf{P}]\!]_{\mathfrak{h}}\{\{2,5\}\} = \{\varnothing, \{5\}, \{4,5\}\}$, computed as $\{\{2,5\}\} \xrightarrow{\mathcal{W}\mathfrak{h}} \{\varnothing, \{5\}, \{4,5\}\}$ where

$$\begin{aligned}&\mathcal{W}\mathfrak{h}^0 = \varnothing;\ \mathcal{W}\mathfrak{h}^1 = \{\varnothing, \{2,5\}\};\ \mathcal{W}\mathfrak{h}^2 = \{\varnothing, \{2,5\}, \{2,3,5\}\};\\ &\mathcal{W}\mathfrak{h}^3 = \{\varnothing, \{2,5\}, \{2,3,5\}, \{2,3,4,5\}\}\end{aligned}$$

The *Mixed lift* collecting hypersemantics is $[\![\mathsf{P}]\!]_{\mathfrak{h}}\{\{2,5\}\} = \{\{5\}, \{4,5\}\}$, computed as $\{\{2,5\}\} \xrightarrow{\mathcal{W}\mathfrak{h}} \{\{5\}, \{4,5\}\}$ where

$$\mathcal{W}\mathfrak{h}^0 = \varnothing;\ \mathcal{W}\mathfrak{h}^1 = \{\{2,5\}\};\ \mathcal{W}\mathfrak{h}^2 = \{\{2,5\}, \{3,5\}\};\ \mathcal{W}\mathfrak{h}^3 = \{\{2,5\}, \{3,5\}, \{4,5\}\}$$

From the iterates $\mathcal{W}\mathfrak{h}^i$ we observe the monotonicity (extensivity) of $\mathcal{W}\mathfrak{h}$. All the semantics are sound, because $[\![\mathsf{P}]\!]\{2,5\} \in [\![\mathsf{P}]\!]_{\mathfrak{h}}\{\{2,5\}\}$. *End example.*

3.2 Soundness and completeness issues

Let $\llbracket \mathsf{P} \rrbracket^{\mathfrak{b}}_{\mathfrak{h}}$, $\llbracket \mathsf{P} \rrbracket^{\mathfrak{i}}_{\mathfrak{h}}$ and $\llbracket \mathsf{P} \rrbracket^{\mathfrak{m}}_{\mathfrak{h}}$ be the collecting hypersemantics defined in terms of the *Bcc*, *Inner* and *Mixed* lifts, respectively, for the while case of $F^{\mathfrak{L}}_{\mathfrak{h}}$, and defined as the additive lift to $\wp(\wp(\mathbb{D}\mathbb{EN}))$ of $F^{\mathfrak{L}}$ for all the other statements. Then, all these collecting hypersemantics are sound.

Theorem 1 (Soundness). *For every $X \in \wp(\mathbb{D}\mathbb{EN})$ we have*

$$\llbracket \mathsf{P} \rrbracket X \in \llbracket \mathsf{P} \rrbracket^{\mathfrak{b}}_{\mathfrak{h}}\{X\} \quad \textit{and} \quad \llbracket \mathsf{P} \rrbracket X \in \llbracket \mathsf{P} \rrbracket^{\mathfrak{i}}_{\mathfrak{h}}\{X\} \quad \textit{and} \quad \llbracket \mathsf{P} \rrbracket X \in \llbracket \mathsf{P} \rrbracket^{\mathfrak{m}}_{\mathfrak{h}}\{X\}$$

This results 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 $\wp(\wp(\mathbb{D}\mathbb{EN}))$. This is somewhat new: Usually the source of incompleteness is the abstraction process (of an abstract semantics), not the collecting semantics itself. Luckily, for subset-closed hyperproperties this is not a real concern. In fact when $\mathfrak{Hp} \in \mathtt{SSC}^{\mathtt{H}}$, we have that $\mathsf{P} \models \mathfrak{Hp}$ iff $\wp(\llbracket \mathsf{P} \rrbracket \mathfrak{I}) \subseteq \mathfrak{Hp}$. Furthermore, the three collecting hypersemantics introduced above, are related as follows.

Proposition 2. $\forall \mathcal{X} \in \wp(\wp(\mathbb{D}\mathbb{EN}))$*:* $\llbracket \mathsf{P} \rrbracket^{\mathfrak{m}}_{\mathfrak{h}} \mathcal{X} \subseteq \llbracket \mathsf{P} \rrbracket^{\mathfrak{b}}_{\mathfrak{h}} \mathcal{X}$ *and* $\llbracket \mathsf{P} \rrbracket^{\mathfrak{i}}_{\mathfrak{h}} \mathcal{X} \subseteq \llbracket \mathsf{P} \rrbracket^{\mathfrak{b}}_{\mathfrak{h}} \mathcal{X}$*.*

Hence we can state that all the proposed collecting hypersemantics are complete verification methods for bounded subset-closed hyperproperties.

Theorem 2 (Completeness). *Let $\mathfrak{Hp} \in \mathtt{SSC}^{\mathtt{H}}_{k}$ (for some $k \in \mathbb{N}$), then:*

$$\mathsf{P} \models \mathfrak{Hp} \;\Leftrightarrow\; \llbracket \mathsf{P} \rrbracket^{\mathfrak{b}}_{\mathfrak{h}}\{\mathfrak{I}\} \subseteq \mathfrak{Hp} \;\Leftrightarrow\; \llbracket \mathsf{P} \rrbracket^{\mathfrak{i}}_{\mathfrak{h}}\{\mathfrak{I}\} \subseteq \mathfrak{Hp} \;\Leftrightarrow\; \llbracket \mathsf{P} \rrbracket^{\mathfrak{m}}_{\mathfrak{h}}\{\mathfrak{I}\} \subseteq \mathfrak{Hp}$$

The theorem follows from the fact that all three semantics, computed from $\mathfrak{I}$, are contained in $\wp(\llbracket \mathsf{P} \rrbracket \mathfrak{I})$. So, even if the collecting hypersemantics inserts spurious information, this information does not lower the precision of the analysis, when we deal with bounded subset-closed hyperproperties. Note that the Thm. 2 also holds with $k = \omega$, i.e., it also holds for unbounded subset-closed hyperproperties.

4 Lifting Abstract Domains

Once we have lifted the semantics, in order to perform verification we need to compute the semantics on an abstract domain[5], namely we have to compute an abstract semantics. In the classic framework of abstract interpretation [12, 13] we compute an over-approximation $O \supseteq \llbracket \mathsf{P} \rrbracket \mathfrak{I}$ of a program semantics, allowing us to soundly verify 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. Let P be a program, $\mathcal{A}$ an abstract domain of $\wp(\mathbb{D}\mathbb{EN})$, forming the Galois connection $(\langle \wp(\mathbb{D}\mathbb{EN}), \subseteq \rangle, \alpha, \gamma, \langle \mathcal{A}, \preccurlyeq \rangle)$, $\mathfrak{P}$ a trace property in $\wp(\mathbb{D}\mathbb{EN})$ and $\llbracket \mathsf{P} \rrbracket^{\mathcal{A}}$ an abstract interpretation of $\llbracket \mathsf{P} \rrbracket$ on $\mathcal{A}$, i.e., $\llbracket \mathsf{P} \rrbracket X \subseteq \gamma \circ \llbracket \mathsf{P} \rrbracket^{\mathcal{A}} \circ \alpha(X)$.

[5] $\mathcal{A}$ is an abstract domain of $\mathcal{C}$ if there exists a Galois connection $(\langle \mathcal{C}, \preccurlyeq \rangle, \alpha, \gamma, \langle \mathcal{A}, \preceq \rangle)$, where α, γ are monotone maps such that: $\forall c \in \mathcal{C}, a \in \mathcal{A} \,.\, \alpha(c) \preceq a \Leftrightarrow c \preccurlyeq \gamma(a)$.

Then $\gamma \circ [\![\mathsf{P}]\!]^{\mathcal{A}} \circ \alpha(\mathfrak{I}) \subseteq \mathfrak{P}$ implies $\mathsf{P} \models \mathfrak{P}$. Similarly, an over-approximation $\mathcal{O} \supseteq [\![\mathsf{P}]\!]_{\mathfrak{h}}\{\mathfrak{I}\}$ leads to a sound verification mechanism for hyperproperties. Let $\mathcal{A}\mathfrak{h}$ be an abstract domain of $\wp(\wp(\mathbb{D}\mathbb{EN}))$, forming the Galois connection $(\langle \wp(\wp(\mathbb{D}\mathbb{EN})), \subseteq \rangle, \alpha_{\mathfrak{h}}, \gamma_{\mathfrak{h}}, \langle \mathcal{A}\mathfrak{h}, \preccurlyeq_{\mathfrak{h}} \rangle)$, $\mathfrak{Hp} \in \wp(\wp(\mathbb{D}\mathbb{EN}))$ an hyperproperty, $[\![\mathsf{P}]\!]_{\mathfrak{h}}$ a sound collecting hypersemantics, i.e., $[\![\mathsf{P}]\!]\mathfrak{I} \in [\![\mathsf{P}]\!]_{\mathfrak{h}}\{\mathfrak{I}\}$, and $[\![\mathsf{P}]\!]_{\mathfrak{h}}^{\mathcal{A}\mathfrak{h}}$ an abstract interpretation of $[\![\mathsf{P}]\!]_{\mathfrak{h}}$ on $\mathcal{A}\mathfrak{h}$, i.e., $[\![\mathsf{P}]\!]_{\mathfrak{h}}\mathcal{X} \subseteq \gamma_{\mathfrak{h}} \circ [\![\mathsf{P}]\!]_{\mathfrak{h}}^{\mathcal{A}\mathfrak{h}} \circ \alpha_{\mathfrak{h}}(\mathcal{X})$. Then:

$$\gamma_{\mathfrak{h}} \circ [\![\mathsf{P}]\!]_{\mathfrak{h}}^{\mathcal{A}\mathfrak{h}} \circ \alpha_{\mathfrak{h}}(\{\mathfrak{I}\}) \subseteq \mathfrak{Hp} \;\text{ implies }\; \mathsf{P} \models \mathfrak{Hp}$$

Hence, at this point we wonder how we can define/lift abstract domains at the hyperlevel, i.e., on sets of sets, in order to approximate hypersemantics, i.e., semantics lifted to the hyperlevel.

4.1 The Compositional Nature of Hyper Abstract Domains.

An 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 concrete semantics is on sets. Let us consider Non-Interference (`NI`) as running example, for providing the intuition beyond these concepts. `NI` requires that, for each set of computations agreeing on the the same low input, the low output is constant.

The *inner abstraction* approximates sets of denotations in $\mathbb{D}\mathbb{EN}$, namely it says which information about program executions should be observed. In `NI`, for each set of computations we are interested in the constant analysis on low variables, i.e., each set of computations (starting from states agreeing on the low variables) should be contained in a set of the form $\mathtt{C}_l \triangleq \{\langle h, l \rangle \mid h \in \mathbb{Z}\}$, $l \in \mathbb{Z}$.

The *outer abstraction* approximates sets of sets of denotations, namely it says which information about programs 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 constants in the 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 standard level $\wp(\mathbb{D}\mathbb{EN})$, we need to lift the inner abstraction to $\wp(\wp(\mathbb{D}\mathbb{EN}))$. In this case, the *lifting function* just leverages the domain at the 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* $\mathcal{A}$ is given by the Galois connection

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

The *lifting transformer* $\mathcal{L} \in (\wp(\mathbb{D}\mathbb{EN}) \rightarrow \mathcal{A}) \rightarrow (\wp(\wp(\mathbb{D}\mathbb{EN})) \rightarrow \wp(\mathcal{A}))$ is the transformer addively lifting functions, namely $\mathcal{L} \triangleq \lambda f \,.\, \lambda \mathcal{X} \,.\, \{f(X) \mid X \in \mathcal{X}\}$ [11]. Let us consider the transformer $\mathcal{G} \in (\wp(\mathbb{D}\mathbb{EN}) \rightarrow \mathcal{A}) \rightarrow (\wp(\mathcal{A}) \rightarrow \wp(\wp(\mathbb{D}\mathbb{EN})))$

[11] defined as $\mathcal{G} \triangleq \lambda f \,.\, \lambda Y \,.\, \{X \mid f(X) \in Y\}$. Due to elementwise set abstraction, we have that $\mathcal{L}(\alpha_\mathsf{i})$ and $\mathcal{G}(\alpha_\mathsf{i})$ form a Galois connection [11], in particular we have

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

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

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

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

$$\langle \wp(\wp(\mathbb{D}\mathbb{EN})), \subseteq\rangle \underset{\alpha_\mathsf{o} \circ \mathcal{L}(\alpha_\mathsf{i})}{\overset{\mathcal{G}(\alpha_\mathsf{i}) \circ \gamma_\mathsf{o}}{\leftrightarrows}} \langle \mathcal{A}_\mathfrak{h}, \preccurlyeq_\mathfrak{h}\rangle$$

Note that, it is not mandatory, for the inner abstraction $\mathcal{A}$, to form a Galois connection. Indeed, in order to apply the lifting transformer, the abstraction function α_i may also fail additivity [11]. Note that, the abstract domains defined in [4] are instances of the pattern proposed here. For instance, cardinality abstraction $\mathtt{crdval} \in \wp(\mathbb{Z}) \to [0, \infty]$ (which is not additive) 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 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 examples of hyper abstract domains obtained starting from initial known abstractions on sets.

4.2 Dealing with Constants Propagation.

Suppose to define an hyperanalysis on the concrete domain $\wp(\wp(\mathbb{Z}))$, and to be interested in constants propagation at the hyperlevel, namely we aim at verifying whether all the sets of computations provide constant results. This corresponds intuitively to an inner abstraction which is the hyperlevel constant propagation (lifted as shown before), and an outer abstraction retrieving information about the constant analysis at standard level. The standard domain of constants $\mathtt{C} \triangleq \mathbb{Z} \cup \{\bot, \top\}$ is defined by the Galois insertion[6] $(\langle \wp(\mathbb{Z}), \subseteq\rangle, \alpha_\mathtt{c}, \gamma_\mathtt{c}, \langle \mathtt{C}, \preceq\rangle)$ 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

$$\alpha_\mathtt{c} \triangleq \lambda X \,.\, \begin{cases} \bot & \text{if } X = \varnothing \\ n & \text{if } X = \{n\} \\ \top & \text{otherwise} \end{cases} \qquad \gamma_\mathtt{c} \triangleq \lambda \mathtt{c} \,.\, \begin{cases} \varnothing & \text{if } \mathtt{c} = \bot \\ \{n\} & \text{if } \mathtt{c} = n \\ \mathbb{Z} & \text{otherwise} \end{cases}$$

[6] It is a Galois connection with surjective abstraction function.

In order to get an abstract domain on sets of sets we rely on the lifting transformer, obtaining the following Galois insertion

$$\langle \wp(\wp(\mathbb{Z})), \subseteq \rangle \underset{\mathcal{L}(\alpha_{\mathsf{c}})}{\overset{\mathcal{G}(\alpha_{\mathsf{c}})}{\leftrightarrows}} \langle \wp(\mathtt{C}), \subseteq \rangle$$

At this point, to look for constant invariants at the hyperlevel, namely in the outer abstraction, means to check whether all the collected sets of values are constants. Hence, we need to retrieve information about what there is inside the analysis at standard level. This is obtained by using the Galois insertion

$$\langle \wp(\mathtt{C}), \subseteq \rangle \underset{\alpha_{\mathtt{cc}}}{\overset{\gamma_{\mathtt{cc}}}{\leftrightarrows}} \langle \wp(\mathbb{Z}) \cup \{\mathtt{C}\}, \subseteq \rangle \text{ where } \alpha_{\mathtt{cc}}(X) \triangleq \begin{cases} X & \text{if } X \subseteq \mathbb{Z} \\ \mathtt{C} & \text{otherwise} \end{cases} \quad \gamma_{\mathtt{cc}} \triangleq id$$

Obtaining, by composition, the insertion

$$\langle \wp(\wp(\mathbb{Z})), \subseteq \rangle \underset{\alpha_{\mathtt{cc}} \circ \mathcal{L}(\alpha_{\mathsf{c}})}{\overset{\mathcal{G}(\alpha_{\mathsf{c}}) \circ \gamma_{\mathtt{cc}}}{\leftrightarrows}} \langle \wp(\mathbb{Z}) \cup \{\mathtt{C}\}, \subseteq \rangle) \tag{2}$$

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 hyper abstract domain *hyperlevel (abstract) constants* of an inner abstraction.

Hyperlevel (Abstract) Constants. 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., $\mathrm{Atm}^{\mathcal{A}} \triangleq \{a \in \mathcal{A} \mid \forall a' \in \mathcal{A} \,.\, a' \preccurlyeq a \Rightarrow (a' = \bot^{\mathcal{A}} \vee a' = a)\}$. Suppose $\mathcal{A}$ is *partitioning*[7] [22, 29], which in particular implies that $\mathrm{Atm}^{\mathcal{A}}$ induces, by means of α, a partition of $\mathcal{C}$, namely for each element $c \in \mathcal{C}$ we have that $\alpha(c) \in \mathrm{Atm}^{\mathcal{A}}$. For instance, consider the abstract domain $\mathtt{Pos} \triangleq \{\varnothing, \mathbb{Z}_{<0}, \{0\}, \mathbb{Z}_{>0}, \mathbb{Z}_{\geq 0}, \mathbb{Z}_{\leq 0}, \mathbb{Z}_{\neq 0}, \mathbb{Z}\}$[8] $\subseteq \wp(\mathbb{Z})$. The set of its atoms is $\mathrm{Atm}^{\mathtt{Pos}} = \{\mathbb{Z}_{<0}, \{0\}, \mathbb{Z}_{>0}\}$. In order to perform hyperlevel constants on $\mathcal{A}$ we consider the set of its atoms, which precisely identify the properties of concrete values observed in $\mathcal{A}$ (in $\mathtt{Pos}$ 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. At this point, we can define the hyperlevel (abstract) constants domain for $\mathcal{A}$ as $\mathcal{A}_{\mathfrak{hc}} \triangleq \wp(\mathrm{Atm}^{\mathcal{A}}) \cup \{\mathcal{A}\}$, forming the following insertion:

$$\langle \wp(\mathcal{A}), \subseteq \rangle \underset{\alpha_{\mathfrak{hc}}}{\overset{\gamma_{\mathfrak{hc}}}{\leftrightarrows}} \langle \mathcal{A}_{\mathfrak{hc}}, \subseteq \rangle \text{ where } \quad \alpha_{\mathfrak{hc}}(X) \triangleq \begin{cases} X & \text{if } X \subseteq \mathrm{Atm}^{\mathcal{A}} \\ \mathcal{A} & \text{otherwise} \end{cases} \quad \gamma_{\mathfrak{hc}} \triangleq id$$

[7] We recall that any abstract domain can be made partitioning [22].

[8] Where $\mathbb{Z}_{<0} \triangleq \{n \in \mathbb{Z} \mid n < 0\}$ and the others are similarly defined.

Then, applying the lifting transformer and composing, we have

$$\langle \wp(\wp(\mathcal{C})), \subseteq \rangle \underset{\mathcal{L}(\alpha)}{\overset{\mathcal{G}(\alpha)}{\leftrightarrows}} \langle \wp(\mathcal{A}), \subseteq \rangle \qquad \langle \wp(\wp(\mathcal{C})), \subseteq \rangle \underset{\alpha_{\mathfrak{hc}} \circ \mathcal{L}(\alpha)}{\overset{\mathcal{G}(\alpha) \circ \gamma_{\mathfrak{hc}}}{\leftrightarrows}} \langle \mathcal{A}_{\mathfrak{hc}}, \subseteq \rangle \tag{3}$$

For instance, if $\mathcal{C} = \mathbb{Z}$ and $\mathcal{A} = \texttt{Pos}$ then $\texttt{Pos}_{\mathfrak{hc}} \triangleq \wp(\{\varnothing, \mathbb{Z}_{<0}, \{0\}, \mathbb{Z}_{>0}\}) \cup \{\texttt{Pos}\}$ is the hyperdomain, abstraction of $\wp(\wp(\mathbb{Z}))$, for hyperlevel (abstract) Pos-constants.

4.3 Dealing with Intervals.

Suppose now to be interested in a hyper intervals analysis. The classic abstract domain of intervals is defined over numerical values, but the interval construction can be easily generalized [13]. Given a complete lattice $\langle \mathcal{C}, \leqslant, \vee, \wedge, \top, \bot \rangle$, we can define its interval domain as:

$$\mathcal{I} = \{[a, b] \mid a \in \mathcal{C} \smallsetminus \{\top\}, b \in \mathcal{C} \smallsetminus \{\bot\}, a \leqslant b\} \cup \{\bot_\imath\}$$

We have that $\langle \mathcal{I}, \sqsubseteq, \sqcup, \sqcap, [\bot, \top], \bot_\imath \rangle$ is a complete lattice where: $\forall I \in \mathcal{I} \,.\, \bot_\imath \sqsubseteq I \sqsubseteq [\bot, \top]$ and $[a, b] \sqsubseteq [c, d]$ iff $c \leqslant a$ and $b \leqslant d$; $[a, b] \sqcup [c, d] \triangleq [a \wedge c, b \vee d]$; $[a, b] \sqcap [c, d] \triangleq [a \vee c, b \wedge d]$ if $a \vee c \leqslant b \wedge d$ and $[a, b] \sqcap [c, d] \triangleq \bot^\imath$ if $a \vee c \nleqslant b \wedge d$. An instance of this pattern is the classic domain of intervals over integers, where the initial domain is the lattice $\langle \mathbb{Z} \cup \{-\infty, +\infty\}, \leq, \max, \min, +\infty, -\infty \rangle$ [13].

The corresponding Galois connection between (the powerset of) the concrete domain $\mathcal{C}$ and its intervals domain is

$$\langle \wp(\mathcal{C}), \subseteq \rangle \underset{\alpha_\imath}{\overset{\gamma_\imath}{\leftrightarrows}} \langle \mathcal{I}, \sqsubseteq \rangle \text{ where } \quad \alpha_\imath(X) \triangleq [\bigwedge X, \bigvee X] \quad \gamma_\imath([a, b]) \triangleq \{c \in \mathcal{C} \mid a \leqslant c \leqslant b\}$$

We can use this 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 a 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 is Pos, then we would characterize the sign properties of interval bounds.

5 Verifying Bounded Subset-Closed Hyperproperties

As we have seen in Sect. 2, for bounded subset-closed hyperproperties the verification process is simplified. Instead of checking the hyperproperty for the set of all inputs $\mathfrak{I}$, or for all its subsets, it is sufficient to check the hyperproperty for a set of *finite* subsets of $\mathfrak{I}$. Namely, if $\mathfrak{Hp} \in \mathtt{SSC}^{\mathtt{H}}_{k}$, we need to check the sets in $\mathfrak{I}^{|k} \triangleq \{X \subseteq \mathfrak{I} \mid |X| = k\}$. Then with a sound collecting hypersemantics $[\![\mathsf{P}]\!]_{\mathfrak{h}}$ (Sect. 3), we can verify the hyperproperty just approximating $[\![\mathsf{P}]\!]_{\mathfrak{h}} \mathfrak{I}^{|k}$.

Theorem 3. *Given* $\mathfrak{Hp} \in \mathtt{SSC}^{\mathtt{H}}_{k}$*, we have that* $[\![\mathsf{P}]\!]_{\mathfrak{h}} \mathfrak{I}^{|k} \subseteq \mathfrak{Hp}$ *iff* $\mathsf{P} \models \mathfrak{Hp}$.

Proof. By soundness and completeness (for $\mathtt{SSC}^{\mathtt{H}}$) of the collecting hypersemantics, stated in Sec. 3, we have that $\{[\![P]\!]X \mid X \in \mathfrak{I}^{|k}\} \subseteq [\![\mathsf{P}]\!]_{\mathfrak{h}}\mathfrak{I}^{|k}$. Then, recalling that we are in a deterministic setting, we have that $\{[\![P]\!]X \mid X \in \mathfrak{I}^{|k}\} = \{X \subseteq [\![\mathsf{P}]\!]\mathfrak{I} \mid |X| = k\}$. Then, the theorem follows from the results of Sect 2. □

Th. 3 allows us to simplify the design of hyperanalyses for bounded subset-closed hyperproperties. It justifies also the methodology used in [4] in order to verify information flow. In fact, despite their analysis starts from $\{\mathfrak{I}\}$, the (abstract) semantics indeed decomposes $\mathfrak{I}$ in all its subsets, in order to apply approximations at the level of sets of sets. Th. 3 confirms the correctness of the approach used in [4] and states that the "decomposition" can be made explicit, splitting the input set from which we start the hyperanalysis.

5.1 Non-Interference

Information flows control is one of the primary motivations that has led researchers to develop a theory about hyperproperties. A well-known information flow property is Non-Interference [10, 21], introduced in Example 1. As we have seen in the example, $\mathtt{NI}$ is defined over I/O traces, i.e., $\mathbb{DEN} \triangleq \mathsf{St} \times \mathsf{St} = (\mathsf{Var} \rightarrow \mathbb{Z}) \times (\mathsf{Var} \rightarrow \mathbb{Z})$, and a program P satisfies $\mathtt{NI}$ iff $[\![\mathsf{P}]\!]\mathfrak{I} \in \mathtt{NI}$. It is trivial to show that $\mathtt{NI} \in \mathtt{SSC}^{\mathtt{H}}_{2}$, hence $\mathsf{P} \models \mathtt{NI}$ iff $\forall X \in \mathfrak{I}^{|2} . [\![\mathsf{P}]\!]X \in \mathtt{NI}$. In particular, we only need to check the sets $\{\mathfrak{d}, \mathfrak{d}'\}$ such that $\mathfrak{d}_{\vdash} =_{\mathsf{L}} \mathfrak{d}'_{\vdash}$. Let $\mathfrak{I}^{|2}_{\mathsf{L}} = \{\{\mathfrak{d}, \mathfrak{d}'\} \in \mathfrak{I}^{|2} \mid \mathfrak{d}_{\vdash} =_{\mathsf{L}} \mathfrak{d}'_{\vdash}\}$. Suppose to have a sound collecting hypersemantics $[\![\mathsf{P}]\!]_{\mathfrak{h}} \in \wp(\wp(\mathbb{DEN})) \rightarrow \wp(\wp(\mathbb{DEN}))$. Then we have that $\mathsf{P} \models \mathtt{NI}$ iff $[\![\mathsf{P}]\!]_{\mathfrak{h}}\mathfrak{I}^{|2}_{\mathsf{L}} \subseteq \mathtt{NI}$. Now we look for a hyper abstract domain allowing us to verify $\mathtt{NI}$. First of all, we abstract sets of sets of traces in sets of traces of sets, namely sets of traces of "abstract memories" $\mathsf{St}^{\natural} \triangleq \mathsf{Var} \rightarrow \wp(\mathbb{Z})$[9]. Hence, consider the following Galois connection $(\langle \wp(\wp(\mathbb{DEN})), \subseteq \rangle, \alpha_{\mathrm{tr}}, \gamma_{\mathrm{tr}}, \langle \wp(\mathsf{St}^{\natural}), \subseteq \rangle)$ with

$$\alpha_{\mathrm{tr}}(\mathcal{X}) = \{\lambda \mathsf{x} \in \mathsf{Var}_{\mathsf{L}} . \{\mathfrak{d}_{\dashv}(\mathsf{x}) \mid \mathfrak{d} \in X\} \mid X \in \mathcal{X}\} \quad \gamma_{\mathrm{tr}}(\mathcal{Y}) = \bigcup\{\mathcal{X} \mid \alpha_{\mathrm{tr}}(\mathcal{X}) \subseteq \mathcal{Y}\}$$

and where $\mathsf{Var}_{\mathsf{L}} \triangleq \{\mathsf{x} \in \mathsf{Var} \mid \mathit{\Gamma}(\mathsf{x}) = \mathsf{L}\}$. This abstraction keeps only the abstract memories collecting values of the low variables, moving from sets of sets of traces to sets of abstract memories. This means that, for all computations starting from sets (of cardinality 2) which agree on low input variables, $\mathtt{NI}$ requires that the resulting sets of values for low variables are constant. Hence, in order to verify $\mathtt{NI}$ we compose this connection with the one defined in Eq. 2. Let $\alpha_{\mathtt{NI}} \triangleq \alpha_{\mathsf{cc}} \circ \mathcal{L}(\alpha_{\mathsf{c}}) \circ \alpha_{\mathrm{tr}}$ where we abuse notation by defining $\forall X \in \wp(\mathsf{St}^{\natural})$, $\alpha_{\mathsf{cc}} \circ \mathcal{L}(\alpha_{\mathsf{c}})(X) \triangleq \lambda \mathsf{x} \in \mathsf{Var}_{\mathsf{L}} . \alpha_{\mathsf{cc}} \circ \mathcal{L}(\alpha_{\mathsf{c}})(\{\mathfrak{d}^{\natural}(\mathsf{x}) \mid \mathfrak{d}^{\natural} \in X\})$, and $\gamma_{\mathtt{NI}}$ is the corresponding concretization. Then we have $(\langle \wp(\wp(\mathbb{DEN})), \subseteq \rangle, \alpha_{\mathtt{NI}}, \gamma_{\mathtt{NI}}, \langle \mathsf{Var} \rightarrow \wp(\mathbb{Z}) \cup \{\mathtt{C}\}, \dot{\subseteq} \rangle)$[10].

Proposition 3. $\mathsf{P} \models \mathtt{NI}$ *iff* $\forall \mathsf{x} \in \mathsf{Var}_{\mathsf{L}} . \alpha_{\mathtt{NI}}([\![\mathsf{P}]\!]_{\mathfrak{h}}\mathfrak{I}^{|2}_{\mathsf{L}})(\mathsf{x}) \neq \mathtt{C}$.

So, we can soundly approximate $\mathtt{NI}$ verification by computing the approximated hypersemantics on the hyper abstract domain $\wp(\mathbb{Z}) \cup \{\mathtt{C}\}$, for all low variables.

[9] Here we implicitly apply a non-relational variables abstraction.

[10] Here $\dot{\subseteq}$ denotes the pointwise set inclusion.

5.2 Abstract Non-Interference

Abstract Non-Interference [18, 19] is a weakening of Non-Interference by abstract interpretation. The idea is to model flows of *properties* of data, modeled as abstractions of data. In particular, let us consider a simplified form of the notion given in [19]. Let $(\langle \wp(\mathbb{Z}), \subseteq \rangle, \alpha_\phi, \gamma_\phi, \langle \Phi, \preccurlyeq_\phi \rangle)$ be an abstraction on input values, fixing what is observable/not-observable of the input. For instance, in the standard case of Non-Interference it is the abstraction observing $\top$ (nothing) of H variables, and the identity of L variables. But it possible to weaken the policy by observing other properties of input variables, where the input property fixed for H variables represents the information we allow to flow, while the property of L ones represents a weakening of what an observer may observe of low inputs. Consider also an output abstraction $(\langle \wp(\mathbb{Z}), \subseteq \rangle, \alpha_\vartheta, \gamma_\vartheta, \langle \Theta, \preccurlyeq_\vartheta \rangle)$, which represents what can be observed in output, in the standard case the identity on L variables and $\top$, i.e., nothing, on H variables. Also in this case, the framework allows us to weaken the policy by fixing a more abstract observable property of L variables. Formally, Abstract Non-Interference is:

$$\mathtt{ANI} = \{X \in \wp(\mathbb{DEN}) \mid \forall \mathfrak{d}, \mathfrak{d}' \in X \,.\, (\alpha_\phi(\mathfrak{d}_\vdash) = \alpha_\phi(\mathfrak{d}'_\vdash) \Rightarrow \alpha_\vartheta(\mathfrak{d}_\dashv) = \alpha_\vartheta(\mathfrak{d}'_\dashv))\}$$

As it happens for Non-Interference, we only need to check $\mathtt{ANI}$ for the sets $\{\mathfrak{d}, \mathfrak{d}'\}$ such that $\alpha_\phi(\mathfrak{d}_\vdash) = \alpha_\phi(\mathfrak{d}'_\vdash)$. Let $\mathfrak{I}^{|2}_\phi \triangleq \{\{\mathfrak{d}, \mathfrak{d}'\} \in \mathfrak{I}^{|2} \mid \alpha_\phi(\mathfrak{d}_\vdash) = \alpha_\phi(\mathfrak{d}'_\vdash)\}$, then we have that $\mathsf{P} \models \mathtt{ANI}$ iff $[\![\mathsf{P}]\!]_\mathfrak{h} \mathfrak{I}^{|2}_\phi \subseteq \mathtt{ANI}$. Consider the Galois insertion of Eq. 3 instantiated on $\mathcal{A} = \Theta$ and $\mathcal{C} = \mathbb{Z}$, and consider the abstraction α_{tr} defined before for $\mathtt{NI}$. Let us define then $\alpha_{\mathtt{ANI}} = \alpha_{\mathfrak{hc}} \circ \mathcal{L}(\alpha_\vartheta) \circ \alpha_{\mathrm{tr}}$. As before, we abuse notation by defining $\forall X \in \wp(\mathsf{St}^\natural)$, $\alpha_{\mathfrak{hc}} \circ \mathcal{L}(\alpha_\vartheta)(X) \triangleq \lambda \mathsf{x} \in \mathsf{Var}_\mathsf{L} \,.\, \alpha_{\mathfrak{hc}} \circ \mathcal{L}(\alpha_\vartheta)(\{\mathfrak{d}^\natural(\mathsf{x}) \mid \mathfrak{d}^\natural \in X\})$, and $\gamma_{\mathtt{ANI}}$ the corresponding concretization. By composition, we have $(\langle \wp(\wp(\mathbb{DEN})) \subseteq \rangle, \alpha_{\mathtt{ANI}}, \gamma_{\mathtt{ANI}}, \langle \Theta_{\mathfrak{hc}}, \dot{\subseteq} \rangle)$.

Proposition 4. $\mathsf{P} \models \mathtt{ANI}$ *iff* $\forall \mathsf{x} \in \mathsf{Var}_\mathsf{L} \,.\, \alpha_{\mathtt{ANI}}([\![\mathsf{P}]\!]_\mathfrak{h} \mathfrak{I}^{|2}_\phi)(\mathsf{x}) \neq \Theta$.

Hence, we can soundly approximate the verification of $\mathtt{ANI}$ by computing the approximated hyper semantics on the hyper domain $\Theta_{\mathfrak{hc}}$ of abstract stores, checking whether all the computations have constant values in Θ for all the low variables.

6 Related Works

The topic of hyperproperties verification is relatively new. In [9], the authors state that it is possible to reduce the verification of a k-hypersafety on a system S to the verification of a safety property on the self-composed system S^k. The self-composition can be sequential, parallel or in an interleaving manner and a lot of works applied this methodology [6, 31, 27, 30]. All these approaches only deal with hypersafety, but we believe that self-composition methods could be extended to the more general bounded subset-closed hyperproperties, in order to verify also non-safety hyperproperties. A very recent work ([3]) proposes a new methodology for proving the absence of timing channels. This work is based on the idea of “decomposition instead of self-composition” [3]. The authors claim that self-composition is computationally to expensive to be used in practice, so they propose a different approach. The idea is to partition the program semantics and to analyze each partition with standard methods. All previous approaches are proven to be sound and complete for k-hypersafety, but our methodology is sound and complete for the more general subset-closed hyperproperties.

Besides the reduction to safety, in [1] the authors introduce a runtime refutation methods for k-safety, based on a three-valued logic. Similarly, [8, 15] 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.

The use of abstract interpretation in hyperproperties verification is limited to [4, 25, 32]. In [4], the authors deal with information flow specifications and they focus on the definition of the abstract domains over sets of sets needed for the analysis. They proposed an hyper collecting semantics computed denotationally on the code of the program to analyze. We already highlighted (Sect. 4.1 and Sect. 5) the links between the present work and [4]. Our approach is a generalization of the methodologies of [4], since their hypercollecting semantics is an instance of our semantics lift and the abstract domains they use follow our inner/outer abstractions pattern. In [25] we extend the hierarchy of semantics of a transition system [11], in order to cope with hyperproperties verification. Furthermore, we introduce the notion of subset-closed hyperproperties. Our present work follows this latter, but it is focused on how it is possible to construct a collecting hypersemantic, for computer programs, lifting a given collecting semantics (Sect. 3). Furthermore, our work aims at the verification of particular subset-closed hyperproperties. Finally, in [32] the authors use abstract interpretation in order to define an ad-hoc semantics at the level of sets of sets suitable for the verification of a particular hyperproperty called “data input usage”. This latter is not subset-closed, hence it is beyond the scope of the present work. Furthermore, their goal is not to give a general methodology for defining new verification methods for hyperproperties, as we do for subset-closed hyperproperties. Indeed, despite the interesting approach, their work can be applied only to the particular hyperproperty they introduced.

7 Conclusion and Future Works

In this work, we made another little step into the understanding of hyperproperties. In particular, we reasoned about particular subset-closed hyperproperties, which are more suitable for verification. Subset-closed hyperproperties are those allowing to disprove program hyperproperties by finding a subset of its semantics which do 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 collecting semantics to sets of sets and how to build hyper abstract domains. Putting all the ingredients together, we specified the general recipe for defining an hyperanalysis (i.e., a static analysis at the level of sets of sets) for bounded subset-closed hyperproperties. It is clear that, such an analysis would be useful, not only for checking (abstract) non-interference in its different forms (e.g., declassified) [19, 5, 23], but also in other contexts related to information flow such as abstract slicing [24, 26] or injection vulnerability analysis [7].

As future works, we want to investigate whether it is possible to compute a collecting hypersemantics reducing as much as possible the spurious information added by lifting semantics at the hyperlevel. We already observed that this is not a problem for $\mathtt{SSC}^{\mathtt{H}}$ hyperproperties, we wonder whether we can improve the proposed framework by enriching the information represented by states, in order to reduce the noise added by collecting at the hyperlevel. Moreover, we want to deepen the link between hyperproperties and the problem of analyzing analyzers, aiming at systematically analyzing static analyses [16]. In particular, we believe that the hyperdomains, introduced in Sect. 4, can be used not only for hyperproperties verification but also for this latter purpose.

Acknowledgments

We thank Roberto Giacobazzi and Francesco Ranzato for sharing with us their preliminary work on analyzing analyses [20], which has many connections with the present work and may create interesting future collaborations. Finally, we would like to thank the anonymous reviewers for the useful suggestions and comments, helping us in improving the presentation of our work.

References

1. Agrawal, S., Bonakdarpour, B.: Runtime verification of k-safety hyperproperties in HyperLTL. In: IEEE 29th Comp. Security Foundations Symp. pp. 239–252 (2016)
2. Alpern, B., Schneider, F.B.: Defining liveness. Information Processing Letters 21(4), 181–185 (1985)
3. Antonopoulos, T., Gazzillo, P., Hicks, M., Koskinen, E., Terauchi, T., Wei, S.: Decomposition instead of self-composition for proving the absence of timing channels. In: PLDI, Proceedings. pp. 362–375 (2017)
4. Assaf, M., Naumann, D.A., Signoles, J., Totel, E., Tronel, F.: Hypercollecting semantics and its application to static analysis of information flow. In: POPL, Proceedings. pp. 874–887 (2017)
5. Banerjee, A., Giacobazzi, R., Mastroeni, I.: What you lose is what you leak: Information leakage in declassification policies. ENTCS 173, 47–66 (2007)
6. Barthe, G., D'Argenio, P.R., Rezk, T.: Secure information flow by self-composition. In: Proc. of 17th IEEE Comp. Sec. Foundations Workshop. pp. 100–114 (2004)
7. Buro, S., Mastroeni, I.: Abstract code injection - A semantic approach based on abstract non-interference. In: VMCAI 2018, Proceedings. pp. 116–137 (2018)

8. Clarkson, M.R., Finkbeiner, B., Koleini, M., Micinski, K.K., Rabe, M.N., Sánchez, C.: Temporal logics for hyperproperties. In: POST; Proceedings (2014)
9. Clarkson, M.R., Schneider, F.B.: Hyperproperties. Journal of Computer Security 18(6), 1157–1210 (sep 2010)
10. Cohen, E.: Information transmission in computational systems. SIGOPS Oper. Syst. Rev. 11(5), 133–139 (nov 1977)
11. Cousot, P.: Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. Theor. Comput. Sci. 277(1-2), 47–103 (2002)
12. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: POPL, Proceedings. pp. 238–252 (1977)
13. Cousot, P., Cousot, R.: Systematic design of program analysis frameworks. In: POPL, Proceedings. pp. 269–282 (1979)
14. Cousot, P., Cousot, R.: Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages). In: ICCL, Proceedings. pp. 95–112 (1994)
15. Finkbeiner, B., Rabe, M.N., Sánchez, C.: Algorithms for model checking HyperLTL and HyperCTL. In: CAV, ProceedingsI. pp. 30–48 (2015)
16. Giacobazzi, R., Logozzo, F., Ranzato, F.: Analyzing program analyses. In: POPL, Proceedings. pp. 261–273 (2015)
17. Giacobazzi, R., Mastroeni, I.: Transforming semantics by abstract interpretation. Theor. Comput. Sci. 337(1-3), 1–50 (2005)
18. Giacobazzi, R., Mastroeni, I.: Abstract non-interference: Parameterizing non-interference by abstract interpretation. In: POPL, Proceedings. pp. 186–197 (2004)
19. Giacobazzi, R., Mastroeni, I.: Abstract non-interference: A unifying framework for weakening information-flow. ACM Trans. Priv. Secur. 21(2), 9:1–9:31 (2018)
20. Giacobazzi, R., Ranzato, F.: Personal communication.
21. Goguen, J.A., Meseguer, J.: Security policies and security models. In: IEEE Symposium on Security and Privacy. pp. 11–20 (1982)
22. Hunt, S., Mastroeni, I.: The PER model of abstract non-interference. In: Static Analysis, 12th International Symposium, Proceedings. pp. 171–185 (2005)
23. Mastroeni, I., Banerjee, A.: Modelling declassification policies using abstract domain completeness. MSCS 21(6), 1253–1299 (2011)
24. Mastroeni, I., Nikolic, D.: Abstract program slicing: From theory towards an implementation. In: ICFEM 2010, Proceedings. pp. 452–467 (2010)
25. Mastroeni, I., Pasqua, M.: Hyperhierarchy of semantics - A formal framework for hyperproperties verification. In: SAS, Proceedings. pp. 232–252 (2017)
26. Mastroeni, I., Zanardini, D.: Abstract program slicing: An abstract interpretation-based approach to program slicing. ACM TOCL 18(1), 7:1–7:58 (2017)
27. Naumann, D.A.: From coupling relations to mated invariants for checking information flow. In: ESORICS, Proceedings. pp. 279–296 (2006)
28. Pasqua, M., Mastroeni, I.: On topologies for (hyper)properties. In: ICTCS, Proceedings. pp. 1–12 (2017), http://ceur-ws.org/Vol-1949/ICTCSpaper13.pdf
29. Ranzato, F., Tapparo, F.: Strong preservation as completeness in abstract interpretation. In: Schmidt, D. (ed.) ESOP, Proceedings. pp. 18–32 (2004)
30. Sousa, M., Dillig, I.: Cartesian hoare logic for verifying k-safety properties. In: PLDI, Proceedings. pp. 57–69 (2016)
31. Terauchi, T., Aiken, A.: Secure information flow as a safety problem. In: Proc. of 12th International Conference on Static Analysis. pp. 352–367 (2005)
32. Urban, C., Müller, P.: An abstract interpretation framework for input data usage. In: ESOP. pp. 683–710 (2018)