Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 337 (2005) 1–50
Theoretical Computer Science
www.elsevier.com/locate/tcs

Fundamental Study

Transforming semantics by abstract interpretation

Roberto Giacobazzi*, Isabella Mastroeni

Dipartimento di Informatica, Università di Verona, Strada Le Grazie 15, 37134 Verona, Italy

Received 3 February 2004; received in revised form 7 December 2004; accepted 13 December 2004

Communicated by G. Levi

Abstract

In 1997, Cousot introduced a hierarchy where semantics are related with each other by abstract interpretation. In this field we consider the standard abstract domain transformers, devoted to refine abstract domains in order to include attribute independent and relational information, respectively the reduced product and power of abstract domains, as domain operations to systematically design and compare semantics of programming languages by abstract interpretation. We first prove that natural semantics can be decomposed in terms of complementary attribute independent observables, leading to an algebraic characterization of the symmetric structure of the hierarchy. Moreover, we characterize some structural property of semantics, such as their compositionality, in terms of simple abstract domain equations. This provides an equational presentation of most well known semantics, which is parametric on the observable and structural property of the semantics, making it possible to systematically derive abstract semantics, e.g. for program analysis, as solutions of abstract domain equations.

Keywords: Abstract interpretation; Comparative semantics; Domain theory; Compositionality; Constraint programming

1. Introduction

Since its origin in 1977, abstract interpretation [11] has been widely used, implicitly or explicitly, to describe and formalize approximate computations in many different areas of computer science, from its very beginning use in formalizing (compile-time) program

* Corresponding author. Tel.: +39 45 802 7995; fax: +39 45 802 7982.
E-mail addresses: roberto.giacobazzi@univr.it (R. Giacobazzi), mastroeni@sci.univr.it (I. Mastroeni).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.12.021

analysis frameworks to more recent applications in model checking, program verification, data security, type inference, automated deduction, and comparative semantics. This justifies a now well established definition of abstract interpretation as *a general theory to approximate the semantics of discrete dynamic systems* [8]. This is particularly striking in comparative semantics, where semantics at different levels of abstraction can be compared with each other by abstract interpretation [10]. In this paper, we analyze the most well-known structural properties of semantics, such as their precision, compositionality, and relation between complementary observables, by using standard abstract interpretation techniques. We prove that most of these properties be characterized in terms of properties of the corresponding abstractions. This is achieved by isolating a suitable set of abstract domain transformers which allows us to design abstractions accordingly, providing a characterization of semantics of programming languages as solutions of simple abstract domain equations, involving both some basic observable property which has to be observed by the semantics and the abstract domain transformers necessary in order to achieve a suitable structural property.

1.1. The scenario

Semantics is central in the construction of any abstract interpretation. The so-called *concrete semantics* specifies the observable property of program behavior and any more abstract semantics, e.g. decidable semantics for program analysis, can be derived by abstraction. As a consequence, a semantics, at any level of abstraction, can be fully specified as an abstract interpretation of a more concrete semantics. This key idea is the basis of Cousot's design of a complete hierarchy of semantics of programming languages [9,15]. A number of semantics including big-step, termination and non-termination, Plotkin's natural, Smyth's demonic, Hoare's angelic relational and corresponding denotational, Dijkstra's predicate transformer weakest-precondition and weakest-liberal precondition and Hoare's partial and total axiomatic semantics, have all been derived by successive abstractions from an (operational) maximal trace semantics of a transition system. The resulting hierarchy (here called Cousot's hierarchy) provides a complete account on the structure and the relative precision of most well known semantics of programming languages. One of the major challenge in Cousot's construction is that *semantics are abstract domains*. Therefore they can be transformed, refined, decomposed, and composed similarly to what is usually done with abstract domains in static program analysis. This view of semantics as domains provides both a better insight on the structure and relative precision of traditional well known semantics of programming languages and the possibility to systematically specify new semantics by composition, decomposition, refinement and simplification of existing ones, by manipulating the corresponding domains.

1.2. The main results

In this paper, we treat the Cousot's hierarchy of semantics as an *algebra of semantics*, namely we apply algebraic operations to semantics, here seen as abstract domains. Our aim is to relate the properties of semantics with the properties of the abstract domain transformations used in their design. This is achieved by considering the main operations

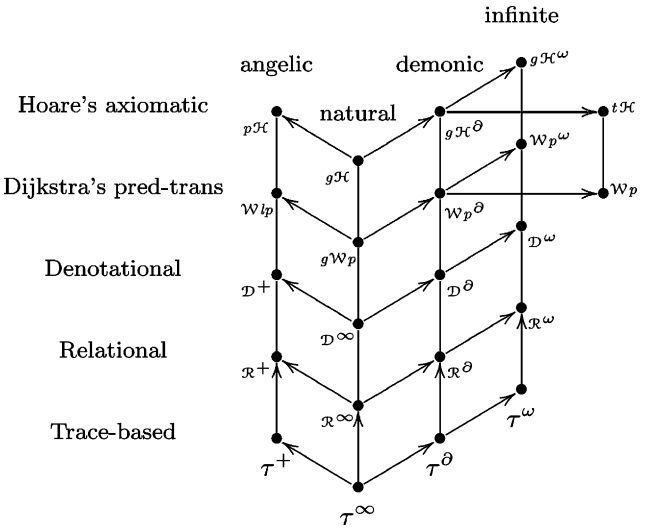

Fig. 1. Cousot's hierarchy.

for abstract domain transformation in [13], i.e., the *attribute independent* reduced product and the *relational* reduced power composition. The reduced product of two domains A and B consists in building the domain that observes all the information contained in both A and B, independently of each other. The reduced power, of two domains A and B, builds the domain of all the functional relations between the elements of A and B. We prove that all the semantics in Cousot's hierarchy can be specified as solutions of simple abstract domain equations involving attribute independent and relational combinators. The duality between relational and attribute independent combination of abstract domains is reflected in the structure of the paper.

In the first part of this paper, we analyze Cousot's hierarchy of semantics and we characterize its symmetric structure (see Fig. 1) in terms of a purely algebraic manipulation of domains. We prove that complementary information characterizes the symmetric structure of Cousot's hierarchy. We consider the *reduced product*, introduced in [13], as the basic operation for composing semantics, and its inverse operation, *abstract domain complementation* introduced in [7], as the basic operation for decomposing semantics. Given two semantics $\mathcal{S}_1$ and $\mathcal{S}_2$, the product semantics $\mathcal{S}_1 \sqcap \mathcal{S}_2$ is the most abstract semantics which is as precise as both $\mathcal{S}_1$ and $\mathcal{S}_2$, namely which is able to observe both the observables of $\mathcal{S}_1$ and $\mathcal{S}_2$. Domain complementation was originally introduced to decompose abstract domains in static program analysis, and it is the inverse operation of reduced product. In our case, this operation provides a systematic methodology for decomposing semantics by characterizing the most abstract semantics $\mathcal{S}$ which, when composed with a given semantics $\mathcal{B}$, yields the semantics $\mathcal{C} = \mathcal{S} \sqcap \mathcal{B}$ as result. These operations provide advanced methods for comparing semantics with respect to their relative expressiveness. This is particularly relevant in the study of semantics observing complementary behaviors of programs, e.g. finite and infinite computations of a transition system. According to Cousot's

construction, in fact, any semantic style (trace-operational, relational, denotational, Dijkstra's predicate transformer and Hoare's axiomatic semantics) may have a corresponding *natural*, *finite/angelic*, *demonic*, and *infinite* nature. The nature of each semantics defines a corresponding observable behavior of programs (later called *observable*), which can be parameterized according to the chosen semantic style, and it corresponds respectively to: terminating, chaotic non-terminating, and infinite computations. We prove that natural semantics are always the reduced product of finite/angelic and demonic or infinite semantics, and that these semantics factorize the natural semantic construction by complementation. In particular any finite/angelic semantics can be systematically derived as the domain complementation in the natural semantics of the demonic or infinite semantics. Moreover, we prove that finite/angelic and infinite semantics form the most abstract decomposition of any natural semantics, and that demonic semantics can be further factorized in terms of infinite semantics and of a new semantics, here called *slothful*, which is unable to observe infinite computations when programs may produce any possible output. Then we prove that this highly symmetric structure is a consequence of a common pattern of abstraction between semantic styles and observables, ranging from operational trace-based to the more abstract Hoare's axiomatic semantics. We characterize this pattern in terms of some basic properties of the closure operators, associated with the semantics abstractions in Cousot's hierarchy. This allows us to prove the basic results on symmetric semantics for the trace-operational semantic style only, deriving the results concerning all the other styles and observables as a simple consequence. These results provide both an algebraic characterization of complementary observable properties in semantics, and a decomposition result for observable properties of programs in terms of complementary observables, similar to the well known Alpern & Schneider's safety/liveness decomposition of properties of concurrent program executions (cf. [2]). This part is an extended and revised version of [25]. The attribute independent combination of semantics does not include in abstractions the relational information which is typically included in compositional semantics, such as in the denotational semantics.

In the second part of this paper, we consider the *reduced power* operation [13,30], for abstract domain refinement, as the basic operation able to include input/output relations in domains. Reduced power has been proved to give the necessary structure of abstract domains in order to model relational properties of in program analysis [14,32,37,42]. Let $\mathcal{S}$ be the concrete domain, and $\mathcal{S}_1$ and $\mathcal{S}_2$ two abstractions of $\mathcal{S}$. The reduced power $\mathcal{S}_1 \longrightarrow \mathcal{S}_2$ is the domain of all the monotone functions from elements of $\mathcal{S}_1$ to elements of $\mathcal{S}_2$. We prove that the compositional semantics observing finite computations only, i.e., angelic denotational and weakest-liberal precondition semantics, can be systematically derived as the most abstract semantics closed under reduced power, and including the semantics which observes, respectively, final and initial states of finite traces only. These semantics are the most abstract ones which are compositional for observing, respectively, final and initial states. Compositionality here means that, if $[\![P_1]\!]$ and $[\![P_2]\!]$ are the semantics of program components P_1 and P_2, and $\diamond$ is a syntactic operator for program composition, then there exists an operation $\circ$ such that: $[\![P_1 \diamond P_2]\!] = [\![P_1]\!] \circ [\![P_2]\!]$. We show that most well known compositional semantics of imperative programs, such as the standard angelic denotational and weakest-liberal precondition semantics, can be systematically derived as solutions of simple abstract domain equations. We consider sequential syntactic composition of

programs and trace composition $\frown$ for composing semantics. In this case, compositionality boils down to $[\![P_1; P_2]\!] = [\![P_1]\!] \frown [\![P_2]\!]$. As a consequence of these results, we obtain a systematic method for the design of semantics, where semantics can be designed as solutions of domain equations involving the basic operations of reduced product, power and domain complementation. Our results are general, and can be applied to any programming language whose semantics can be defined in terms of traces of program states in a transition system. As an example, we apply our construction to the case of concurrent constraint programming *ccp* languages [40]. These languages well fit into Cousot's hierarchy of semantics being easily defined in terms of traces of constraints in a transition system. We prove that both, Saraswat's closure-based denotational [41], and de Boer et al.'s predicate transformer semantics [16], can be derived by composing non-compositional semantics observing, respectively, the final and initial constraints in terminating computations. This provides an equational presentation of semantics as abstract interpretation of the maximal traces of constraints, associated with an operational small-step transition system semantics of *ccp* programs. Consequently, the corresponding finite/angelic, demonic, and infinite semantics, can be specified by domain complementation.

1.3. State of the art

The foundation for a theory of abstract domains was fixed in [13]. In this paper, the authors provide the main structure of abstract domains enjoying *Galois connections*, and some basic operators to systematically compose domains, i.e., the *reduced product* and the *reduced power* operations. Since then, a number of papers have developed new domain operations, and studied the impact of these operations in the design of abstract interpretations (e.g. see [29] for a survey). The notion of *domain refinement* and *domain simplification*, introduced in [21,29], provided the very first generalization of these ideas. Intuitively, a refinement is any operator performing an action of refinement with respect to the standard order of precision, e.g. by adding information to domains; while simplificators perform the dual action of "taking out" information from domains. Few examples are known on the use of systematic domain operations in abstract interpretation to reason about the structure and the expressiveness of semantics of programming languages. Most of these examples are in the semantics of logic programs, which basically relies on the hierarchy of semantics developed in [6,24]. In [28], the authors study the relations between different semantics of logic programs, namely success pattern semantics, computed answer substitution semantics and call pattern semantics by means of complementation. This is the very first and unique example of the use of complementation in systematic semantics design. In [9], the domain operation of *tensor product* [43] is considered in order to design Hoare's axiomatic semantics by exploiting the adjoint relation between pre- and post-conditions in Hoare triples. As far as compositionality is concerned, the very first and, up to our knowledge, unique example of construction of compositional semantics by abstract domain transformation, is in [30]. In this work, the authors proved that compositional semantics of logic programs in [5,23] can be systematically designed by a generalization of Cousot's reduced cardinal power operation [13], from non-compositional semantics of computed answer substitution. This work represents a starting point for the second part of our paper, which generalizes the results in [30] to arbitrary programming languages whose semantics can be specified by a

transition system of states. In [26], a similar method has been considered in order to derive compositional models for program slicing. These models allow transfinite semantics and provide an adequate framework for specifying natural compositional semantics observing both termination and non-termination.

2. Preliminaries

2.1. Basic notions

If S and T are sets, then $\wp(S)$ denotes the powerset of S, $S \setminus T$ denotes the set-difference between S and T, $S \subset T$ denotes strict inclusion, and for a function $f : S \to T$ and $X \subseteq S$, $f(X) \stackrel{\text{def}}{=} \{f(x) \mid x \in X\}$. By $f|_X$ we denote the function f whose domain is restricted to X. By $g \circ f$ we denote the composition of the functions f and g, i.e., $g \circ f \stackrel{\text{def}}{=} \lambda x.g(f(x))$. The notation $\langle P, \leqslant \rangle$ denotes a poset P with ordering relation $\leqslant$, while $\langle P, \leqslant, \vee, \wedge, \top, \bot \rangle$ denotes a complete lattice P, with ordering $\leqslant$, *lub* $\vee$, *glb* $\wedge$, greatest element (top) $\top$, and least element (bottom) $\bot$. Often, $\leqslant_P$ will be used to denote the underlying ordering of a poset P, and $\vee_P$, $\wedge_P$, $\top_P$ and $\bot_P$ denote the basic operations and elements of a complete lattice. The notation $C \cong A$ denotes that C and A are isomorphic ordered structures. An element $x \in P$ is *meet-irreducible* if $x \neq \top$ and $x = a \wedge b$ implies $x \in \{a, b\}$. The set of meet-irreducible elements in P is denoted $Mirr(P)$. The downward closure of $S \subseteq P$ is defined as $\downarrow S \stackrel{\text{def}}{=} \{x \in P \mid \exists y \in S.\ x \leqslant_P y\}$, and for $x \in P$, $\downarrow x$ is a shorthand for $\downarrow \{x\}$, while the upward closure $\uparrow$ is dually defined. $S \longrightarrow T$ denotes the set of all functions from S to T. We use the symbol $\sqsubseteq$ to denote pointwise ordering between functions: If S is any set, P a poset, and $f, g : S \to P$ then $f \sqsubseteq g$ if for all $x \in S$, $f(x) \leqslant_P g(x)$. Let C and A be complete lattices. Then, $C \xrightarrow{\text{m}} A$, $C \xrightarrow{\text{c}} A$, $C \xrightarrow{\text{a}} A$, and $C \xrightarrow{\text{coa}} A$ denote, respectively, the set of all monotone, (Scott-)continuous, additive, and co-additive functions from C to A. Recall [1] that $f \in C \xrightarrow{\text{c}} A$ iff f preserves *lub*'s of (non-empty) chains iff f preserves *lub*'s of directed subsets, and $f : C \to A$ is (completely) additive if f preserves *lub*'s of all subsets of C (emptyset included). Co-additivity is defined by duality. We denote by $lfp_{\bot}^{\leqslant} f$ and $gfp_{\top}^{\leqslant} f$, respectively, the least and greatest fix-point, when they exist, of an operator f on a poset. If $f \in C \xrightarrow{\text{c}} C$ then $lfp_{\bot_C}^{\leqslant_C} f = \vee_{i \in \mathbb{N}} f^i(\bot_C)$, where, for any $i \in \mathbb{N}$ and $x \in C$, the ith power of f in x is inductively defined as follows: $f^0(x) = x$; $f^{i+1}(x) = f(f^i(x))$. Dually, if f is co-continuous then $gfp_{\top_C}^{\leqslant_C} f = \wedge_{i \in \mathbb{N}} f^i(\top_C)$. $\{f^i(\bot_C)\}_{i \in \mathbb{N}}$ and $\{f^i(\top_C)\}_{i \in \mathbb{N}}$ are called, respectively, the *lower* and *upper Kleene's iteration sequences of* f (see [12]).

2.2. Abstract interpretation

Abstract domains can be equivalently formulated in many different ways. The most used ones are Galois connections and upper closure operators [13]. An *upper closure operator* on a poset P is an operator $\rho : P \to P$ monotone, idempotent and extensive ($\forall x \in P.\ x \leqslant_P \rho(x)$). The set of all upper closure operators on P is denoted by $uco(P)$. Let $\langle C, \leqslant, \vee, \wedge, \top, \bot \rangle$ be a complete lattice. A basic property of closure operators is that each closure is uniquely determined by the set of its fix-points $\rho(C)$. For upper closures: $X \subseteq C$ is the set of fix-points of an upper closure on C iff X is a *Moore-family* of C, i.e.,

$X = \mathcal{M}(X) \stackrel{\text{def}}{=} \{\wedge S \mid S \subseteq X\}$—where $\wedge\emptyset = \top \in \mathcal{M}(X)$. For any $X \subseteq C$, $\mathcal{M}(X)$ is called the *Moore-closure* of X in C, i.e., $\mathcal{M}(X)$ is the least (w.r.t. set-inclusion) subset of C which contains X and it is a Moore-family of C. It turns out that $\langle \rho(C), \leqslant \rangle$ is a complete meet subsemilattice of C (i.e., $\wedge$ is its *glb*). Often, we will find particularly convenient to identify closure operators with their sets of fix-points. If C is a complete lattice then $uco(C)$ ordered pointwise is also a complete lattice, denoted by $\langle uco(C), \sqsubseteq, \sqcup, \sqcap, \lambda x.\top, \lambda x.x \rangle$, where for every $\rho, \eta \in uco(C)$, $\{\rho_i\}_{i\in I} \subseteq uco(C)$ and $x \in C$:

- $\rho \sqsubseteq \eta$ iff $\forall y \in C.\ \rho(y) \leqslant \eta(y)$ iff $\eta(C) \subseteq \rho(C)$;
- $(\sqcap_{i\in I}\rho_i)(x) = \wedge_{i\in I}\rho_i(x)$;
- $(\sqcup_{i\in I}\rho_i)(x) = x \Leftrightarrow \forall i \in I.\ \rho_i(x) = x$.

If $\alpha \in C \xrightarrow{\text{m}} A$ and $\gamma \in A \xrightarrow{\text{m}} C$ are monotone functions such that $\lambda x.x \sqsubseteq \gamma \circ \alpha$ and $\alpha \circ \gamma \sqsubseteq \lambda x.x$, then (A, α, γ, C) is called a *Galois connection* (GC for short) or *adjunction* between C and A, also denoted $\langle C, \leqslant_C \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leqslant_A \rangle$. Note that in a GC, for any $x \in C$ and $y \in A$: $\alpha(x) \leqslant_A y \Leftrightarrow x \leqslant_C \gamma(y)$ where the functions are $\gamma(y) = \bigvee \{x \mid \alpha(x) \leqslant y\}$ and $\alpha(x) = \bigwedge \{y \mid x \leqslant \gamma(y)\}$. The set of all GCs between two complete lattices A and C is the tensor product $A \otimes C$, which is a complete lattice and $A \otimes C \cong A \xrightarrow{\text{a}} C \cong C \xrightarrow{\text{coa}} A$ [43]. If in addition $\alpha \circ \gamma = \lambda x.x$ ($\gamma \circ \alpha = \lambda x.x$), then (A, α, γ, C) is a *Galois insertion* (GI) (resp. *projection*) also denoted $\langle C, \leqslant_C \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows\!\!\!\!\rightarrow}} \langle A, \leqslant_A \rangle$ (resp. $\langle C, \leqslant_C \rangle \underset{\alpha}{\overset{\gamma}{\leftarrow\!\!\!\!\leftrightarrows}} \langle A, \leqslant_A \rangle$) of A in C. It is worth noting that $A \cong C$ if and only if the connection in an isomorphism, i.e., $\langle C, \leqslant_C \rangle \underset{\alpha}{\overset{\gamma}{\leftarrow\!\!\!\!\leftrightarrows\!\!\!\!\rightarrow}} \langle A, \leqslant_A \rangle$.

Let $f : C \to C$ be a monotone concrete semantic function and let $f^\sharp : A \to A$ be a corresponding *abstract function*, where $\langle C, \leqslant_C \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows\!\!\!\!\rightarrow}} \langle A, \leqslant_A \rangle$. Then, $\langle A, f^\sharp \rangle$ is a sound abstract interpretation — or $f^\sharp$ is a correct approximation of f relatively to A — when $\forall c \in C.\ \alpha(f(c)) \leqslant_A f^\sharp(\alpha(c))$. On the other hand, $\langle A, f^\sharp \rangle$ is complete when the equality holds, i.e., $\alpha \circ f = f^\sharp \circ \alpha$.

The standard abstract interpretation framework is based on the adjoint relation between abstraction and concretization functions [11]. The concrete and abstract domains, C and A, are assumed to be complete lattices and are related by two maps forming a GC (A, α, γ, C). Following a standard terminology, A is called an abstraction of C, and C is a concretization of A. If (A, α, γ, C) is a GI, then each value of the abstract domain A is useful in representing C, because all the elements of A represent distinct members of C, being γ 1-1. Any GC may be lifted to a GI by identifying, in an equivalence class, those values, of the abstract domain, with the same concretization. This process is known as *reduction* of the abstract domain. Note that any GI (A, α, γ, C) uniquely determines an upper closure operator, i.e., $\gamma \circ \alpha \in uco(C)$, and conversely, any closure operator $\rho \in uco(C)$ uniquely determines a GI $(\rho(C), \rho, id, C)$, up to isomorphic representation of domain's objects. Hence, we will identify $uco(C)$ with the so-called *lattice $\mathfrak{L}_C$ of abstract interpretations* of C (cf. [11, Section 7] and [13, Section 8]), i.e., the complete lattice of all possible abstract domains (modulo isomorphic representation of their objects) of the concrete domain C. The pointwise ordering on $uco(C)$ corresponds precisely to the standard ordering used to compare abstract domains, as regards their precision: A_1 is more precise than A_2 (i.e., A_2 is an abstraction of A_1) iff $A_1 \sqsubseteq A_2$ in $uco(C)$ iff $\langle A_1, \leqslant_{A_1} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows\!\!\!\!\rightarrow}} \langle A_2, \leqslant_{A_2} \rangle$.

Let $\{A_i\}_{i\in I} \subseteq uco(C)$: $\sqcup_{i\in I} A_i$ is the most concrete in $\mathfrak{L}_C$ which is an abstraction of all the A_i's, i.e., $\sqcup_{i\in I} A_i$ is the *least* (w.r.t. $\sqsubseteq$) *common abstraction* of all the A_i's; and $\sqcap_{i\in I} A_i$ is (isomorphic to) the well known *reduced product* (basically cartesian product plus reduction) of all the A_i's, or, equivalently, it is the most abstract domain in $\mathfrak{L}_C$ which is more concrete than every A_i. Let us remark that the reduced product can be also characterized as Moore-closure of set-union, i.e., $\sqcap_{i\in I} A_i = \mathcal{M}(\cup_{i\in I} A_i)$.

3. Cousot's semantics hierarchy

In this section, we recall Cousot's hierarchy of semantics [10,15]. Semantics, in the hierarchy, are derived as abstract interpretations of a more concrete operational semantics that associates a discrete transition system with each well-formed program. A transition system is a pair $\langle \Sigma, \tau\rangle$, where Σ is a non-empty set of states, and $\tau \subseteq \Sigma \times \Sigma$ is a binary transition relation between a state and its possible successors. In the following, Σ^+ and $\Sigma^\omega \stackrel{\text{def}}{=} \mathbb{N} \longrightarrow \Sigma$ denote, respectively, the set of all the finite non-empty sequences, and the set of all the infinite sequences, of symbols in Σ. Given a sequence $\sigma \in \Sigma^\infty \stackrel{\text{def}}{=} \Sigma^+ \cup \Sigma^\omega$, its length is denoted by $|\sigma| \in \mathbb{N} \cup \{\omega\}$ and its ith element is denoted by σ_i. Moreover, in the following, when $|\sigma| = n \leqslant \omega$, $\sigma_\vdash$ will denote σ_0 and $\sigma_\dashv$ will denote σ_{n-1}. A non-empty finite (infinite) *trace* σ is a finite (infinite) sequence of program states, where two consecutive elements are in the transition relation τ, i.e., for all $i < |\sigma|$: $\langle \sigma_i, \sigma_{i+1}\rangle \in \tau$. A *generic trace* is any such element in Σ^∞. The *maximal trace semantics* of a transition system [15] is $\tau^\infty \stackrel{\text{def}}{=} \tau^+ \cup \tau^\omega$, where if $T \subseteq \Sigma$ is a set of final/blocking states $\tau^{\dot{n}} = \{\sigma \in \Sigma^+ \,||\sigma| = n, \forall i \in [1, n) \,.\, \langle \sigma_{i-1}, \sigma_i\rangle \in \tau\}$, $\tau^\omega = \{\sigma \in \Sigma^\omega \,|\, \forall i \in \mathbb{N} \,.\, \langle \sigma_i, \sigma_{i+1}\rangle \in \tau\}$, $\tau^+ = \cup_{n>0}\{\sigma \in \tau^{\dot{n}} \,|\, \sigma_\dashv \in T\}$, and $\tau^n = \tau^{\dot{n}} \cap \tau^+$. In the following, we will use the *concatenation* operation between traces: the concatenation $\sigma = \eta^\frown\xi$ of the traces $\eta, \xi \in \Sigma^\infty$ is defined only if $\eta_{|\eta|-1} = \xi_0$. In this case, σ has length $|\sigma| = |\eta| + |\xi| - 1$ and it is such that $\sigma_l = \eta_l$ for each $0 \leqslant l < |\eta|$, while $\sigma_{|\eta|-1+n} = \xi_n$ if $0 \leqslant n < |\xi|$. Moreover, if $\eta \in \Sigma^\omega$, then, for each $\xi \in \Sigma^\infty$, we have $\eta^\frown\xi = \eta$.

The semantics τ^∞ has been obtained in [15] as the least fix-point of the monotone operator $F^\infty : \wp(\Sigma^\infty) \to \wp(\Sigma^\infty)$, defined on traces as $F^\infty(X) = \tau^1 \cup \tau^{\dot{2}} {}^\frown X$. This operator provides a bi-induction (induction and co-induction) on the complete lattice of the maximal trace semantics $\langle \wp(\Sigma^\infty), \sqsubseteq^\infty, \sqcap^\infty, \sqcup^\infty, \sqcap^\infty, \Sigma^+, \Sigma^\omega\rangle$, where $X \sqsubseteq^\infty Y$ if and only if $X \cap \Sigma^+ \subseteq Y \cap \Sigma^+$ and $Y \cap \Sigma^\omega \subseteq X \cap \Sigma^\omega$. This order, later called the *computational order*, allows us to combine both least and greatest fix-point process in a unique least fix-point presentation: finite (terminating) traces are obtained by induction (*least fix-point*) of F^∞ on $\langle \wp(\Sigma^+), \subseteq\rangle$, and infinite traces are obtained by co-induction (*greatest fix-point*) on $\langle \wp(\Sigma^\omega), \subseteq\rangle$, which corresponds to the *least fix-point* of F^∞ on $\langle \wp(\Sigma^\omega), \supseteq\rangle$. In this case: $\tau^\infty = lfp^{\sqsubseteq^\infty}_{\Sigma^\omega} F^\infty$ (see [10,15] for details). Cousot proved also that the natural trace semantics can be calculated as the *greatest fix-point*, of the same function, on the domain with the usual inclusion order, here called *approximation order*, namely $\tau^\infty = gfp^{\subseteq}_{\Sigma^\infty} F^\infty$.

All the semantics, in the hierarchy, are derived as abstract interpretation of the trace-based semantics. In particular, each semantics in natural style corresponds to a suitable

Table 1
Basic natural-style semantics as abstract interpretations

Semantics	Domain relation	Abstraction and concretization
$\mathcal{R}^\infty = \alpha^{\mathcal{R}}(\tau^\infty)$	$\langle \wp(\Sigma^\infty), \subseteq\rangle \underset{\alpha^{\mathcal{R}}}{\overset{\gamma^{\mathcal{R}}}{\leftrightarrows}} \langle \wp(\Sigma \times \Sigma_\perp), \subseteq\rangle$	$\alpha^{\mathcal{R}}(X) = \{\langle \sigma_\vdash, \sigma_\dashv\rangle \mid \sigma \in X^+ \} \cup \{\langle \sigma_\vdash, \perp\rangle \mid \sigma \in X^\omega \}$ $\gamma^{\mathcal{R}}(Y) = \{\sigma \in \Sigma^+ \mid \langle \sigma_\vdash, \sigma_\dashv\rangle \in Y \} \cup \{\sigma \in \Sigma^\omega \mid \langle \sigma_\vdash, \perp\rangle \in Y \}$
$\mathcal{D}^\infty = \alpha^{\mathcal{D}}(\mathcal{R}^\infty)$	$\langle \wp(\Sigma \times \Sigma_\perp), \subseteq\rangle \underset{\alpha^{\mathcal{D}}}{\overset{\gamma^{\mathcal{D}}}{\leftrightarrows}} \langle \Sigma \to \wp(\Sigma_\perp), \sqsubseteq\rangle$	$\alpha^{\mathcal{D}}(X) \overset{\text{def}}{=} \lambda s.\{s' \in \Sigma_\perp \mid \langle s, s'\rangle \in X\}$ $\gamma^{\mathcal{D}}(f) = \{\langle x, y\rangle \mid y \in f(x) \}$
$g\mathcal{W}p = \alpha^{g\mathcal{W}p}(\mathcal{D}^\infty)$	$\langle \Sigma \to \wp(\Sigma_\perp), \sqsubseteq\rangle \underset{\alpha^{g\mathcal{W}p}}{\overset{\gamma^{g\mathcal{W}p}}{\leftrightarrows}} \langle \wp(\Sigma_\perp) \overset{\text{coa}}{\to} \wp(\Sigma), \supseteq\rangle$	$\alpha^{g\mathcal{W}p}(f) = \lambda P.\ \{s \in \Sigma \mid f(s) \subseteq P \}$ $\gamma^{g\mathcal{W}p}(\Phi) = \lambda s\ .\ \{s' \mid s \notin \Phi(\Sigma_\perp \setminus \{s'\}) \}$
$g\mathcal{H} = \alpha^{g\mathcal{H}}(g\mathcal{W}p)$	$\langle \wp(\Sigma_\perp) \overset{\text{coa}}{\to} \wp(\Sigma), \supseteq\rangle \underset{\alpha^{g\mathcal{H}}}{\overset{\gamma^{g\mathcal{H}}}{\leftrightarrows}} \langle \wp(\Sigma) \otimes \wp(\Sigma_\perp), \supseteq\rangle$	$\alpha^{g\mathcal{H}}(\Phi) = \{\langle X, Y\rangle \mid X \subseteq \Phi(Y) \}$ $\gamma^{g\mathcal{H}}(H) = \lambda Y\ .\ \bigcup\{X \mid \langle X, Y\rangle \in H \}$

abstraction of the basic natural trace-based semantics τ^∞. In the following we denote by *Nat* the identical abstraction of the maximal trace semantics.

Relational semantics. The relational semantics $\mathcal{R}^\infty$ associates, with program traces, an input–output relation by using the bottom symbol, $\perp \notin \Sigma$, to denote non-termination. This corresponds to an abstraction of the maximal trace semantics, where intermediate computation states are ignored. The abstraction function $\alpha^{\mathcal{R}}$, that allows to get the relational semantics as abstraction of the maximal trace one, i.e., $\mathcal{R}^\infty = \alpha^{\mathcal{R}}(\tau^\infty)$, is given in Table 1. The corresponding closure is

$$\begin{aligned} \mathit{Rel}(X) \overset{\text{def}}{=} \gamma^{\mathcal{R}}\alpha^{\mathcal{R}}(X) = &\left\{\sigma \in \Sigma^+ \;\middle|\; \exists\, \delta \in X^+ \,.\, \sigma_\vdash = \delta_\vdash \ \wedge\ \sigma_\dashv = \delta_\dashv \right\} \\ &\cup \left\{\sigma \in \Sigma^\omega \;\middle|\; \exists\, \delta \in X^\omega \,.\, \sigma_\vdash = \delta_\vdash \right\} \end{aligned}$$

Denotational semantics. The denotational semantics $\mathcal{D}^\infty$ abstracts away from the history of computations, by considering input–output functions. This semantics is isomorphic to relational semantics. The abstraction function $\alpha^{\mathcal{D}}$, that allows to get the denotational semantics as abstraction of the relational one, i.e., $\mathcal{D}^\infty = \alpha^{\mathcal{D}}(\mathcal{R}^\infty)$, is given in Table 1. The corresponding closure operator on the trace semantics is

$$\begin{aligned} \mathit{Den}(X) &\overset{\text{def}}{=} \gamma^{\mathcal{R}}\gamma^{\mathcal{D}}\alpha^{\mathcal{D}}\alpha^{\mathcal{R}}(X) \\ &= \left\{\sigma \in \Sigma^+ \;\middle|\; \exists\, \delta \in X^+ \,.\, \sigma_\vdash = \delta_\vdash \ \wedge\ \sigma_\dashv = \delta_\dashv \right\} \\ &\quad \cup \left\{\sigma \in \Sigma^\omega \;\middle|\; \exists\, \delta \in X^\omega \,.\, \sigma_\vdash = \delta_\vdash \right\} \end{aligned}$$

Weakest precondition semantics. Dijkstra's predicate transformer $g\mathcal{W}p$ is represented as co-additive functions, denoting weakest-precondition predicate transformers [18]. We consider the program *S*, and a *post-condition* (set of desired final states) *P*, that we want to hold after the execution of *S*. The semantics consists in finding the weakest *precondition*, namely the biggest set of possible initial states, which allows the program to finish in *P*.

Table 2
Observable semantics as abstract interpretations

Semantics	Domain relation	Abstraction and concretization
$\tau^+ = \alpha^+(\tau^\infty)$	$\langle \wp(\Sigma^\infty), \subseteq \rangle \underset{\alpha^+}{\overset{\gamma^+}{\leftrightarrows}} \langle \wp(\Sigma^+), \subseteq \rangle$	$\alpha^+(X) = X \cap \Sigma^+ \stackrel{\text{def}}{=} X^+$ $\gamma^+(Y) = Y \cup \Sigma^\omega$
$\tau^{\hat{0}} = \alpha^{\hat{0}}(\tau^\infty)$	$\langle \wp(\Sigma^\infty), \subseteq \rangle \underset{\alpha^{\hat{0}}}{\overset{\gamma^{\hat{0}}}{\leftrightarrows}} \langle D^{\hat{0}}, \sqsubseteq \rangle$	$\alpha^{\hat{0}}(X) \stackrel{\text{def}}{=} X \cup \bigcup \{ chaos(\sigma_\vdash) \mid \sigma \in X \cap \Sigma^\omega \}$ $\gamma^{\hat{0}}(Y) = Y$
$\tau^\omega = \alpha^\omega(\tau^\infty)$	$\langle \wp(\Sigma^\infty), \subseteq \rangle \underset{\alpha^\omega}{\overset{\gamma^\omega}{\leftrightarrows}} \langle \wp(\Sigma^\omega), \subseteq \rangle$	$\alpha^\omega(X) = X \cap \Sigma^\omega \stackrel{\text{def}}{=} X^\omega$ $\gamma^\omega = X \cup \Sigma^+$

The abstraction function $\alpha^{g\mathcal{W}p}$, that allows to get the weakest precondition semantics as abstraction of the denotational one, i.e., $g\mathcal{W}p = \alpha^{g\mathcal{W}p}(\mathcal{D}^\infty)$, is given in Table 1. The corresponding closure operator on the trace semantics is

$$\begin{aligned} gWp(X) &\stackrel{\text{def}}{=} \gamma^{\mathcal{R}}\gamma^{\mathcal{D}}\gamma^{g\mathcal{W}p}\alpha^{g\mathcal{W}p}\alpha^{\mathcal{D}}\alpha^{\mathcal{R}}(X) \\ &= \{\sigma \in \Sigma^+ \mid \exists\, \delta \in X^+ \,.\, \sigma_\vdash = \delta_\vdash \ \wedge\ \sigma_\dashv = \delta_\dashv \} \\ &\quad \cup \{\sigma \in \Sigma^\omega \mid \exists\, \delta \in X^\omega \,.\, \sigma_\vdash = \delta_\vdash \} \end{aligned}$$

Hoare's axiomatic semantics. Similar to the $g\mathcal{W}p$ semantics, in the Hoare axiomatic semantics we consider triples of the kind $\{Q\}\ S\ \{P\}$, and, in this case, we give semantics to the program S by finding all the pairs $\langle P, Q\rangle$ such that $\{Q\}\ S\ \{P\}$ is a valid Hoare triple [35]. Hoare's axiomatic semantics $g\mathcal{H}$ is represented as elements in tensor product domains, i.e., GCs, specifying the adjoint relation between weakest-precondition and strongest-post-condition in Hoare's triples $\{P\}\ C\ \{Q\}$. The abstraction function $\alpha^{g\mathcal{H}}$, that allows to get the axiomatic semantics as abstraction of the weakest precondition one, i.e., $g\mathcal{H} = \alpha^{g\mathcal{H}}(g\mathcal{W}p)$, is given in Table 1. The corresponding closure operator on the trace semantics is the same as the denotational semantics.

Each semantics in natural style may have a corresponding *angelic*, *demonic*, and *infinite* observable, which is again an abstraction. For each semantics, all the observables are derived as the fix-points, in the computational order, of semantic functions obtained by applying the fix-point transfer theorems [10].

Angelic. The angelic trace semantics τ^+ is designed as an abstraction of the maximal trace semantics, and it is obtained by approximating sets, of possibly finite or infinite traces, with the set of finite traces only, i.e., $\tau^+ = \alpha^+(\tau^\infty)$ (see Table 2).

We denote by $\mathcal{R}^+, \mathcal{D}^+, \mathcal{W}lp$, and $p\mathcal{H}$, respectively, the big-step relational semantics [38], angelic denotational, weakest-liberal precondition [18], and Hoare's partial correctness semantics [35]. All these semantics have been proved, in [9], to be the angelic abstractions of the corresponding semantics in natural style. The basic angelic trace semantics is constructively derived as the least fix-point, in the computational order, of a semantic function: $\tau^+ = lfp^{\subseteq}_{\varnothing} F^+$ where $F^+(X) = \tau^1 \cup \dot{\tau^2}^\frown X$.

Demonic. The demonic trace semantics, denoted as $\tau^{\hat{\partial}}$, is derived from the maximal trace semantics by approximating non-termination by *chaos*, namely by the set of all the possible finite computations starting from the state that leads to non-termination. This corresponds to allowing the worst possible behavior of the program [10,17,18]. This semantics is obtained as abstraction of the natural semantics by using the function $\alpha^{\hat{\partial}}$, i.e., $\tau^{\hat{\partial}} = \alpha^{\hat{\partial}}(\tau^{\infty})$ (see Table 2). In this way, the demonic observable is defined on the domain $D^{\hat{\partial}} = \alpha^{\hat{\partial}}(\wp(\Sigma^{\infty}))$, which is such that $X \in D^{\hat{\partial}}$ if and only if

$$\sigma \in X^{\omega} \;\Rightarrow\; chaos(\sigma_{\vdash}) \subseteq X^{+}$$

where $chaos(\sigma_{\vdash}) \stackrel{\text{def}}{=} \left\{ \delta \in \Sigma^{+} \;\middle|\; \delta_{\vdash} = \sigma_{\vdash} \right\}$.

We denote by $\mathcal{R}^{\hat{\partial}}$, $\mathcal{D}^{\hat{\partial}}$, $\mathcal{W}p^{\hat{\partial}}$, and $g\mathcal{H}^{\hat{\partial}}$ the demonic relational, demonic denotational [3], demonic weakest-precondition and demonic Hoare's semantics. These semantics have been proved, in [9], to be the demonic abstractions of the corresponding semantics in natural style. The basic demonic trace semantics is constructively derived as the least fix-point, in the computational order, of a semantic function: $\tau^{\hat{\partial}} = lfp^{\stackrel{\sqsubseteq}{=}^{\hat{\partial}}}_{\Sigma^{\infty}} F^{\hat{\partial}}$ where $F^{\hat{\partial}}(X) = \tau^{1} \cup \tau^{\dot{2}} \frown X$.

Infinite. The infinite trace semantics, denoted τ^{ω}, is derived by observing non-terminating traces only, i.e., $\tau^{\omega} = \alpha^{\omega}(\tau^{\infty})$ (see Table 2). The corresponding infinite semantics are denoted by $\mathcal{R}^{\omega}$, $\mathcal{D}^{\omega}$, $\mathcal{W}p^{\omega}$, and $g\mathcal{H}^{\omega}$. The basic infinite trace semantics is constructively derived as the greatest fix-point, in the computational order, of a semantic function: $\tau^{\omega} = gfp^{\subseteq}_{\Sigma^{\omega}} F^{\omega}$ where $F^{\omega}(X) = \tau^{\dot{2}} \frown X$.

Weakest precondition. The weakest precondition semantics for total correctness $\mathcal{W}p$, is modeled as a further abstraction of the natural trace semantics. This semantics considers only those computations that surely terminate, in other words, the weakest precondition is the largest set of initial states terminating in the given post-condition. This observable is obtained as abstraction of the $g\mathcal{W}p$ semantics: $\mathcal{W}p = \alpha^{\mathcal{W}p}(g\mathcal{W}p)$ where

$$\alpha^{\mathcal{W}p}(\Phi) = \Phi\,|_{\wp(\Sigma)}$$

$$\gamma^{\mathcal{W}p}(\Psi) = \lambda P\,.\,(\text{if } \bot \notin P \text{ then } \Psi(P) \text{ else } \varnothing)$$

and $\left\langle (\wp(\Sigma_{\bot}) \stackrel{\text{coa}}{\longrightarrow} \wp(\Sigma)), \supseteq \right\rangle \underset{\alpha^{\mathcal{W}p}}{\overset{\gamma^{\mathcal{W}p}}{\leftrightarrows}} \left\langle (\wp(\Sigma) \stackrel{\text{coa}}{\longrightarrow} \wp(\Sigma)), \supseteq \right\rangle$.

The semantics $t\mathcal{H}$ is the Hoare's axiomatic abstraction of $\mathcal{W}p$, i.e., $t\mathcal{H} = \alpha^{g\mathcal{H}}(\mathcal{W}p)$.

The whole hierarchy, relating semantics styles and observables, is shown in Fig. 1, where lines and arrows denote, respectively, isomorphisms and strict abstractions between semantics.

In the following sections, we characterize the properties of the semantics in Cousot's hierarchy, in terms of the basic operations that compose and decompose abstract domains. We consider, first, the attribute independent composition of semantics, which is provided by the reduced product operation. This operation, and its inverse, which is domain complementation, provides a formal method for isolating complementary and independent observables in well known semantics of programming languages. Afterwards, we consider the relational combinator of domains, which is reduced relative power. This provides a characterization of compositionality of semantics, which is parametric on the observation made. The result is an algebra of semantics, where both, concrete and abstract semantics for program analysis, can be obtained as solutions of the same domain equations, involving reduced product,

reduced power and domain complementation. These equations are parametric on the chosen observable property.

4. Independent composition and decomposition of semantics

Let *Program* be the collection of all well-formed programs in a programming language. Let C be a domain of semantic denotations, e.g. execution traces, functions, sets of states, etc., and $[\![\cdot]\!] : \mathit{Program} \to C$ is the semantics assigning, with each program $P \in \mathit{Program}$, its meaning in C. For any $\rho \in \mathit{uco}(C)$, we define the abstract semantics function $[\![\cdot]\!]_\rho : \mathit{Program} \to \rho(C)$, as $[\![P]\!]_\rho \stackrel{\text{def}}{=} \rho([\![P]\!])$. The following result [28] formally expresses the intuition that the reduced product semantics corresponds precisely to the logical conjunction of the observables associated with each semantics in the product.

Theorem 4.1. *If $P, Q \in \mathit{Program}$ and $\{A_i\}_{i\in I} \subseteq \mathit{uco}(C)$ then $[\![P]\!]_{\sqcap_{i\in I} A_i} = [\![Q]\!]_{\sqcap_{i\in I} A_i}$ iff $\forall i \in I.\ [\![P]\!]_{A_i} = [\![Q]\!]_{A_i}$.*

Proof. Consider $P, Q \in \mathit{Program}$.

($\Rightarrow$) Assume that $k \in I$. Since $\sqcap_{i\in I}\rho_i \sqsubseteq \rho_k$, then we have $[\![P]\!]_{\rho_k} = [\![Q]\!]_{\rho_k}$, as desired.

($\Leftarrow$) Since $\forall i \in I.\ \rho_i([\![P]\!]) = \rho_i([\![Q]\!])$, we have $\wedge_{i\in I}\rho_i([\![P]\!]) = \wedge_{i\in I}\rho_i([\![Q]\!])$ which proves the thesis. □

A sequence of abstract domains $\{A_i\}_{i\in I}$ is a (conjunctive) decomposition of the abstract domain B, if $B = \sqcap_{i\in I} A_i$. In this context, we can characterize the independent observables contained in a given semantics, by identifying the most abstract observables that, composed with each other, gives back the semantics. This allows to find the most abstract decomposition of a semantics, as regards a given observable. In this way, we are able to identify the complementary observables contained in a semantics.

4.1. Domain complementation

Abstract domain complementation, introduced in [7], provides a systematic method for decomposing abstract domains. Complementation is the *inverse* operation of the reduced product (see [29]) in the sense that, starting from any two domains $C \sqsubseteq D$, it gives, as result, the most abstract domain $C \ominus D$, whose reduced product with D is exactly C (i.e., $(C \ominus D) \sqcap D = C$). By the equivalence between closure operators and abstract domains, the above notion of complementation corresponds precisely to *pseudo-complementation* on closures. In particular the complement described above is the pseudo-complement of the closure ρ_D, corresponding to D, in $\mathit{uco}(C)$. Recall that, if L is a meet-semilattice with bottom, then the *pseudo-complement* of $x \in L$, when it exists, is the unique element $x^* \in L$ such that $x \wedge x^* = \bot$ and such that $\forall y \in L.\ (x \wedge y = \bot) \Rightarrow (y \leqslant x^*)$ [4]. In a complete lattice L, if x^* exists, then $x^* = \vee\{y \in L \mid x \wedge y = \bot\}$. If every $x \in L$ has the pseudo-complement, L is *pseudo-complemented*. It is worth noting that pseudo-complementation is the only possible form of complementation in abstract interpretation. Indeed, it is well known [20,36] that $\mathit{uco}(C)$ is complemented (in the standard sense) iff C is a complete well-ordered chain, and this is a far too restrictive hypothesis for semantic domains. The

following results [22,27] provide two sufficient conditions, on C, that make $uco(C)$ pseudo-complemented. Recall that a complete lattice C is *meet-continuous* if for any chain $Y \subseteq C$ and for any $x \in C$, $x \wedge (\vee Y) = \vee_{y \in Y}(x \wedge y)$. Moreover C is meet-generated, by $S \subseteq C$, if $C = \mathcal{M}(S)$.

Theorem 4.2. *Let C be a complete lattice.*
1. *If C is a meet-continuous then $uco(C)$ is pseudo-complemented* [27].
2. *If C is meet-generated by $Mirr(C)$ then $uco(C)$ is pseudo-complemented and, for any $A \in uco(C)$, we have $A^* \stackrel{\text{def}}{=} C \ominus A = \mathcal{M}(Mirr(C) \setminus A)$* [22].

Note that for any $A, B \in uco(C)$ such that $A \sqsubseteq B$, $A \ominus B$ is well defined if $\uparrow A = uco(A)$ is pseudo-complemented.

4.2. Decomposing trace-based semantics

Domain complementation is the standard operation used for factorizing semantics. Given any two semantics $X, A \in uco(C)$, such that $X \sqsubseteq A$, the complement semantics $[\![\cdot]\!]_{(X \ominus A)}$, is the most abstract semantics such that $[\![P]\!]_X = [\![Q]\!]_X$ iff $[\![P]\!]_A = [\![Q]\!]_A$ and $[\![P]\!]_{(X \ominus A)} = [\![Q]\!]_{(X \ominus A)}$. In practice, it is always possible to define complements of semantics, since the hypotheses in Theorem 4.2, assuring their existence, are extremely weak. In most cases, in fact, the domain of abstract interpretations is a continuous, or even algebraic lattice, or it is generated by its meet-irreducible elements.

In this section, we prove that angelic and demonic semantics provide a conjunctive decomposition of natural semantics, and that angelic and infinite semantics form a minimal (most abstract) decomposition of natural semantics. This is proved for the basic operational trace-based semantics only, which represents the bottom (most concrete) semantics in the Cousot's hierarchy (see Fig. 1). We will generalize this construction to the whole hierarchy, in Section 4.3.

Consider the angelic, demonic, and infinite closure operators on maximal traces, i.e., $Ang, Dem, Inf \in uco(\wp(\Sigma^\infty), \subseteq)$, induced by, respectively, the angelic, demonic, and infinite abstractions on the trace semantics: $Ang \stackrel{\text{def}}{=} \gamma^+ \circ \alpha^+$, $Dem \stackrel{\text{def}}{=} \gamma^{\hat{0}} \circ \alpha^{\hat{0}}$, and $Inf \stackrel{\text{def}}{=} \gamma^\omega \circ \alpha^\omega$ (see Table 2). It is immediate to observe that for any $X \in \Sigma^\infty$:

$$\begin{aligned} Nat(X) &= X \\ Ang(X) &= X \cup \Sigma^\omega \\ Dem(X) &= X \cup \bigcup \left\{ chaos(\delta_\vdash) \;\middle|\; \delta \in X^\omega \right\} \\ Inf(X) &= X \cup \Sigma^+ \end{aligned}$$

In order to prove that infinite and angelic semantics factorize the natural semantics, we have to characterize the meet-irreducible elements of the domains involved. The semantics τ^∞ is defined on the domain $\wp(\Sigma^\infty)$, whose meet-irreducibles are $\Sigma^\infty \setminus \{\sigma\}$, for each $\sigma \in \Sigma^\infty$. The semantics τ^+ is defined on $\wp(\Sigma^+)$ whose meet-irreducible elements are $\Sigma^+ \setminus \{\sigma\}$, for each $\sigma \in \Sigma^+$. Finally the semantics Σ^ω is defined on $\wp(\Sigma^\omega)$, whose meet-irreducibles are $\Sigma^\omega \setminus \{\sigma\}$, for each $\sigma \in \Sigma^\omega$.

Lemma 4.3. *Let $\sigma \in \Sigma^+$ and $\delta \in \Sigma^\omega$. Then we have*

$$\Sigma^\infty \setminus \{\sigma\} \in \mathit{Ang}, \qquad \Sigma^\infty \setminus \{\delta\} \notin \mathit{Ang}$$
$$\Sigma^\infty \setminus \{\sigma\} \notin \mathit{Dem}, \qquad \Sigma^\infty \setminus \{\delta\} \in \mathit{Dem}$$

Proof. It is immediate, by definition of *Ang*, that if $\sigma \in \Sigma^+$ and $\delta \in \Sigma^\omega$ then $\mathit{Ang}(\Sigma^\infty \setminus \{\sigma\}) = (\Sigma^\infty \setminus \{\sigma\}) \cup \Sigma^\omega = \Sigma^\infty \setminus \{\sigma\}$, while we have that $\mathit{Ang}(\Sigma^\infty \setminus \{\delta\}) = (\Sigma^\infty \setminus \{\delta\}) \cup \Sigma^\omega = \Sigma^\infty$.

An analogous reasoning can be done for the demonic semantics *Dem*, indeed we can note that $\mathit{Dem}(\Sigma^\infty \setminus \{\sigma\}) = \Sigma^\infty \setminus \{\sigma\} \cup \left\{\sigma' \in \Sigma^+ \;\middle|\; \exists\, \beta \in \Sigma^\omega \,.\, \sigma'_\vdash = \beta_\vdash \right\} = \Sigma^\infty$. On the other hand, $\mathit{Dem}(\Sigma^\infty \setminus \{\delta\}) = \Sigma^\infty \setminus \{\delta\} \cup \left\{\sigma' \in \Sigma^+ \;\middle|\; \exists\, \beta \in \Sigma^\omega \,.\, \sigma'_\vdash = \beta_\vdash \right\} = \Sigma^\infty \setminus \{\delta\}$. □

The angelic semantics *Ang* factorizes the maximal trace semantics together with *Inf* and *Dem*. Moreover, the angelic semantics does not share information with both, the infinite and the demonic semantics.

Proposition 4.4. *$\mathit{Nat} \ominus \mathit{Ang} = \mathit{Inf}$, $\mathit{Nat} \ominus \mathit{Inf} = \mathit{Ang}$, $\mathit{Nat} \ominus \mathit{Dem} = \mathit{Ang}$, $\mathit{Ang} \sqcup \mathit{Dem} = \Sigma^\infty$ and $\mathit{Ang} \sqcup \mathit{Inf} = \Sigma^\infty$.*

Proof. We know that $\mathit{Nat} \ominus \mathit{Ang} = \mathcal{M}(\mathit{Mirr}(\wp(\Sigma^\infty)) \setminus \mathit{Ang})$, namely $\mathit{Mirr}(\mathit{Nat} \ominus \mathit{Ang}) = \mathit{Mirr}(\wp(\Sigma^\infty)) \setminus \mathit{Ang}$. Since $\mathit{Mirr}(\wp(\Sigma^\infty))$ is the set of all the elements of the kind $\Sigma^\infty \setminus \{\delta\}$, with $\delta \in \Sigma^\infty$, we have that $X \in \mathit{Mirr}(\mathit{Nat} \ominus \mathit{Ang})$ iff $X = \Sigma^\infty \setminus \{\delta\}$ with $\delta \in \Sigma^\omega$, by Lemma 4.3. At this point, since $X = \Sigma^\omega \setminus \{\delta\}$ is a meet-irreducible element in $\wp(\Sigma^\omega)$, then it is immediate to verify that $\Sigma^+ \cup (\Sigma^\omega \setminus \{\delta\})$ is meet-irreducible in $\left\{\Sigma^+ \cup X \;\middle|\; X \in \wp(\Sigma^\omega)\right\}$, the set of the fix-points of *Inf*. Hence, we can conclude that $\mathit{Mirr}(\mathit{Nat} \ominus \mathit{Ang}) = \mathit{Mirr}(\mathit{Inf})$, i.e., $\mathit{Nat} \ominus \mathit{Ang} = \mathit{Inf}$. The proof for $\mathit{Nat} \ominus \mathit{Inf} = \mathit{Ang}$ is analogous. The proof for $\mathit{Nat} \ominus \mathit{Dem} = \mathit{Ang}$ and $\mathit{Ang} \sqcup \mathit{Dem} = \Sigma^\infty$ is immediate by Theorem 4.2 and Lemma 4.3. □

It is worth noting that the angelic and demonic abstractions do not factorize natural semantics in most abstract factors. In fact, while the complement of demonic semantics is angelic, the converse does not hold. It is worth noting that $\mathit{Nat} \ominus \mathit{Ang} \neq \mathit{Dem}$. In particular, for any finite trace $\sigma \in \Sigma^+$, $\mathit{Dem}(\{\sigma\}) = \{\sigma\}$, while $\mathit{Nat} \ominus \mathit{Ang}(\{\sigma\}) = \Sigma^+ \supset \{\sigma\}$. In order to provide an example of the relationship between the angelic and infinite observables, we represent sets of traces by the pair of their initial (input) and final (output) states, $\perp$ for infinite traces. This corresponds to mapping, the factorization given above, on the relational semantics. In Figs. 2 and 5, we can see a representation of the relational angelic and infinite semantics on the alphabet $\Sigma = \{a, b\}$.

4.3. Decomposing the hierarchy

In this section, we characterize the symmetric structure of Cousot's hierarchy of semantics, in terms of a general algebraic property of closure operators. Indeed, note that the angelic/demonic/infinite observables are abstractions of the natural semantics, in any style (trace-based, relational, denotational, predicate transformer and axiomatic), since the abstractions, that relates the different styles of semantics and observables, commute all over Cousot's hierarchy.

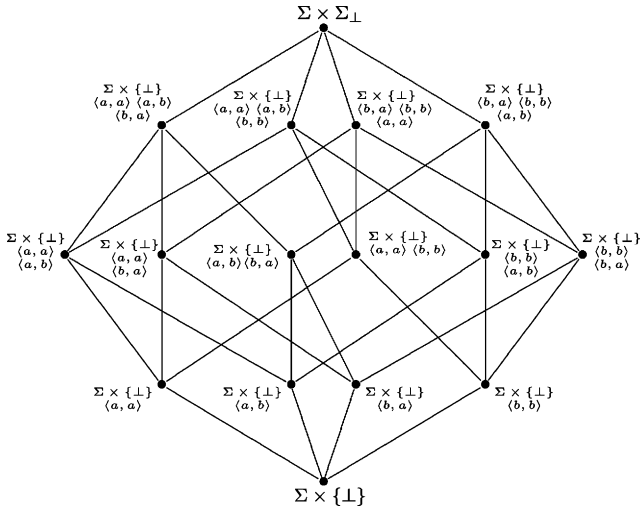

Fig. 2. Relational angelic semantics on $\Sigma = \{a, b\}$.

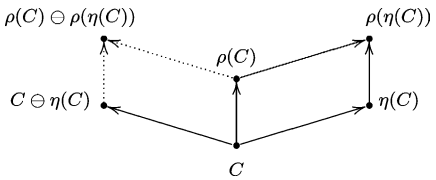

Fig. 3. Basic abstraction structure.

In order to understand this situation better, we consider a concrete semantic domain C, generated by its meet-irreducibles, and two closure operators $\rho, \eta \in uco(C)$. Recall that $\rho \circ \eta \in uco(C)$ iff $\rho \circ \eta = \eta \circ \rho = \eta \sqcup \rho$ [36]. Namely, the composition of two closures is a closure if and only if they commute. Consider the structure in Fig. 3. It is perfectly symmetric if the closure $\rho(C) \ominus \rho(\eta(C))$ corresponds precisely to the abstraction, by ρ, of the complementary closure $C \ominus \eta(C)$, namely if the closure ρ is an $\ominus$-morphism: $\rho(C \ominus \eta(C)) = \rho(C) \ominus \rho(\eta(C))$ or, equivalently, whenever $(\rho \sqcup \eta)^*$, computed in $uco(\rho(C))$, is the same closure as $\rho \sqcup \eta^*$.

Our aim is that of finding sufficient conditions that guarantee that a closure distributes on the complement operation, and of proving that all the closures in the Cousot's hierarchy satisfy these conditions. This would mean that all the symmetric semantics in the hierarchy are complements, as it happens in Fig. 3.

Unfortunately, not all abstractions, viz. closures, commute with respect to complementation. The following example shows this situation.

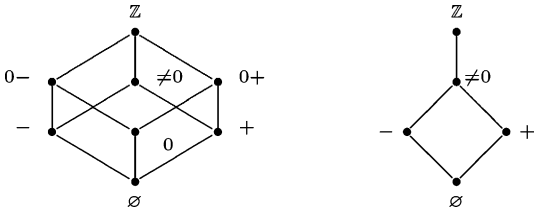

Fig. 4. The *Sign* and *Nzero* domains.

Example 4.5. Let *Sign* be the domain, for sign analysis of integer variables, represented in Fig. 4, and let $\rho(Sign) = \{\mathbb{Z}, 0-, -\}$. If $\eta(Sign) = \{\mathbb{Z}, 0-, 0\}$, then $\eta\rho(Sign) = \rho\eta(Sign) = \rho(Sign) \cap \eta(Sign)$ for definition of $\sqcup$, so $\rho(\eta(Sign)) = \{\mathbb{Z}, 0-\}$.

Then we can verify that $\rho(Sign \ominus \eta(Sign)) = \rho(\{\mathbb{Z}, 0+, \neq 0, +\}) = \{\mathbb{Z}\}$ while we have $\rho(Sign) \ominus \rho(\eta(Sign)) = \{\mathbb{Z}, -\}$.

Lemma 4.6. *Let $\rho, \eta \in uco(C)$. If $\rho \circ \eta = \eta \circ \rho$ and $\rho \circ \eta^* = \eta^* \circ \rho$ then $(\rho \sqcup \eta^*) \sqsubseteq \rho \ominus (\rho \sqcup \eta)$.*

Proof. It is well known that $\rho \circ \eta \in uco(C)$ if and only if $\rho \circ \eta = \eta \circ \rho = \rho \sqcup \eta$ [36]. Moreover, $\rho \sqsubseteq (\rho \sqcup \eta) \sqcap (\rho \sqcup \eta^*)$, since, both $\rho \sqcup \eta^*$ and $\rho \sqcup \eta$, by definition of *lub* $\sqcup$, are closures more abstract than ρ, and therefore also their reduced product is more abstract than ρ. On the other hand, by definition of domain complementation, we have that $\eta \sqcap \eta^* = C$, so for each $x \in C$ we have $\eta(x) \wedge \eta^*(x) = x$. Therefore, if $x \in \rho(C) \subseteq C$ we have

$$\begin{aligned} x &= \eta(x) \wedge \eta^*(x) \\ &= \eta(\rho(x)) \wedge \eta^*(\rho(x)) \\ &= (\rho \sqcup \eta)(x) \wedge (\rho \sqcup \eta^*)(x) \end{aligned}$$

Hence by definition of reduced product we have $\rho = (\rho \sqcup \eta) \sqcap (\rho \sqcup \eta^*)$, and, by definition of pseudo-complementation in $uco(\rho(C))$, we know that $\rho \ominus (\rho \sqcup \eta)$ is the most abstract closure whose reduced product with $\rho \sqcup \eta$ returns ρ. Therefore, since the reduced product between $\rho \sqcup \eta^*$ and $\rho \sqcup \eta$ is ρ, we can conclude that $\rho \sqcup \eta^*$ is more concrete than its complement, namely $\rho \sqcup \eta^* \sqsubseteq \rho \ominus (\rho \sqcup \eta)$. □

This lemma tells us that one of the inclusions, implicit in the equality, holds under certain hypotheses. Next lemma, instead, gives a sufficient condition for the other inclusion.

Lemma 4.7. *Let $\rho, \eta \in uco(C)$. If $\eta \sqcup \eta^* = \{\top\}$ and $Mirr(\rho(C) \cap \eta^*(C)) \subseteq Mirr(\rho(C))$ then $\rho \ominus (\rho \sqcup \eta) \sqsubseteq (\rho \sqcup \eta^*)$.*

Proof. By hypothesis, we have that $\eta \sqcup \eta^* = \{\top\}$, namely $\eta(C) \cap \eta^*(C) = \{\top\}$. On the other hand, we have that $\rho(C) \cap \eta^*(C) \subseteq \eta^*(C)$, therefore the conjunction of these two facts implies that $(\rho(C) \cap \eta^*(C)) \cap \eta(C) = \{\top\}$. This means that $Mirr(\rho(C) \cap \eta^*(C)) \cap \eta(C) = \varnothing$ (1). Moreover, by hypothesis, we have $Mirr(\rho(C) \cap \eta^*(C)) \subseteq Mirr(\rho(C))$ then the

following relations hold:

$$\begin{aligned}
&Mirr(\rho \ominus (\rho \sqcup \eta)) \text{ (by Theorem 4.2)}\\
&\quad = Mirr(\rho(C)) \setminus (\rho(C) \cap \eta(C)) \text{ (by hypothesis)}\\
&\quad \supseteq Mirr(\rho(C) \cap \eta^*(C)) \setminus (\rho(C) \cap \eta(C)) (*)\\
&\quad = Mirr(\rho(C) \cap \eta^*(C))\\
&\quad = Mirr(\rho \sqcup \eta^*)
\end{aligned}$$

where the step $(*)$ holds because, if $x \in Mirr(\rho(C) \cap \eta^*(C))$, then, by condition (1), we have $x \notin \eta(C)$, and so $x \notin \rho(C) \cap \eta(C)$.

Since $(\rho \ominus (\rho \sqcup \eta))(C) \supseteq (\rho \sqcup \eta^*)(C)$, we can conclude that $\rho \ominus (\rho \sqcup \eta) \sqsubseteq \rho \sqcup \eta^*$. □

By Theorem 4.2 we know that the complement of a closure depends on the meet-irreducible elements of the concrete domain. For this reason it seems sufficient that a closure transforms meet-irreducibles into meet-irreducibles for making the closure commuting with respect to domain complementation. But it is immediate to note that the structure of meet-irreducible elements is not always left unchanged by an abstraction. Indeed an abstraction erases elements from the concrete domain and nothing prevents it from eliminating meet-irreducibles, too. Moreover, the abstraction can also create new meet-irreducible elements. Indeed, if we extract a chain from a more complex domain (such as *Sign*) all its elements become meet-irreducible. This is a consequence of the structure of the abstract domains, which are Moore families. Indeed $\rho(Mirr(C)) \subseteq Mirr(\rho(C))$, but the inverse inclusion generally does not hold, as we can see in Example 4.5 where the element $-$ is meet-irreducible in $\rho(Sign)$ but it is not the image, as regards ρ, of any meet-irreducible element of *Sign*.

These observations lead to the following theorem. This theorem provides two independent sufficient conditions for making ρ commuting with $\ominus$.

Theorem 4.8. *Let $\rho, \eta \in uco(C)$ such that $\rho \circ \eta = \eta \circ \rho$, $\eta \sqcup \eta^* = \{\top\}$ and $\rho \circ \eta^* = \eta^* \circ \rho$.*
(i) *If $Mirr(\rho(C) \cap \eta^*(C)) \subseteq Mirr(\rho(C))$ then $\rho(C \ominus \eta(C)) = \rho(C) \ominus \rho(\eta(C))$.*
(ii) *If $Mirr(\rho(C)) = \rho(Mirr(C))$ then $\rho(C \ominus \eta(C)) = \rho(C) \ominus \rho(\eta(C))$.*

Proof. (i) Immediate by Lemmas 4.6 and 4.7.

(ii) We have to prove that $\rho(C \ominus \eta(C)) = \rho(C) \ominus \rho(\eta(C))$. First of all, we prove that $\rho(Mirr(C)) \setminus \rho(\eta(C)) = \rho(Mirr(C) \setminus \eta(C))$. We know that, in general, for each map f, we have $f(A) \setminus f(B) \subseteq f(A \setminus B)$, hence the inclusion $\subseteq$ holds. Now, we prove that also the other inclusion holds. In order to show this, we prove that each element in the set $\rho(Mirr(C) \setminus \eta(C))$ belongs to $\rho(Mirr(C))$ and does not belong to $\rho\eta(C)$. We know that $Mirr(C) \setminus \eta(C) \subseteq Mirr(C)$, hence, by monotonicity of ρ, we obtain that $\rho(Mirr(C) \setminus \eta(C)) \subseteq \rho(Mirr(C))$. Moreover, since $Mirr(\rho(C)) = \rho(Mirr(C))$, and since, by definition, $\top \notin Mirr(\rho(C))$, we have that $\top \notin \rho(Mirr(C))$ which implies $\top \notin \rho(Mirr(C) \setminus \eta(C))$. Finally, we have to prove that

$$x \in \rho(Mirr(C) \setminus \eta(C)) \;\Rightarrow\; x \notin \rho(\eta(C))$$

where $x \neq \top$ for the consideration above. Note that, if an element belongs to the composition of two closures that commute, then it belongs to each closure. Moreover $\eta \sqcup \eta^* = \{\top\}$,

therefore the following implications hold:

$$\begin{aligned} x \in \rho(Mirr(C) \setminus \eta(C)) &\Rightarrow x \in \rho\eta^*(C) \\ &\Rightarrow x \in \rho(C) \ \wedge \ x \in \eta^*(C) \\ &\Rightarrow x \notin \rho\eta(C) \end{aligned}$$

Since $\rho(Mirr(C) \setminus \eta(C)) \subseteq \rho(Mirr(C))$, we have:

$$\rho(Mirr(C) \setminus \eta(C)) \subseteq \rho(Mirr(C)) \setminus \rho(\eta(C))$$

Hence, we have the equality. By the argument above, this implies the following equalities:

$$\begin{aligned} Mirr(\rho(C) \ominus \rho(\eta(C))) &= Mirr(\rho(C)) \setminus \rho(\eta(C)) \text{ (by Theorem 4.2)} \\ &= \rho(Mirr(C)) \setminus \rho(\eta(C)) \text{ (by hypothesis)} \\ &= \rho(Mirr(C) \setminus \eta(C)) \end{aligned}$$

from which the thesis follows. □

The following example shows that the converse of Theorem 4.8-(ii) does not, in general, hold.

Example 4.9. Let *Sign* and $\rho(Sign) = Nzero$ be the domains represented in Fig. 4, for sign and non-zero analysis of integer variables. If $\eta(Sign) = \{\mathbb{Z}, 0-, -\}$, then $\rho(\eta(Sign)) = \{\mathbb{Z}, -\}$.

In this case, we have that $\rho(Sign \ominus \eta(Sign)) = \rho(\{\mathbb{Z}, 0+, \neq 0, +\}) = \{\mathbb{Z}, \neq 0, +\}$. Moreover, it is simple to verify that $\rho(Sign) \ominus \rho(\eta(Sign)) = Nzero \ominus \{\mathbb{Z}, -\} = \{\mathbb{Z}, \neq 0, +\}$, while we have $\rho(Mirr(Sign)) = \{\neq 0\}$ and $Mirr(\rho(Sign)) = Mirr(Nzero) = \{\neq 0, +, -\}$.

4.4. Symmetric abstractions in the hierarchy

In this section, we extend the complementary relation, existing among the angelic, demonic, and infinite observables on maximal traces of a transition system, all over the hierarchy. We use Theorem 4.8 to obtain this generalization. In particular, we prove that the domain complementation commutes with respect to all the abstractions connecting the different semantics styles. Let us consider the relational abstraction *Rel* (see Section 3 and [10]):

$$\begin{aligned} Rel(X) = &\left\{\sigma \in \Sigma^+ \;\middle|\; \exists\, \delta \in X^+ \,.\, \sigma_\vdash = \delta_\vdash \ \wedge \ \sigma_\dashv = \delta_\dashv \right\} \\ &\cup \left\{\sigma \in \Sigma^\omega \;\middle|\; \exists\, \delta \in X^\omega \,.\, \sigma_\vdash = \delta_\vdash \right\} \end{aligned}$$

Lemma 4.10. *Let* $\rho, \eta \in uco(C)$. *If* $\rho \circ \eta = \eta \circ \rho$ *and* $\eta \circ \eta^* = \eta \sqcup \eta^* = \{\top\}$ *then* $\rho \circ \eta^* = \eta^* \circ \rho$.

Proof. For any $x \in C$ we have:

$$\begin{aligned} \rho(\eta^*(x)) &= \eta(\rho(\eta^*(x))) \wedge \eta^*(\rho(\eta^*(x))) \\ &\qquad \text{(by definition of pseudo-complement)} \\ &= \rho(\eta(\eta^*(x))) \wedge \eta^*(\rho(\eta^*(x))) \text{ (by hypothesis)} \\ &= \top \wedge \eta^*(\rho(\eta^*(x))) \text{ (by hypothesis)} \\ &= \eta^*(\rho(\eta^*(x))) \end{aligned}$$

These relations say that $\rho(\eta^*(x)) = \rho(\eta^*(\rho(\eta^*(x))))$ because ρ is a closure, namely it is idempotent. Moreover, we know that ρ and η^* are upper closures, namely they are extensive, which means that $x \leqslant \eta^*(x)$; by monotonicity of ρ we have also that $\rho(x) \leqslant \rho(\eta^*(x))$ and therefore, by the extensivity of ρ, we have $x \leqslant \rho(\eta^*(x))$, namely also $\rho\eta^*$ is extensive. Finally we know that the composition of monotone maps is monotone. Then $\rho \circ \eta^*$ is idempotent, extensive and monotone, namely $\rho \circ \eta^* \in uco(C)$, which holds if and only if $\rho \circ \eta^* = \eta^* \circ \rho$. □

Proposition 4.11. 1. $Rel \circ Ang = Ang \circ Rel$.
2. $Rel \circ Dem = Dem \circ Rel$.
3. $Rel \circ Inf = Inf \circ Rel$.

Proof. It is easy to prove that *Rel* is additive. Therefore

$$\begin{aligned} Rel(Ang(X)) &= Rel(X \cup \Sigma^\omega) \\ &= Rel(X) \cup Rel(\Sigma^\omega) \\ &= Rel(X) \cup \Sigma^\omega \\ &= Ang(Rel(X)) \end{aligned}$$

Let us prove that $Rel \circ Dem = Dem \circ Rel$. By definition and additivity of *Rel*, we have $Rel(Dem(X)) = Rel(X) \cup Rel(\{\sigma \in \Sigma^+ | \delta \in X \cap \Sigma^\omega, \delta_\vdash = \sigma_\vdash\})$ and $Dem(Rel(X)) = Rel(X) \cup \{\sigma \in \Sigma^+ | \delta \in Rel(X) \cap \Sigma^\omega, \delta_\vdash = \sigma_\vdash\}$. Note that $\theta \in Rel(\{\sigma \in \Sigma^+ | \delta \in X \cap \Sigma^\omega, \delta_\vdash = \sigma_\vdash\})$ if and only if $\theta \in \Sigma^+$, and there exist $\sigma \in \Sigma^+$ and $\delta \in X \cap \Sigma^\omega$ such that $\delta_\vdash = \sigma_\vdash$, $\sigma_\vdash = \theta_\vdash$ and $\theta_\dashv = \sigma_\dashv$. Therefore $\theta \in \{\sigma \in \Sigma^+ | \delta \in X \cap \Sigma^\omega, \delta_\vdash = \sigma_\vdash\}$. This implies that $Rel(\{\sigma \in \Sigma^+ | \delta \in X \cap \Sigma^\omega, \delta_\vdash = \sigma_\vdash\}) = \{\sigma \in \Sigma^+ | \delta \in X \cap \Sigma^\omega, \delta_\vdash = \sigma_\vdash\}$. Moreover if θ is such that $\delta \in Rel(X) \cap \Sigma^\omega$ and $\delta_\vdash = \theta_\vdash$, then by definition of *Rel*, there exists $\delta' \in X \cap \Sigma^\omega$ such that $\delta_\vdash = \delta'_\vdash = \theta_\vdash$. This implies that $\theta \in Rel(\{\sigma \in \Sigma^+ | \delta \in X \cap \Sigma^\omega, \delta_\vdash = \sigma_\vdash\})$ if and only if $\theta \in \{\sigma \in \Sigma^+ | \delta \in Rel(X) \cap \Sigma^\omega, \delta_\vdash = \sigma_\vdash\}$. The third equality of the proposition holds because we proved that $Rel \circ Ang = Ang \circ Rel$, moreover by Proposition 4.4 we know that $Ang^* = Inf$ and $Ang \sqcup Inf = \Sigma^\infty$. Moreover, it is worth noting that $Ang \circ Inf = Inf \circ Ang = \Sigma^\infty$. Hence, by Lemma 4.10, we have $Rel \circ Inf = Inf \circ Rel$. □

Proposition 4.11 tells us that, for each possible closure $\eta \in \{Ang, Inf, Dem\}$, we have $Rel \circ \eta = \eta \circ Rel$ and $Rel \circ \eta^* = \eta^* \circ Rel$. Moreover, it is worth noting that the relational semantics does not factorize the trace-based one, being too abstract, i.e., $Nat \ominus Rel = Nat$. This does not allow us to use Theorem 4.8-(ii) for relating the observables in the trace-based and relational semantic styles in the Cousot's hierarchy. Instead, we can use Theorem 4.8-(i), as shown below.

Proposition 4.12. 1. $Mirr(Rel(\wp(\Sigma^\infty)) \cap Ang(\wp(\Sigma^\infty))) \subseteq Mirr(Rel(\wp(\Sigma^\infty)))$.
2. $Mirr(Rel(\wp(\Sigma^\infty)) \cap Inf(\wp(\Sigma^\infty))) \subseteq Mirr(Rel(\wp(\Sigma^\infty)))$.

Proof. We know that *Rel* and *Ang* commute by the previous proposition, so it is immediate to see that the intersection between *Rel* and *Ang* is exactly the angelic relational closure,

that we denote $Ang^{\mathcal{R}}$.

$$Mirr(Rel(\wp(\Sigma^\infty)) \cap Ang(\wp(\Sigma^\infty))) = Mirr(Ang^{\mathcal{R}}(\wp(\Sigma^\infty))))$$

Let X be meet-irreducible in $Ang^{\mathcal{R}}$. Then, it contains all the infinite traces and, moreover, there exist $s, s' \in \Sigma$ such that $\left\{\sigma \in \Sigma^+ \,\middle|\, \sigma_\vdash = s \wedge \sigma_\dashv = s'\right\} \cap X = \varnothing$ (these meet-irreducibles are the concretizations, as regards $\gamma^{\mathcal{R}}$, of the meet-irreducible elements of $\wp(\Sigma \times \Sigma)$).

We have to prove that X is meet-irreducible in Rel. Suppose that there exist $A, B \in Rel$ such that $A \cap B = X$, with $A \neq X$ and $B \neq X$. Since X is meet-irreducible in $Ang^{\mathcal{R}}$, either A or B is in $Ang^{\mathcal{R}}$. Suppose A in $Ang^{\mathcal{R}}$. This means that A does not contain all the infinite traces, which implies that it cannot generate, by intersection, all the sets containing all the infinite traces. Hence, if X is meet-irreducible in $Ang^{\mathcal{R}}$ then it is meet-irreducible in Rel. The other cases are analogous. □

Therefore, it is immediate, by Theorem 4.8-(i), that:

$$\begin{aligned} Rel &= Ang^{\mathcal{R}} \sqcap Dem^{\mathcal{R}} = Ang^{\mathcal{R}} \sqcap Inf^{\mathcal{R}} \\ Inf^{\mathcal{R}} &= Rel \ominus Ang^{\mathcal{R}} \\ Ang^{\mathcal{R}} &= Rel \ominus Dem^{\mathcal{R}} = Rel \ominus Inf^{\mathcal{R}} \end{aligned}$$

where the apex $\mathcal{R}$ denotes the relational version of the corresponding closures, i.e., the composition of each closure with the relational one. We know that the part of the hierarchy over the trace level is constituted by isomorphic levels. Then, we can think of applying Theorem 4.8-(ii), in order to propagate the properties of complementation. We can prove that the basic pattern of the hierarchy between the angelic, infinite and demonic observables can always be characterized by complementation, at any level of the Cousot's semantics hierarchy.

The following lemma proves that all the semantics, in the Cousot's hierarchy, satisfy the hypotheses of Theorem 4.8-(ii). In the following we denote the meet-irreducible elements of $\mathcal{R}^\infty$ by $Mirr_{\mathcal{R}} \stackrel{\text{def}}{=} Mirr(\langle \wp(\Sigma \times \Sigma_\perp), \subseteq \rangle)$.

Lemma 4.13.

$$\begin{aligned} Mirr(\langle \Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq \rangle) &= \alpha^{\mathcal{D}}(Mirr_{\mathcal{R}}) \\ Mirr(\left\langle \wp(\Sigma_\perp) \xrightarrow{\text{coa}} \wp(\Sigma), \supseteq \right\rangle) &= \alpha^{g\mathcal{W}p}(\alpha^{\mathcal{D}}(Mirr_{\mathcal{R}})) \\ Mirr(\langle \wp(\Sigma) \otimes \wp(\Sigma_\perp), \supseteq \rangle) &= \alpha^{g\mathcal{H}}(\alpha^{g\mathcal{W}p}(\alpha^{\mathcal{D}}(Mirr_{\mathcal{R}}))) \end{aligned}$$

Proof. Note that, *Den* is isomorphic to *Rel* [10]. Therefore, it is immediate to determine the structure of its meet-irreducible elements:

$$Mirr(\Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq) = \left\{ f \,\middle|\, \begin{array}{l} \exists\, \overline{s} \in \Sigma, \exists s' \in \Sigma_\perp \,.\, f(\overline{s}) = \Sigma_\perp \setminus \{s'\}, \\ \forall\, s \in \Sigma \setminus \{\overline{s}\} \,.\, f(s) = \Sigma_\perp \end{array} \right\}$$

We have to prove that $Mirr(\langle \Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq \rangle) = \alpha^{\mathcal{D}}(Mirr_{\mathcal{R}})$. Recall that $\alpha^{\mathcal{D}}(X) = \lambda s.\{s' \mid \langle s, s' \rangle \in X\}$. Let us consider the two implications of the equality separately.

Let $X \in Mirr_{\mathcal{R}}$, then

$$\alpha^{\mathcal{D}}(X) = \alpha^{\mathcal{D}}((\Sigma \times \Sigma_\perp)\backslash\{\langle s_0, s_1\rangle\}) = \begin{cases} \lambda s.\Sigma_\perp & \text{if } s \neq s_0 \\ \lambda s.\Sigma_\perp\backslash\{s_1\} & \text{otherwise} \end{cases}$$

This implies that $\alpha^{\mathcal{D}}(X) \in Mirr(\langle \Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq\rangle)$. Consider now the function $f \in Mirr(\langle \Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq\rangle)$, then there exist $s_0 \in \Sigma$ and $s_1 \in \Sigma_\perp$ such that $f(s_0) = \Sigma_\perp\backslash\{s_1\}$, and for all $s \neq s_0$ we have $f(s) = \Sigma_\perp$. At this point, we can take $X = \Sigma \times \Sigma_\perp\backslash\{\langle s_0, s_1\rangle\} \in Mirr_{\mathcal{R}}$ such that $f = \alpha^{\mathcal{D}}(X)$. The other cases are similar. □

The lemma above, together with Theorem 4.8-(ii), and since all the complementary observables share only the top element, implies the following result relating program semantics at different levels of abstraction (see Fig. 7). In this case, we extend the scope of $\ominus$ from closures to semantics in the obvious way: Let $\mathcal{A} = \rho^{\mathcal{A}}(\tau^\infty)$ and $\mathcal{B} = \eta^{\mathcal{B}}(\tau^\infty)$ then $\mathcal{A}\ominus\mathcal{B} = (\rho\ominus\eta)(\tau^\infty)$.

Theorem 4.14. *In Cousot's hierarchy of semantics we have* $\tau^\infty \ominus \tau^+ = \tau^\omega$ *and* $\tau^\infty \ominus \tau^\omega = \tau^\infty \ominus \tau^{\eth} = \tau^+$. *Moreover*:

$$\mathcal{R}^\infty \ominus \mathcal{R}^+ = \mathcal{R}^\omega, \qquad \mathcal{D}^\infty \ominus \mathcal{D}^+ = \mathcal{D}^\omega$$
$$\mathcal{R}^\infty \ominus \mathcal{R}^{\eth} = \mathcal{R}^\infty \ominus \mathcal{R}^\omega = \mathcal{R}^+, \qquad \mathcal{D}^\infty \ominus \mathcal{D}^{\eth} = \mathcal{D}^\infty \ominus \mathcal{D}^\omega = \mathcal{D}^+$$

$$g\mathcal{W}p \ominus \mathcal{W}lp = \mathcal{W}p^\omega, \qquad g\mathcal{H} \ominus p\mathcal{H} = g\mathcal{H}^\omega$$
$$g\mathcal{W}p \ominus \mathcal{W}p^{\eth} = g\mathcal{W}p \ominus \mathcal{W}p^\omega = \mathcal{W}lp, \quad g\mathcal{H} \ominus g\mathcal{H}^{\eth} = g\mathcal{H} \ominus g\mathcal{H}^\omega = p\mathcal{H}$$

4.5. Decomposing predicate transformers

The predicate transformer semantics provides an intensional description of programming language semantics in terms of functions transforming logic-based descriptions of computational states. In this context, a predicate is a set of states, while a *predicate transformer* is a function transforming predicates. Consider the presentation of a transition system as $\langle \Gamma, \rightarrow\rangle$, with configurations Γ consisting of pairs of program components (e.g. expressions or commands), and program states $s \in \Sigma$ which are mappings from variables into values, and a transition relation $\rightarrow \subseteq \Gamma \times \Gamma$. The weakest precondition semantics [18] is traditionally defined as follows, where $P, Q \subseteq \Sigma$, and S is a program fragment:

$$P \Rightarrow \mathcal{W}lp(S, Q) \Leftrightarrow \forall s\,.\,(s \in P \Rightarrow (\langle S, s\rangle \rightarrow s' \wedge s' \in Q) \vee \langle S, s\rangle\uparrow)$$

A similar definition can be made for its infinite counterpart:

$$P \Rightarrow \mathcal{W}p^\omega(S, Q) \Leftrightarrow \forall s\,.\,(s \in P \Rightarrow (\langle S, s\rangle\uparrow \wedge \perp \in Q) \vee (\exists s'\,.\,\langle S, s\rangle \rightarrow s'))$$

It is immediate to transform the above relations into the following sets of states: the weakest-liberal precondition $\mathcal{W}lp(S, Q) = \{s \mid \forall s' \in \Sigma\,.\,(\langle S, s\rangle \not\rightarrow s' \vee s' \in Q)\}$ and the infinite one $\mathcal{W}p^\omega(S, Q) = \{s \mid \exists s' \in \Sigma\,.\,(\langle S, s\rangle \rightarrow s' \vee \perp \in Q)\}$. By complementing these sets, we obtain the following set-theoretic complements: the complement of the

weakest-liberal precondition, $\neg\mathcal{W}lp(S,Q) = \{s \mid \exists s' \in \Sigma\,.\,(\langle S,s\rangle \to s' \wedge s' \notin Q)\}$, and of the infinite one, $\neg\mathcal{W}p^{\omega}(S,Q) = \{s \mid \forall s' \in \Sigma.(\langle S,s\rangle \not\to s' \wedge \bot \notin Q)\} \equiv \{s \mid \langle S,s\rangle \uparrow \wedge \bot \notin Q\}$.

By Theorem 4.1, we have that the natural semantics corresponds to the reduced product of the angelic and infinite semantics. Therefore, in this context, it is the conjunction of the two semantics above, namely:

$$g\mathcal{W}p(S,Q) = \{s \mid (\langle S,s\rangle\uparrow \wedge \bot \in Q) \vee (\langle S,s\rangle \to s' \wedge s' \in Q)\}$$

In this framework, it is possible to compare logical and algebraic complementation of observables. While the algebraic complementation corresponds to abstract domain complementation, the logical one boils down to the set-theoretic complementation of predicate transformers. The following complementary relations hold between infinite weakest precondition semantics and the angelic (liberal) one: $\mathcal{W}p^{\omega}(S,Q) \setminus g\mathcal{W}p(S,Q) = \neg\mathcal{W}lp(S,Q)$ and $\mathcal{W}lp(S,Q) \setminus g\mathcal{W}p(S,Q) = \neg\mathcal{W}p^{\omega}(S,Q)$. Hence we have:

$$\begin{aligned}
\mathcal{W}lp(S,Q) \setminus g\mathcal{W}p(S,Q) &= \neg\mathcal{W}p^{\omega}(S,Q) \Leftrightarrow \\
&\mathcal{W}lp(S,Q) \Rightarrow g\mathcal{W}p(S,Q) = \mathcal{W}p^{\omega}(S,Q) \\
\mathcal{W}p^{\omega}(S,Q) \setminus g\mathcal{W}p(S,Q) &= \neg\mathcal{W}lp(S,Q) \Leftrightarrow \\
&\mathcal{W}p^{\omega}(S,Q) \Rightarrow g\mathcal{W}p(S,Q) = \mathcal{W}lp(S,Q)
\end{aligned}$$

In this way, we have the following relation between the algebraic and logical complementation of predicate transformers:

$$\begin{array}{ll}
g\mathcal{W}p \ominus \mathcal{W}lp = \mathcal{W}p^{\omega} & (\mathcal{W}lp(S,Q) \Rightarrow g\mathcal{W}p(S,Q)) = \mathcal{W}p^{\omega}(S,Q) \\
g\mathcal{W}p \ominus \mathcal{W}p^{\omega} = \mathcal{W}lp & (\mathcal{W}p^{\omega}(S,Q) \Rightarrow g\mathcal{W}p(S,Q)) = \mathcal{W}lp(S,Q)
\end{array}$$

Algebraic transformation Logic transformation

This implies that $P \Rightarrow \mathcal{W}p^{\omega}(S,Q)$ iff $P \wedge \mathcal{W}lp(S,Q) \Rightarrow g\mathcal{W}p(S,Q)$. An analogous, but dual, formulation holds for the weakest-liberal precondition semantics, with respect to the infinite one. This shows that the domain complementation corresponds to the classical implication, as far as predicate transformers are concerned.

We conclude this section by proving that the weakest precondition semantics, which abstracts the demonic relational semantics [10], is too abstract to provide any significant decomposition of the demonic relational semantics. We consider the relational semantics, which is the simplest semantics in Cousot's hierarchy, which is isomorphic to the weakest precondition one.

Lemma 4.15. *Let $\alpha^{\mathcal{R}}(Dem^{\mathcal{R}})$ be the demonic relational closure, and let $\alpha^{\mathcal{R}}(Inf^{\mathcal{R}})$ be the infinite one. Then we have*:

$$\begin{aligned}
Mirr(\alpha^{\mathcal{R}}(Dem^{\mathcal{R}})) = {} & Mirr(\alpha^{\mathcal{R}}(Inf^{\mathcal{R}})) \\
& \cup \{X \in \wp(\Sigma \times \Sigma_{\bot}) \mid X = (\Sigma \times \Sigma_{\bot}) \setminus \{\langle s,s'\rangle \langle s,\bot\rangle\}, s,s' \in \Sigma\}
\end{aligned}$$

Proof. Let us denote with M the set on the right side of the equality. We prove the two inclusions separately. Consider an element $X \in M$. We can prove that it is meet-irreducible in the demonic relational closure. If X belongs to the set of meet-irreducible elements of $\alpha^{\mathcal{R}}(Inf^{\mathcal{R}})$, then X is meet-irreducible and it is in $\alpha^{\mathcal{R}}(Dem^{\mathcal{R}})$ because, by definition, the demonic semantics contains the infinite one. Suppose now that $X \in M$, but not in $Mirr(\alpha^{\mathcal{R}}(Inf^{\mathcal{R}}))$, i.e., $X \in \left\{ X \in \wp(\Sigma \times \Sigma_\perp) \;\middle|\; X = (\Sigma \times \Sigma_\perp) \setminus \{\langle s, s' \rangle, \langle s, \perp \rangle\}, s, s' \in \Sigma \right\}$, in other words let $X = (\Sigma \times \Sigma) \setminus \{\langle s, s' \rangle, \langle s, \perp \rangle\}$. We can consider $A, B \in \alpha^{\mathcal{R}}(Dem^{\mathcal{R}})$ such that $A \cap B = X$, this means that $(\Sigma \times \Sigma) \setminus \{\langle s, s' \rangle\} \subseteq A^+$ and $(\Sigma \times \perp) \setminus \{\langle s, \perp \rangle\} \subseteq A^\omega$, and the same holds for B. We can see that each possible combination implies that $X = A$ or $X = B$, or it implies a contradiction. Since we are in the demonic observable, if $A^\omega = \Sigma \times \perp$, then $A^+ = \Sigma \times \Sigma$. In these conditions, if $B^+ = \Sigma \times \Sigma$ or $B^\omega = \Sigma \times \perp$, then $A \cap B$ cannot be X, so we have $X = B$. Suppose now that $A^\omega = (\Sigma \times \perp) \setminus \{\langle s, \perp \rangle\}$. If $A^+ = (\Sigma \times \Sigma) \setminus \{\langle s, s' \rangle\}$, then $X = A$. Hence, consider $A^+ = \Sigma \times \Sigma$. In this case, if $B^+ = \Sigma \times \Sigma$ then $A \cap B \neq X$, therefore $B^+ = (\Sigma \times \Sigma) \setminus \{\langle s, s' \rangle\}$ and, since B is in the demonic closure, this implies that $B^\omega = (\Sigma \times \perp) \setminus \{\langle s, \perp \rangle\}$, namely $B = X$. We can conclude that M is a subset of the meet-irreducibles of the demonic relational closure.

Consider, now, X meet-irreducible in the relational demonic semantics. This means that, if $X = A \cap B$, then either $X = A$ or $X = B$. Suppose that $X \notin M$, then we have the following possible situations:

1. $X^+ = (\Sigma \times \Sigma) \setminus D_1$, with $D_1 \subseteq \Sigma \times \Sigma$ and $|D_1| > 1$;
2. $X = (\Sigma \times \Sigma_\perp) \setminus \langle s, s' \rangle$;
3. $X^\omega = (\Sigma \times \perp) \setminus D_2$, with $D_2 \subseteq \Sigma \times \perp$ and $|D_2| > 1$.

It is worth noting that, in the second case, we have that X does not belong to the demonic closure, which is a contradiction. Consider, then, the condition (1), we can define the sets $A = X^\omega \cup ((\Sigma \times \Sigma) \setminus D_1')$ and $B = X^\omega \cup ((\Sigma \times \Sigma) \setminus \{x\})$, with $x \in D_1$ and $D_1' = D_1 \setminus \{x\}$. Then, it is immediate to note that $A \cap B = X$ with $X \neq A$ and $X \neq B$, which contradicts the hypothesis on X. Analogously, we can prove that the third point leads to a contradiction. Therefore, we can conclude that, if X is meet-irreducible in the relational demonic semantics, then it belongs to M. □

The lemma above tells us that all the meet-irreducible elements of the demonic semantics include infinite computations. In this case, the $\mathcal{W}p$ semantics forgets about the input states that may lead to an infinite computation [10]. This means that this semantics does not include the meet-irreducibles of the demonic relational one. In this sense, it does not factorize the demonic closure. In order to understand this situation better, let us consider the following example.

Example 4.16. Let $X = \left\{ \langle a, a \rangle, \langle a, b \rangle, \langle b, a \rangle, \langle a, \perp \rangle \right\}$ be an element of the domain in Fig. 6 (meet-irreducible). Then the map $f = \alpha^{\mathcal{D}}(X)$ is such that

$$a \mapsto \{a, b, \perp\}$$
$$b \mapsto \{a\}$$

At this point, in order to calculate the $g\mathcal{W}p$ semantics, we consider the *predicate transformer* $\Phi = \alpha^{g\mathcal{W}p}(f) : \wp(\Sigma_\perp) \to \wp(\Sigma)$, and we obtain the function that executes the following

associations:

$$\begin{array}{rcl} \{a\} & \mapsto & \{b\} \\ \{a,b,\bot\} & \mapsto & \{a\} \\ \varnothing, \{b\}, \{\bot\}, \{a,\bot\}, \{b,\bot\}, \{a,b\} & \mapsto & \varnothing \end{array}$$

At this point, the abstraction $\alpha^{\mathcal{W}_P}$ reduces the domain of this function to $\wp(\Sigma)$, obtaining

$$\begin{array}{rcl} \{a\} & \mapsto & \{b\} \\ \varnothing, \{b\}, \{a,b\} & \mapsto & \varnothing \end{array}$$

But, if we concretize with $\gamma^{\mathcal{W}_P}$, we obtain the function Φ' that is such that $\Phi' : \{a\} \mapsto \{b\}$ while for each set $X \in \wp(\Sigma_\bot) \setminus \{a\}$ we have $\Phi' : X \mapsto \varnothing$. So, concretizing this map till the relational domain, we obtain the set $Y = \Sigma \times \Sigma_\bot$ which strictly contains X.

4.6. Decomposing demonic semantics

In Section 4.2, we proved that the angelic and demonic abstractions do not factorize natural semantics in most abstract factors. This means that it is possible to isolate an observable which is more abstract than the demonic semantics, and complementary with respect to the infinite one. In this section, we give a computational meaning to this new observable.

Lemma 4.17. *Let $\sigma \in \Sigma^\infty$ and $\Delta_\sigma \stackrel{\text{def}}{=} \{\delta \in \Sigma^\omega \mid \delta_\vdash = \sigma_\vdash\}$. Then we have*:

$$Mirr(Dem) = Mirr(Inf) \cup \left\{ X \subseteq \Sigma^\infty \;\middle|\; \exists\, \sigma \in \Sigma^+ \,.\, X = \Sigma^\infty \setminus (\Delta_\sigma \cup \{\sigma\}) \right\}$$

Proof. Let us use the denotation $M \stackrel{\text{def}}{=} \left\{ X \subseteq \Sigma^\infty \;\middle|\; \exists\, \sigma \in \Sigma^+ \,.\, X = \Sigma^\infty \setminus (\Delta_\sigma \cup \{\sigma\}) \right\}$, and consider the two inclusions separately. Let us prove that $M \subseteq Mirr(Dem)$, namely consider $X \in M$. If $X \in Mirr(Inf)$, then X is meet-irreducible also in *Dem*. This because the only sets of traces in *Dem* containing all the finite traces are elements in *Inf*. Now, if $X = \Sigma^\infty \setminus (\Delta_\sigma \cup \{\sigma\})$, we have the following possible cases. Consider $A_1, A_2 \in Dem$ such that $A_1 \cap A_2 = X$, then we have that $\Sigma^\omega \setminus \Delta_\sigma \subseteq A_i^\omega$, and $\Sigma^+ \setminus \{\sigma\} \subseteq A_i^+$, for $i = 1, 2$. Therefore, for the infinite part of the sets, we have the following possible cases: $A_i^\omega = \Sigma^\omega \setminus \Delta_\sigma$, $A_i^\omega = \Sigma^\omega \setminus D_0$, where $D_0 \subset \Delta_\sigma$, or $A_i^\omega = \Sigma^\omega$. While, for the finite part of the sets, we have the following possible cases: $A_i^+ = \Sigma^+ \setminus \{\sigma\}$ or $A_i^+ = \Sigma^+$. We can prove that in all the combinations of these cases either $A_i = X$ or we find a contradiction.

- Consider $A_1^\omega = \Sigma^\omega \setminus \Delta_\sigma$. If $A_1^+ = \Sigma^+ \setminus \{\sigma\}$, then $A_1 = X$. Therefore, let us consider $A_1^+ = \Sigma^+$, i.e., $A_1 = \Sigma^\infty \setminus \Delta_\sigma$. In these conditions, if $A_2^+ = \Sigma^+$, then $A_1 \cap A_2 \neq X$, therefore $A_2^+ = \Sigma^+ \setminus \{\sigma\}$. As far as A_2^ω is concerned, if $A_2^\omega = \Sigma^\omega \setminus \Delta_\sigma$, then $A_2 = X$, and if $A_2^\omega = \Sigma^\omega$ or $A_2^\omega = \Sigma^\omega \setminus D_0$, then $A_2 \notin Dem$.
- Consider $A_1^\omega = \Sigma^\omega$. Since we are considering elements in *Dem*, this implies that $A_1^+ = \Sigma^+$, i.e., $A_1 = \Sigma^\infty$. In these conditions, it is clear that, if $A_2^+ = \Sigma^+$, then $A_1 \cap A_2 \neq X$, hence $A_2^+ = \Sigma^+ \setminus \{\sigma\}$. As far as A_2^ω is concerned, if $A_2^\omega = \Sigma^\omega$, then $A_1 \cap A_2 \neq X$, if $A_2^\omega = \Sigma^\omega \setminus \Delta_\sigma$, then $A_2 = X$, and if $A_2^\omega = \Sigma^\omega \setminus D_0$, then we would have $A_2 \notin Dem$.
- Consider $A_1^\omega = \Sigma^\omega \setminus D_0$. In this case, we have $A_1^+ = \Sigma^+$, otherwise $A_1 \notin Dem$, i.e., $A_1 = \Sigma^\infty \setminus D_0$. On the other hand, $A_2^+ = \Sigma^+ \setminus \{\sigma\}$, otherwise $A_1 \cap A_2 \neq X$. Finally,

if $A_2^\omega = \Sigma^\omega$, then $A_1 \cap A_2 \notin \mathit{Dem}$, if $A_2^\omega = \Sigma^\omega \setminus D_0$, then $A_1 \cap A_2 \neq X$, and if $A_2^\omega = \Sigma^\omega \setminus \Delta_\sigma$, then $A_2 = X$.

Now we have to prove that $\mathit{Mirr}(\mathit{Dem}) \subseteq M$, namely that if $X \notin M$, then $X \notin \mathit{Mirr}(\mathit{Dem})$. Hence, consider $X \notin M$. Then, there exists $D_0 \subseteq \Sigma^\omega$, with $D_0 \supset \Delta_\sigma$, such that $X^\omega = \Sigma^\omega \setminus D_0$, or there exists $D_1 \subseteq \Sigma^+$, with $|D_1| > 1$, such that $X^+ = \Sigma^+ \setminus D_1$, and $X \notin \mathit{Mirr}(\mathit{Inf})$. In these conditions, if $X^+ = \Sigma^+$, then $X \in \mathit{Inf}$, by definition of Inf, and X is meet-irreducible. All these facts together imply that $X \in \mathit{Mirr}(\mathit{Inf})$, which is a contradiction, since $X \notin \mathit{Mirr}(\mathit{Inf})$. Therefore, consider $X^+ \subset \Sigma^+$ and, in particular, consider $X^+ = \Sigma^+ \setminus D_1$. We can define the sets $D_1' \stackrel{\text{def}}{=} D_1 \setminus \{\sigma'\}$, with $\sigma' \in D_1$, $A \stackrel{\text{def}}{=} X^\omega \cup (\Sigma^+ \setminus D_1')$ and $B \stackrel{\text{def}}{=} X^\omega \cup (\Sigma^+ \setminus \{\sigma'\})$. It is worth noting that $A \cap B = X$ with $A \neq X$ and $B \neq X$, namely X is not meet irreducible in Dem. Finally, if $X^+ = \Sigma^+ \setminus \{\sigma\}$ and $X^\omega = \Sigma^\omega \setminus D_0$ (by hypothesis), then we can define the sets $D_0' \stackrel{\text{def}}{=} D_0 \setminus \{\delta'\}$, with $\delta' \notin \Delta_\sigma$ and $\delta' \in D_0$, $A \stackrel{\text{def}}{=} \Sigma^\infty \setminus (D_0' \cup \{\sigma\})$ and $B \stackrel{\text{def}}{=} \Sigma^\infty \setminus \{\delta'\}$. It is worth noting that, also in this case, $A \cap B = X$ with $A \neq X$ and $B \neq X$, namely also in this case $X \notin \mathit{Mirr}(\mathit{Dem})$. We proved, in this way, that $M = \mathit{Mirr}(\mathit{Dem})$. □

It is clear that, for each $x \in \mathit{Mirr}(\mathit{Dem}) \setminus \mathit{Mirr}(\mathit{Inf})$, we have $x \notin \mathit{Inf}$. Therefore, we can obtain a new observable which is generated from the objects of the form $X = \Sigma^\infty \setminus (\Delta_\sigma \cup \{\sigma\})$, with $\sigma \in \Sigma^+$. In particular, we can define a new closure operator, called *slothful*, that characterizes the new complementary semantics defined by $\mathit{Dem} \ominus \mathit{Inf}$. This is a closure on the demonic domain $\mathit{Slo} \in \mathit{uco}(\mathit{Dem}(\wp(\Sigma^\infty)))$, which is defined as follows:

$$\mathit{Slo} \stackrel{\text{def}}{=} \lambda X \,.\, X \cup \left\{ \delta \in \Sigma^\omega \,\middle|\, \mathit{chaos}(\delta_\vdash) \subseteq X^+ \right\}$$

On the other hand, if $X \in \mathcal{M}\left(\left\{X \subseteq \Sigma^\infty \,\middle|\, \exists\, \sigma \in \Sigma^+ \,.\, X = \Sigma^\infty \setminus (\Delta_\sigma \cup \{\sigma\})\right\}\right)$, then it means that ,when there exists a trace $\sigma \notin X^+$, then all the traces $\delta \in \Sigma^\omega$, such that $\delta_\vdash = \sigma_\vdash$, cannot be in X.

Proposition 4.18.

$$X \in \mathit{Slo} \;\Leftrightarrow\; X \in \mathcal{M}\left(\left\{X \subseteq \Sigma^\infty \,\middle|\, \exists\, \sigma \in \Sigma^+ \,.\, X = \Sigma^\infty \setminus (\Delta_\sigma \cup \{\sigma\})\right\}\right)$$

Proof. Let us prove the two implications separately. ($\Rightarrow$) Consider $X \in \mathit{Slo}$, we know by definition of slothful closure that $\sigma \in X^\omega$ implies $\mathit{chaos}(\sigma_\vdash) \subseteq X^+$. Hence

$$\begin{aligned} X \in \mathit{Slo} &\Leftrightarrow X = X \cup \left\{ \delta \in \Sigma^\omega \,\middle|\, \mathit{chaos}(\delta_\vdash) \subseteq X^+ \right\} \\ &\Leftrightarrow \forall\, \delta \in X^\omega \,.\, \mathit{chaos}(\delta_\vdash) \subseteq X^+ \end{aligned}$$

Suppose that X is not the intersection of elements in $\mathit{Mirr}(\mathit{Dem}) \setminus \mathit{Inf}$, namely there exists an infinite trace $\delta \in X$, starting with the initial state of a finite trace $\sigma \notin X$. By definition of X, we know that $\mathit{chaos}(\delta_\vdash) \subseteq X^+$, then, since $\sigma \in \mathit{chaos}(\delta_\vdash)$ (being $\delta_\vdash = \sigma_\vdash$), we would have $\sigma \in X^+$, which is a contradiction. ($\Leftarrow$) Let us prove that each element in $\mathit{Mirr}(\mathit{Dem}) \setminus \mathit{Inf}$ belongs to the closure operator. Consider $\mathit{Slo}(\Sigma^\infty \setminus (\Delta_\sigma \cup \{\sigma\}))$. By definition, this closure adds, with each set A, the chaos of each infinite trace in A. This means that the operation cannot add any trace contained in Δ_σ, because they are all infinite. Moreover, it cannot add σ, since there are no infinite traces starting with $\sigma_\vdash$, being these traces in Δ_σ. Hence, we can

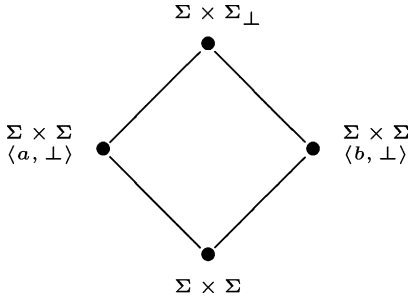

Fig. 5. Relational infinite semantics on $\Sigma = \{a, b\}$.

conclude that all the meet-irreducible elements in $Mirr(Dem) \setminus Inf$ are fix-points of *Slo*, which is meet-generated by them. □

It is simple to verify that this new semantics is unable to distinguish whether, in a set of traces,the set of all the finite traces with the same given initial state *s*, is generated by the existence of an infinite trace in τ^∞ starting from *s*, or it is produced by the program itself. This abstraction is achieved by enhancing any set of traces X with all the infinite traces $\delta \in \Sigma^\omega$, whenever the chaos generated by δ, namely $chaos(\delta_\vdash)$, is contained in X.

The following result is straightforward by Theorem 4.2 and by Lemma 4.17. As expected, this new semantics $Slo(Dem(\tau^\infty))$ is unable to observe infinite behaviors. Moreover, $(Dem \ominus Inf) \circ Dem$ is unable to factorize the basic trace semantics. Indeed, for any $X \in Mirr(\wp(\Sigma^\infty))$ we have $(Dem \ominus Inf)(Dem(X)) = \Sigma^\infty$ (Figs. 5 and 6).

Theorem 4.19. *$Dem \ominus Inf = Slo$, $Dem \ominus Slo = Inf$, and $Nat \ominus (Slo \circ Dem) = Nat$.*

We can conclude that the infinite and the slothful semantics are fully complementary, namely they share only the demonic top element Σ^∞.

Proposition 4.20. *$Slo \sqcup Inf = \Sigma^\infty$.*

The following result proves that the complementary structure of the slothful semantics can be extended all over the hierarchy.

Lemma 4.21. 1. *$Rel \circ Slo = Slo \circ Rel$.*
2. *$Mirr(Rel(Dem(\wp(\Sigma^\infty))) \cap Slo(\wp(\Sigma^\infty))) \subseteq Mirr(Rel(Dem(\wp(\Sigma^\infty))))$.*

Proof. By Theorem 4.19, Proposition 4.20 and Lemma 4.10, the proof is analogous to the one of Propositions 4.11 and 4.12. □

Therefore, by using also Proposition 4.11, we can apply Theorem 4.8-(i), obtaining that:

$$\begin{aligned} Dem^{\mathcal{R}} &= Slo^{\mathcal{R}} \sqcap Inf^{\mathcal{R}} \\ Inf^{\mathcal{R}} &= Dem^{\mathcal{R}} \ominus Slo^{\mathcal{R}} \\ Slo^{\mathcal{R}} &= Dem^{\mathcal{R}} \ominus Inf^{\mathcal{R}} \end{aligned}$$

The following theorem is analogous to Theorem 4.14.

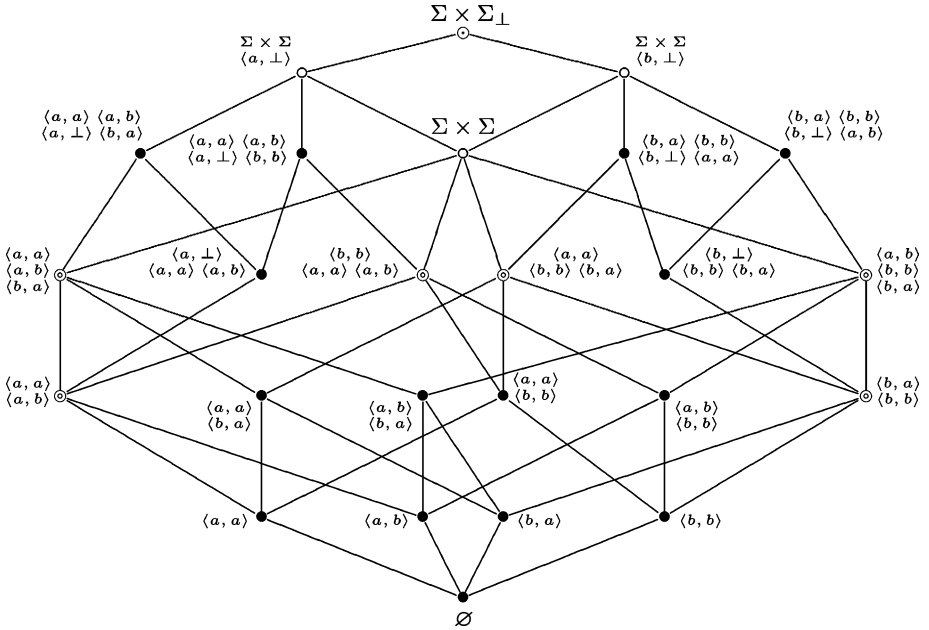

Fig. 6. Demonic relational semantics on $\Sigma = \{a, b\}$.

Theorem 4.22. *In Cousot's hierarchy of semantics we have*:

$$\mathcal{R}^{\partial} \ominus \mathcal{R}^{\varrho} = \mathcal{R}^{\omega}, \qquad \mathcal{D}^{\partial} \ominus \mathcal{D}^{\varrho} = \mathcal{D}^{\omega}$$
$$\mathcal{R}^{\partial} \ominus \mathcal{R}^{\omega} = \mathcal{R}^{\varrho}, \qquad \mathcal{D}^{\partial} \ominus \mathcal{D}^{\omega} = \mathcal{D}^{\varrho}$$

$$\mathcal{W}p^{\partial} \ominus \mathcal{W}p^{\varrho} = \mathcal{W}p^{\omega}, \; g\mathcal{H}^{\partial} \ominus g\mathcal{H}^{\varrho} = g\mathcal{H}^{\omega}$$
$$\mathcal{W}p^{\partial} \ominus \mathcal{W}p^{\omega} = \mathcal{W}p^{\varrho}, \; g\mathcal{H}^{\partial} \ominus g\mathcal{H}^{\omega} = g\mathcal{H}^{\varrho}$$

In Fig. 6 we have the relational version, on the alphabet $\Sigma = \{a, b\}$, of the demonic semantics. In this representation, we underline with empty points the elements that belong both,to the demonic and to the infinite semantics. In this figure, the slothful domain is represented by full points.

The factorization of the demonic semantics led us to the definition of a new observable, which is complementary with respect to the infinite semantics relatively to the demonic observable. We prove that this semantics can be constructively derived as the least fix-point of a monotone operator, under particular conditions. The slothful semantics can be viewed as an abstraction of the natural semantics by composing the demonic and the slothful closures. This leads to the following closure operator on the maximal trace semantics:

$$\begin{aligned} \mathit{Slo}^{\varrho}(X) &\stackrel{\text{def}}{=} \mathit{Slo} \circ \mathit{Dem}(X) \\ &= X \cup \bigcup \left\{ \mathit{chaos}(\delta_{\vdash}) \,\middle|\, \delta \in X^{\omega} \right\} \cup \left\{ \delta \in \Sigma^{\omega} \,\middle|\, \mathit{chaos}(\delta_{\vdash}) \subseteq X^{+} \right\} \end{aligned}$$

In order to define the abstraction of the demonic observable corresponding to the slothful domain, we can think of distinguishing, in the set of all the finite traces of a

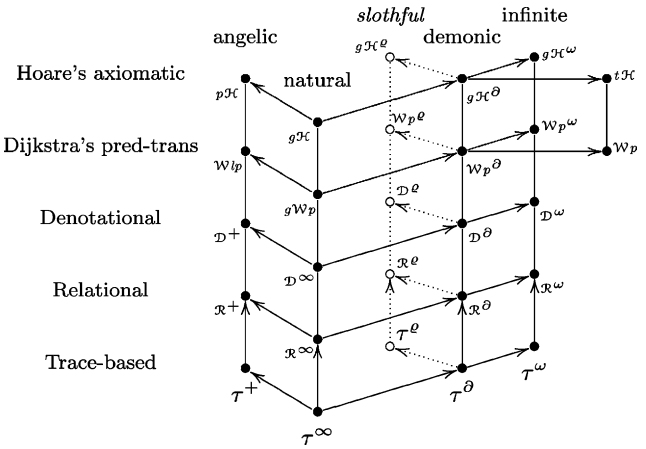

Fig. 7. Semantics in Cousot's hierarchy.

program, those traces which are the effect of a computation, from those which belong to the chaos given by an infinite computation. In the following we will use the notation: $CH \stackrel{\text{def}}{=} \left\{ chaos(s) \,\middle|\, s \in \Sigma \right\} \subseteq \wp(\Sigma^+)$. Consider the set D^{ϱ}, with elements $X = \langle X^+, X^{ch} \rangle$, where we consider $X^+ \in \wp(\Sigma^+)$ and $X^{ch} \in \wp(CH)$, defined as

$$D^{\varrho} = \left\{ \left\langle X^+, X^{ch} \right\rangle \in \wp(\Sigma^+) \times \wp(CH) \,\middle|\, \begin{array}{l} \sigma \in X^+ \;\Rightarrow\; chaos(\sigma_{\vdash}) \not\subseteq X^+, \\ chaos(s) \in X^{ch} \\ \quad \Rightarrow\; \forall \delta \in \Sigma^+ \,.\, s\delta \notin X^+ \end{array} \right\}$$

The abstraction and concretization functions are, respectively, $\alpha^d : D^{\hat{\partial}} \to D^{\varrho}$ and $\gamma^d : D^{\varrho} \to D^{\hat{\partial}}$, and they are defined as follows: If $X \in D^{\hat{\partial}}$ and $Y \in D^{\varrho}$ then

$$\alpha^d(X) = \left\langle \left\{ \sigma \in X^+ \,\middle|\, chaos(\sigma_{\vdash}) \not\subseteq X^+ \right\}, \left\{ chaos(\sigma_{\vdash}) \,\middle|\, chaos(\sigma_{\vdash}) \subseteq X^+ \right\} \right\rangle$$
$$\gamma^d(Y) = Y^+ \cup \left\{ \delta \in \Sigma^{\infty} \,\middle|\, chaos(\delta_{\vdash}) \in Y^{ch} \right\}$$

The idea is the following. The abstraction ignores the infinite traces, while it keeps their chaos, which is represented by the set $chaos(s)$, for their initial state s. The concretization, instead, leaves unchanged the finite traces and substitutes each set $chaos(s)$ with all the finite and infinite traces starting with the state s. It is worth noting that $Slo = \gamma^d \circ \alpha^d$.

The semantics, not originally included in Cousot's hierarchy, are represented in Fig. 7 with dashed lines and empty points.

The characterization of the slothful semantics as a fix-point of a monotone operator on traces requires the definition of a computational order which has to be coherent with the structure of the slothful abstraction. In order to obtain this, we apply the abstraction above to the demonic domain reordered by the computational order, $\langle D^{\hat{\partial}}, \sqsubseteq^{\hat{\partial}} \rangle$, defined in [10]: $X \sqsubseteq^{\hat{\partial}} Y$ if for each $\sigma \in \Sigma^{\omega}$:

$$\sigma \in X \;\vee\; (\sigma \notin Y \;\wedge\; \forall \delta \in \Sigma^+ .\, \sigma_{\vdash}\delta \in X \;\Rightarrow\; \sigma_{\vdash}\delta \in Y)$$

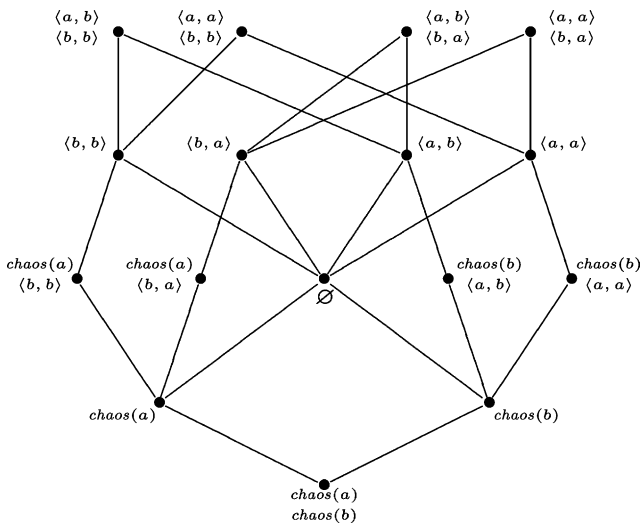

Fig. 8. The reordered slothful domain.

This order says that $X \cap \Sigma^\omega \supseteq Y \cap \Sigma^\omega$, and for any state s, which in both X and Y does not lead to non-termination, we have $X \cap \mathit{chaos}(s) \subseteq Y \cap \mathit{chaos}(s)$. It is trivial to prove that the computational order, induced on the domain D^ϱ, is defined as follows: $X \sqsubseteq^\varrho Y$ iff $X^+ \subseteq Y^+$ and $X^{ch} \supseteq Y^{ch}$. We can observe that the induced least upper bound is

$$\bigsqcup{}^\varrho X_i = \left\langle \bigcup_i X_i^+, \bigcap_i X_i^{ch} \right\rangle$$

where $\{X_i\}_{i\in I} \subseteq D^\varrho$, i.e., it is the union on the finite traces part and the intersection on the chaos part, of the sets involved in the operation. The greatest lower bound is defined dually. The problem here is that $\langle D^\varrho, \sqsubseteq^\varrho \rangle$ is not a complete partial order (CPO). This implies that we cannot specify the slothful semantics of a program as the least fix-point of a monotone operator, i.e., the slothful semantics does not have a computational meaning for infinite state systems, such as programs. In order to observe this fact, we remind that a set X is in D^ϱ iff $\forall s \in \Sigma\,.\,\mathit{chaos}(s) \in X \;\Rightarrow\; \forall s\delta \in \Sigma^+\,.\,s\delta \notin X^+$ and $\forall s\delta \in X^+\,.\,\mathit{chaos}(s) \not\subseteq X^+$. Consider a state $s \in \Sigma$, and consider the increasing chain $\{X_i\}_{i<\omega} \subseteq D^\varrho$ defined as follows:

$$X_0 = \langle \varnothing, \varnothing \rangle$$
$$X_n = \left\langle X_{n-1}^+ \cup (\mathit{chaos}(s) \cap \Sigma^n), \varnothing \right\rangle$$

where Σ^n is the set of all the finite traces whose length is $n < \omega$. It is worth noting that $\forall i\,.\,X_i \in D^\varrho$ and that $\forall i\,.\,X_i \sqsubseteq^\varrho X_{i+1}$. Then, we have $X_\omega = \bigsqcup_n X_n$ and it is clear that $\mathit{chaos}(s) \subseteq X_\omega^+ = \bigcup_n X_n^+$, while $X_\omega^{ch} = \varnothing$, namely $X_\omega \notin D^\varrho$. The problem here is that the chaos of a state is an infinite set. This means that we are not able to systematically build the slothful observable on transition systems involving infinite states. Indeed, the only situation, where the argument above fails, is when the domain is finite, namely when we consider the relational domain with a finite set of states such as in Fig. 8. The idea is that of finding a monotone operator able to systematically derive the slothful relational semantics,

under the hypothesis of finite states. We can rewrite, in the relational domain, all the objects defined above, as follows:

- $chaos^{\mathcal{R}}(s) = \left\{ \langle s, s' \rangle \,\middle|\, s' \in \Sigma \right\}$;
- $CH^{\mathcal{R}} = \left\{ chaos^{\mathcal{R}}(s) \,\middle|\, s \in \Sigma \right\}$;
- $D^{\mathcal{R}_\varrho} \stackrel{\text{def}}{=} \alpha^{\mathcal{R}}(D^\varrho)$
$$= \left\{ \langle X^+, X^{ch} \rangle \in \wp(\Sigma \times \Sigma) \times \wp(CH^{\mathcal{R}}) \,\middle|\, \begin{array}{l} \langle s, s' \rangle \in X^+ \;\Rightarrow\; chaos^{\mathcal{R}}(s) \not\subseteq X^+, \\ chaos^{\mathcal{R}}(s) \in X^{ch} \\ \quad \Rightarrow\; \forall s' \in \Sigma \,.\, \langle s, s' \rangle \notin X^+ \end{array} \right\};$$
- $X, Y \in D^{\mathcal{R}_\varrho} :\; X \sqsubseteq^\varrho Y \;\Leftrightarrow\; X^+ \subseteq Y^+ \;\wedge\; X^{ch} \supseteq Y^{ch}$.

In the following theorem, we will denote with $chaos(s)$ the set $chaos^{\mathcal{R}}(s)$ and with CH the set $CH^{\mathcal{R}}$. Remind that T is the set of all the final/blocking states.

Theorem 4.23. *Let $s_1, s_2, s \in \Sigma$ and $\overline{\tau} = \left\{ \langle s, s \rangle \,\middle|\, s \in T \right\}$. Let $F^\varrho(X) \in D^{\mathcal{R}_\varrho} \longrightarrow D^{\mathcal{R}_\varrho}$ be a monotone operator defined as*

$$F^\varrho(X) = \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2 \rangle \,\middle|\, s_1 \tau s,\; \langle s, s_2 \rangle \in X^+,\; chaos(s_1) \notin X^{ch} \right\}, \right. \\ \left. \left\{ chaos(s) \,\middle|\, s \tau s_1,\; chaos(s_1) \in X^{ch} \right\} \right\rangle$$

Then $\mathcal{R}^\varrho = lfp^{\sqsubseteq^\varrho}_{CH} F^\varrho$.

Proof. Let us define the following notation. Consider $a, b \in \Sigma$ and $0 < i < \omega$:

$$a\tau^i b \;\Leftrightarrow\; \exists s_1, s_2, \ldots, s_{i-1} \,.\, a\tau s_1 \tau s_2 \tau \ldots \tau s_{i-1} \tau b \;\text{ and }\; a \in \tau^i \;\Leftrightarrow\; \exists b \in \Sigma \,.\, a\tau^i b$$

We prove that the nth iteration of F^ϱ is

$$F^\varrho_n = \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2 \rangle \,\middle|\, \exists\, i \,.\, 0 < i < n \,.\, s_1 \tau^i s_2,\; s_2 \in T,\; s_1 \notin \tau^n \right\}, \right. \\ \left. \left\{ chaos(s) \,\middle|\, s \in \tau^{n+1} \right\} \right\rangle$$

We prove this by induction on n. Consider the base of the induction, namely $n = 0$:

$$F^\varrho_0(CH) = \left\langle \overline{\tau}, \left\{ chaos(s) \,\middle|\, s \tau s_1,\; chaos(s_1) \in CH \right\} \right\rangle = \left\langle \overline{\tau}, \left\{ chaos(s) \,\middle|\, s \in \tau^1 \right\} \right\rangle$$

Suppose now that the hypothesis holds for n, we can calculate the $(n+1)$th iteration.

$$F^\varrho_{n+1} = F^\varrho(F^\varrho_n) = \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2 \rangle \,\middle|\, s_1 \tau s,\; \langle s, s_2 \rangle \in F^\varrho_n,\; chaos(s_1) \notin F^\varrho_n \right\}, \right. \\ \left. \left\{ chaos(s) \,\middle|\, s \tau s_1,\; chaos(s_1) \in F^\varrho_n \right\} \right\rangle$$

$$= \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2 \rangle \,\middle|\, \begin{array}{l} s_1 \tau s,\; \langle s, s_2 \rangle \in \overline{\tau} \cup \left\{ \langle s_1, s_2 \rangle \,\middle|\, \begin{array}{l} \exists\, 0 < i < n \,.\, s_1 \tau^i s_2, \\ s_2 \in T,\; s_1 \notin \tau^n \end{array} \right\}, \\ s_2 \in T,\; chaos(s_1) \notin \left\{ chaos(s) \,\middle|\, s \in \tau^{n+1} \right\} \end{array} \right\}, \right\}, \right.$$

$$\left. \left\{ chaos(s) \,\middle|\, s \tau s_1,\; s_1 \in \tau^{n+1} \right\} \right\rangle$$

$$= \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2\rangle \;\middle|\; \begin{array}{l} s_1\tau s,\ \langle s, s_2\rangle \in \overline{\tau} \cup \left\{ \langle s_1, s_2\rangle \;\middle|\; \begin{array}{l} \exists\, i\ .\ 0 < i < n\ .\ s_1\tau^i s_2, \\ s_2 \in T,\ s_1 \notin \tau^n \end{array} \right\}, \\ s_2 \in T,\ s_1 \notin \tau^{n+1} \end{array} \right\}, \right.$$

$$\left. \left\{ \mathit{chaos}(s) \;\middle|\; s \in \tau^{n+2} \right\} \right\rangle$$

$$= \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2\rangle \;\middle|\; s_1\tau s_2,\ s_2 \in T,\ s_1 \notin \tau^{n+1} \right\} \cup \right.$$

$$\left\{ \langle s_1, s_2\rangle \;\middle|\; s_1\tau s,\ s_2 \in T,\ \exists i\ .\ 0 < i < n\ .\ s\tau^i s_2,\ s \notin \tau^n,\ s_1 \notin \tau^{n+1} \right\}$$

$$\left. \left\{ \mathit{chaos}(s) \;\middle|\; s \in \tau^{n+2} \right\} \right\rangle$$

$$= \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2\rangle \;\middle|\; s_1\tau s_2,\ s_2 \in T,\ s_1 \notin \tau^{n+1} \right\} \cup \right.$$

$$\left\{ \langle s_1, s_2\rangle \;\middle|\; \begin{array}{l} \exists\, 1 < i < n+1\ .\ s_1\tau^i s_2, \\ s_2 \in T,\ s_1 \notin \tau^{n+1} \end{array} \right\}, \left. \left\{ \mathit{chaos}(s) \;\middle|\; s \in \tau^{n+2} \right\} \right\rangle$$

$$= \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2\rangle \;\middle|\; \exists\, 0 < i < n+1\ .\ s_1\tau^i s_2,\ s_2 \in T,\ s_1 \notin \tau^{n+1} \right\}, \right.$$

$$\left. \left\{ \mathit{chaos}(s) \;\middle|\; s \in \tau^{n+2} \right\} \right\rangle$$

It is worth noting that the resulting chain is increasing because at each iteration the condition on the finite traces part of each set become weaker, while the condition on the chaos part become stronger. Now, we have to prove that the fix-point of the function defined above is, precisely, the slothful semantics of the transition system. Namely, we have to compute the limit $\sqcup^{\mathcal{R}_\varrho}$, which is the relational version of the operator defined previously.

$$\bigsqcup_n^{\mathcal{R}_\varrho} F_n^\varrho = \bigsqcup_n^{\mathcal{R}_\varrho} \left(\left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2\rangle \;\middle|\; \begin{array}{l} \exists\, i\ .\ 0 < i < n\ .\ s_1\tau^i s_2, \\ s_2 \in T,\ s_1 \notin \tau^n \end{array} \right\}, \right. \right.$$

$$\left. \left. \left\{ \mathit{chaos}(s) \;\middle|\; s \in \tau^{n+1} \right\} \right\rangle \right)$$

$$= \left\langle \overline{\tau} \cup \left(\bigcup_n \left\{ \langle s_1, s_2\rangle \;\middle|\; \exists\, i\ .\ 0 < i < n\ .\ s_1\tau^i s_2,\ s_2 \in T,\ s_1 \notin \tau^n \right\} \right), \right.$$

$$\left. \left(\bigcap_n \left\{ \mathit{chaos}(s) \;\middle|\; s \in \tau^{n+1} \right\} \right) \right\rangle$$

$$= \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2\rangle \;\middle|\; \exists\, n > 0,\ i\ .\ 0 < i < n\ .\ s_1\tau^i s_2,\ s_2 \in T,\ s_1 \notin \tau^n \right\}, \right.$$

$$\left. \left\{ \mathit{chaos}(s) \;\middle|\; \forall\, n > 1\ .\ s \in \tau^n \right\} \right\rangle$$

It is worth noting that the set $\left\{ \langle s_1, s_2\rangle \;\middle|\; \exists\, n > 0,\ 0 < i < n\ .\ s_1\tau^i s_2,\ s_2 \in T,\ s_1 \notin \tau^n \right\}$ takes all the pairs of states, initial and final of finite traces, where the first state cannot lead to non-termination, while the set $\left\{ \mathit{chaos}(s) \;\middle|\; \forall\, n > 1\ .\ s \in \tau^n \right\}$ takes all the $\mathit{chaos}(s)$ where s may lead to non-termination. This is exactly the slothful semantics. □

Example 4.24. Consider the following transition system:

$$\langle \Sigma, \tau \rangle \text{ with } \Sigma = \{a, b, c, d\},\ \tau = \{\langle a, a\rangle, \langle a, b\rangle, \langle c, b\rangle, \langle c, d\rangle, \langle d, b\rangle\} \text{ and } T = \{b\}$$

It is worth noting that the maximal trace semantics corresponding to this transition system is the set of traces $\{cb, cdb, db, b, ab, aab, aa\ldots b, \ldots, aa\ldots a\ldots\}$. Moreover, it is clear that the slothful semantics is $\langle\{b, cb, cdb, db\}, \{chaos(a)\}\rangle$ and the relational version is $\left\langle\{\langle b, b\rangle, \langle c, b\rangle, \langle d, b\rangle\}\{chaos^{\mathcal{R}}(a)\}\right\rangle$. Let us see how this semantics can be derived systematically by using the function above. In the following, we will denote again the set $chaos^{\mathcal{R}}(s)$ simply by $chaos(s)$.

$$\begin{aligned}
F_0^{\varrho}(CH) &= \left\langle \overline{\tau}, \left\{ chaos(s) \,\middle|\, s \in \tau^1 \right\} \right\rangle \\
&= \langle \{\langle b, b\rangle\}, \{chaos(a), chaos(c), chaos(d)\} \rangle \\
F_1^{\varrho}(CH) &= \left\langle \overline{\tau}, \left\{ chaos(s) \,\middle|\, s \in \tau^2 \right\} \right\rangle \\
&= \langle \{\langle b, b\rangle\}, \{chaos(a), chaos(c)\} \rangle \\
F_2^{\varrho}(CH) &= \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2\rangle \,\middle|\, \begin{array}{l} \exists\, 0 < i < 2 \,.\, s_1\tau^i s_2, \\ s_2 \in T,\ s_1 \notin \tau^2 \end{array} \right\}, \left\{ chaos(s) \,\middle|\, s \in \tau^3 \right\} \right\rangle \\
&= \langle \{\langle b, b\rangle\} \cup \{\langle d, b\rangle\}, \{chaos(a)\} \rangle \\
F_3^{\varrho}(CH) &= \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2\rangle \,\middle|\, \begin{array}{l} \exists\, i \,.\, 0 < i < 3 \,.\, s_1\tau^i s_2, \\ s_2 \in T,\ s_1 \notin \tau^3 \end{array} \right\}, \left\{ chaos(s) \,\middle|\, s \in \tau^4 \right\} \right\rangle \\
&= \langle \{\langle b, b\rangle\} \cup \{\langle d, b\rangle, \langle c, b\rangle\}, \{chaos(a)\} \rangle \\
F_4^{\varrho}(CH) &= \left\langle \overline{\tau} \cup \left\{ \langle s_1, s_2\rangle \,\middle|\, \begin{array}{l} \exists\, i \,.\, 0 < i < 4 \,.\, s_1\tau^i s_2, \\ s_2 \in T,\ s_1 \notin \tau^4 \end{array} \right\}, \left\{ chaos(s) \,\middle|\, s \in \tau^5 \right\} \right\rangle \\
&= \langle \{\langle b, b\rangle\} \cup \{\langle d, b\rangle, \langle c, b\rangle\}, \{chaos(a)\} \rangle
\end{aligned}$$

We have reached, in this way, the fix-point $\{\langle b, b\rangle, \langle d, b\rangle, \langle c, b\rangle\} \cup \{chaos(a)\}$, which is exactly the slothful semantics of the given transition system.

5. Relational composition of semantics: compositionality

The independent composition of observables does not model the way relational information can be extracted from traces by abstract interpretation. In particular the independent composition is inadequate for modeling compositional semantics as abstract interpretations of trace semantics. In general, a semantics is said to be compositional when the semantics of a program can be reconstructed from the semantics of its components. In this section, we specify the property of semantics compositionality as a property of the corresponding closure operator on maximal traces. Indeed, the maximal trace semantics is also a well known *compositional* semantics, namely it is equal to the composition of the semantics of program's sub-components. The idea is that we can compose the observations made on partial computations, obtaining back, as result and without any loss of precision, the

observation of the whole computation. We remind the reader that, if we denote as $[\![\cdot]\!]$ this semantics, then we can describe its compositionality as $[\![P_1; P_2]\!] = [\![P_1]\!]^\frown[\![P_2]\!]$, where P_1 and P_2 are generic programs. Consider a GI: $\langle \wp(\Sigma^\infty), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leqslant_A\rangle$ defined on the concrete domain of the maximal traces $\wp(\Sigma^\infty)$. This induces an abstract semantics which is defined on the abstract domain of denotations A: $[\![\cdot]\!]^A \stackrel{\text{def}}{=} \alpha([\![\cdot]\!])$. Our aim is that of characterizing those abstract semantics that are compositional as regards as the concatenation of programs, i.e., such that $[\![P_1; P_2]\!]^A = [\![P_1]\!]^A{}_\diamond[\![P_2]\!]^A$. In this equation, we have the abstract operation $\diamond$ that has to approximate the concrete composition of traces on the domain of abstract denotations. The best correct approximation of $^\frown$ in A is, by definition, the best choice defining $\diamond$: $[\![P_1]\!]^A{}_\diamond[\![P_2]\!]^A \stackrel{\text{def}}{=} \alpha(\gamma([\![P_1]\!]^A)^\frown\gamma([\![P_2]\!]^A))$. Note that we have $[\![P_1; P_2]\!]^A = \alpha([\![P_1; P_2]\!]) = \alpha([\![P_1]\!]^\frown[\![P_2]\!])$. If $\diamond$ is the best correct approximation of $^\frown$, then $\alpha([\![P_1]\!]^\frown[\![P_2]\!]) \leqslant [\![P_1]\!]^A{}_\diamond[\![P_2]\!]^A = \alpha(\gamma([\![P_1]\!]^A)^\frown\gamma([\![P_2]\!]^A))$. Because semantics can be modeled as abstract domains, we can think of formulating the problem of compositionality, in terms of closure operators, namely we would like to characterize the closure operators which describe compositional semantics. It is clear that the abstract semantics that makes the relation $\alpha([\![P_1]\!]^\frown[\![P_2]\!]) \leqslant \alpha(\gamma([\![P_1]\!]^A)^\frown\gamma([\![P_2]\!]^A))$ an equality satisfy the following equation: If X and Y are two sets of traces, representing the semantics of the components of a program, and ρ is a closure operator, representing an observable property of the semantics, then the corresponding semantics is compositional if

$$\text{(COMP)}\quad \rho(X^\frown Y) = \rho(\rho(X)^\frown\rho(Y))$$

where the concatenation operator, $^\frown$, is the canonical way of composing traces. This means that the equation (COMP) characterizes precisely the semantics that are compositional as regards as the concatenation of traces. Clearly, not all the semantics satisfy the condition (COMP). This is the case of the semantics observing a single state in a computation, e.g. the final or the initial state, as shown in the following section.

5.1. Forward/backward potential termination semantics

Consider a semantics which identifies the final states of finite traces, namely which considers only the states that are terminating in the traces of a given program. We call this semantics *forward potential termination semantics*. This observable is the dual of the *potential termination semantics* introduced in [10], here called *backward potential termination semantics*, observing the initial states of all the finite traces, namely which considers only those states which, potentially, lead to termination. Both semantics can be specified as abstractions of the natural trace semantics, by using a pair of adjoint functions:

$$\begin{array}{ll}
\alpha^{\vdash?} : \wp(\Sigma^\infty) \to \wp(\Sigma), & \alpha^{\vdash?}(X) = \{\sigma_\dashv \mid \sigma \in X^+\} \\
\gamma^{\vdash?} : \wp(\Sigma) \to \wp(\Sigma^\infty), & \gamma^{\vdash?}(Y) = \{\sigma \in \Sigma^+ \mid \sigma_\dashv \in Y\} \cup \Sigma^\omega \\
\alpha^{\dashv?} : \wp(\Sigma^\infty) \to \wp(\Sigma), & \alpha^{\dashv?}(X) = \{\sigma_\vdash \mid \sigma \in X^+\} \\
\gamma^{\dashv?} : \wp(\Sigma) \to \wp(\Sigma^\infty), & \gamma^{\dashv?}(Y) = \{\sigma \in \Sigma^+ \mid \sigma_\vdash \in Y\} \cup \Sigma^\omega
\end{array}$$

Proposition 5.1. $\langle \wp(\Sigma^\infty), \subseteq\rangle \underset{\alpha^{\vdash?}}{\overset{\gamma^{\vdash?}}{\leftrightarrows}} \langle \wp(\Sigma), \subseteq\rangle$ *and* $\langle \wp(\Sigma^\infty), \subseteq\rangle \underset{\alpha^{\dashv?}}{\overset{\gamma^{\dashv?}}{\leftrightarrows}} \langle \wp(\Sigma), \subseteq\rangle$.

Proof. The maps are trivially monotone, we prove that they form a Galois insertion.

$$\begin{aligned}\alpha^{\vdash?}\gamma^{\vdash?}(Y) &= \alpha^{\vdash?}\left(\left\{\sigma \in \Sigma^+ \,\middle|\, \sigma_\dashv \in Y\ \right\} \cup \Sigma^\omega\right)\\ &= \left\{\sigma_\dashv \,\middle|\, \sigma \in \left\{\delta \in \Sigma^+ \,\middle|\, \delta_\dashv \in Y\ \right\}\ \right\}\\ &= \left\{\sigma_\dashv \,\middle|\, \sigma_\dashv \in Y\ \right\} = Y\end{aligned}$$

$$\begin{aligned}\gamma^{\vdash?}\alpha^{\vdash?}(X) &= \gamma^{\vdash?}\left(\left\{\sigma_\dashv \,\middle|\, \sigma \in X^+\ \right\}\right)\\ &= \left\{\delta \in \Sigma^+ \,\middle|\, \delta_\dashv \in \left\{\sigma_\dashv \,\middle|\, \sigma \in X^+\ \right\}\ \right\} \cup \Sigma^\omega\\ &= \left\{\delta \in \Sigma^+ \,\middle|\, \exists\, \sigma \in X^+ \,.\, \delta_\dashv = \sigma_\dashv\ \right\} \cup \Sigma^\omega \supseteq X\end{aligned}$$

The other adjunction follows from the definition of potential termination semantics in [10]. □

We can define the corresponding closure operators as follows:

$$\begin{aligned}Pot^{\vdash?}(X) &\stackrel{\text{def}}{=} \gamma^{\vdash?}\alpha^{\vdash?}(X) = \left\{\sigma \in \Sigma^+ \,\middle|\, \exists\, \delta \in X^+ \,.\, \delta_\dashv = \sigma_\dashv\ \right\} \cup \Sigma^\omega\\ Pot^{\dashv?}(X) &\stackrel{\text{def}}{=} \gamma^{\dashv?}\alpha^{\dashv?} = \left\{\sigma \in \Sigma^+ \,\middle|\, \exists\, \delta \in X^+ \,.\, \delta_\vdash = \sigma_\vdash\ \right\} \cup \Sigma^\omega\end{aligned}$$

In the following we will identify with $\tau^{\vdash?}$ and $\tau^{\dashv?}$, respectively the semantics obtained with the operators just defined, namely $\tau^{\vdash?} \stackrel{\text{def}}{=} \alpha^{\vdash?}(\tau^\infty)$ and $\tau^{\dashv?} \stackrel{\text{def}}{=} \alpha^{\dashv?}(\tau^\infty)$.

The forward potential termination semantics is not adequate for modeling relational properties of trace semantics. This because the history of the computation is lost in forward potential termination semantics, and this information cannot be retrieved when it is required for composing semantics to get the semantics of program composition.

Example 5.2. Consider the forward potential termination semantics that observes the final states of finite computations only, and consider the program

$$P\left[\begin{array}{l} P_1\left[\begin{array}{l} x := 0;\\ \textbf{while } x \leqslant 3 \textbf{ do } x := x + 1; \end{array}\right.\\[2ex] P_2\left[\begin{array}{l} y := 0;\\ z := x + y; \end{array}\right. \end{array}\right.$$

In this context, the states of the program are identified with the triples of values in $\mathbb{N}$ that can be assigned to the corresponding variables of P. The concrete semantics of P is the set of all the finite traces of states, each one with a possible initial value for each variable:

$$\begin{aligned}[\![P]\!] = \{&\langle x, y, z\rangle \to \langle 0, y, z\rangle \to \langle 1, y, z\rangle \to \langle 2, y, z\rangle \to \langle 3, y, z\rangle\\ &\to \langle 4, y, z\rangle \to \langle 4, 0, z\rangle \to \langle 4, 0, 4\rangle \mid x, y, z \in \mathbb{N}\}\end{aligned}$$

On the other hand, the concrete semantics of P_1 and P_2 are

$$\begin{aligned}[\![P_1]\!] = {}& \{\langle x, y, z\rangle \to \langle 0, y, z\rangle \to \langle 1, y, z\rangle \to \langle 2, y, z\rangle \to \langle 3, y, z\rangle\\ &\to \langle 4, y, z\rangle \to \langle 4, y, z\rangle\}\\ [\![P_2]\!] = {}& \{\langle x, y, z\rangle \to \langle x, 0, z\rangle \to \langle x, 0, x\rangle\}\end{aligned}$$

again with $x, y, z \in \mathbb{N}$. Then the abstract semantics, denoted by $[\![\cdot]\!]^{\vdash?} \stackrel{\text{def}}{=} \alpha^{\vdash?}([\![\cdot]\!])$, are

$$\begin{aligned}
[\![P]\!]^{\vdash?} &= \{\langle 4, 0, 4\rangle\} \\
[\![P_1]\!]^{\vdash?} &= \{\langle 4, y, z\rangle \mid y, z \in \mathbb{N}\} \\
[\![P_2]\!]^{\vdash?} &= \{\langle x, 0, x\rangle \mid x \in \mathbb{N}\}
\end{aligned}$$

It is easy to observe that any trace with initial state $\langle 4, 5, 6\rangle$ and final state $\langle 10, 0, 10\rangle$ is in $\gamma^{\vdash?}([\![P_2]\!]^{\vdash?})$. Therefore, $\langle 10, 0, 10\rangle \in \alpha^{\vdash?}(\gamma^{\vdash?}([\![P_1]\!]^{\vdash?})^\frown\gamma^{\vdash?}([\![P_2]\!]^{\vdash?}))$. This fact proves that $Pot^{\vdash?}$ is not compositional, i.e., $[\![P]\!]^{\vdash?} \subset \alpha^{\vdash?}(\gamma^{\vdash?}([\![P_1]\!]^{\vdash?})^\frown\gamma^{\vdash?}([\![P_2]\!]^{\vdash?}))$.

As shown in Example 5.2 above, there are semantics which fail in modeling the input/output behavior of program traces. This information is not even captured by the independent composition of forward and backward potential termination semantics, as shown in the following example. In this case note that:

$$(Pot^{\vdash?} \sqcap Pot^{\dashv?})(X) = \left\{\sigma \in \Sigma^+ \;\middle|\; \exists\, \delta, \eta \in X : . \; \sigma_\vdash = \delta_\vdash \;\wedge\; \sigma_\dashv = \eta_\dashv \right\} \cup \Sigma^\omega$$

$Pot^{\vdash?} \sqcap Pot^{\dashv?}$ does not represent input/output relations. Indeed, there are traces that do not have necessarily the same initial and final state. $Pot^{\vdash?} \sqcap Pot^{\dashv?}$ includes the product of all the possible initial states, with all the possible final states of traces in X.

Example 5.3. Consider the program P_1 in Example 5.2. Let us denote by $s_0 \to^* s_n$ a trace with input state s_0 and output terminating state s_n. It is clear that both $\langle 5, 6, 7\rangle \to^* \langle 4, 6, 7\rangle$ and $\langle 10, 11, 12\rangle \to^* \langle 4, 11, 12\rangle$ are in $[\![P_1]\!]$. Therefore

$$\langle 5, 6, 7\rangle \to^* \langle 4, 11, 12\rangle \in Pot^{\vdash?} \sqcap Pot^{\dashv?}([\![P_1]\!])$$

which clearly fails to model input/output relations in P_1.

5.2. The reduced relative power

In the following sections, we apply the reduced relative power in order to derive compositional semantics systematically, starting from simpler and non-compositional ones. This operations is a well known method for refining abstract domain, by including relational (attribute dependent) information. In order to apply this operation, the concrete domain must be a *semi-quantale* [39], i.e., a structure $\langle D, \leqslant, \odot\rangle$ where $\langle D, \leqslant\rangle$ is a complete lattice and $\odot : D \times D \to D$ is an associative, monotone and left-adjoint operation.

The reduced relative power of two abstract domains of D, D_1 and D_2, is the set of all the monotone functions [30] $\lambda x.\alpha_2(d \odot \gamma_1(x))$ from D_1 to D_2, $D_1 \xrightarrow{\odot} D_2$, where d ranges over concrete values, γ_1 is the concretization map for D_1 and α_2 is the abstraction map for D_2.

Such functions are called dependences because they establish a dependency relation between the values of D_1 and the values of D_2. Moreover, the operation $\odot$ can be considered as a kind of combinator of the concrete denotations.

A dual operation can be defined simply applying the $\odot$ operation to the same elements but interchanged, namely we denote as $D_2 \xleftarrow{\odot} D_1$ the set of all monotone functions $\lambda x.\alpha_2(\gamma_1(x) \odot d)$ where the elements are the same defined before. In the following, we

will call, the first operation, *forward reduced power*, and the second one *backward reduced power*.

Theorem 5.4 (*Giacobazzi and Ranzato [30]*). *Let $\langle D, \leqslant, \odot \rangle$ be a semi-quantale, D_1 and D_2 be complete lattices, $\gamma_1 : D_1 \to D$ be a monotone function and $\langle D_2, \alpha_2, \gamma_2, D \rangle$ a Galois connection. The map $\alpha : D \to (D_1 \xrightarrow{\odot} D_2)$ defined as $\alpha(d) \stackrel{\text{def}}{=} \lambda x.\ \alpha_2(d \odot \gamma_1(x))$ is the left adjoint of a Galois insertion, namely there exists γ such that $\left\langle D_1 \xrightarrow{\odot} D_2, \alpha, \gamma, D \right\rangle$ is a GI.*

A dual theorem can be proved for the forward reduced relative power $D_2 \xleftarrow{\odot} D_1$.

In order to apply the reduced relative power to the abstractions of the maximal trace semantics, we need the following result.

Proposition 5.5. *$\langle \wp(\Sigma^\infty), \subseteq, \frown \rangle$ is a unitary quantale, with unity Σ.*

5.3. Systematic construction of the angelic denotational semantics

We can characterize the angelic denotational semantics as upper closure operator on the domain of finite and infinite traces (see Table 2 and [10]):

$$Ang^{\mathcal{D}}(X) = \left\{ \sigma \in \Sigma^+ \;\middle|\; \exists\, \delta \in X^+ \,.\, \sigma_\vdash = \delta_\vdash \ \wedge\ \sigma_\dashv = \delta_\dashv \right\} \cup \Sigma^\omega$$

The idea is that of obtaining the denotational angelic closure as the set of functional relations between the terminating states of finite computations. For this reason, we use the reduced relative power on the concrete domain $\wp(\Sigma^\infty)$, where the concrete combinator is the trace concatenation. Moreover, the domains involved in the operation coincide both with the forward potential termination semantics. Therefore we build the closure operator $Pot^{\vdash ?} \xleftarrow{\frown} Pot^{\vdash ?}$.

Proposition 5.6. *Let $\alpha^{D^+} : \wp(\Sigma^\infty) \to (\wp(\Sigma) \leftarrow \wp(\Sigma))$ be the map obtained by reduced power of backward potential termination semantics, $\alpha^{D^+}(X) = \lambda Y.\ \alpha^{\vdash ?}(\gamma^{\vdash ?}(Y) \frown X) = \lambda Y.\ \left\{ \sigma_\dashv \;\middle|\; \sigma \in X^+,\ \sigma_\vdash \in Y \right\}$. Then $\mathcal{D}^+ \cong \alpha^{D^+}(\tau^\infty)$.*

Proof. First of all we calculate the abstraction by using the backward reduced relative power. Consider the set $X \in \wp(\Sigma^\infty)$, then:

$$\begin{aligned}
\alpha^{D^+}(X) &= \lambda Y.\ \alpha^{\vdash ?}(\gamma^{\vdash ?}(Y) \frown X) \\
&= \lambda Y.\ \alpha^{\vdash ?}\left(\left(\left\{ \eta \in \Sigma^+ \;\middle|\; \eta_\dashv \in Y \right\} \frown X\right) \cup \Sigma^\omega\right) \\
&= \lambda Y.\ \alpha^{\vdash ?}\left(\left\{ \eta\delta \;\middle|\; \delta \in X,\ \delta_\vdash \in Y,\ \eta \in \Sigma^+ \right\} \cup \Sigma^\omega\right) \\
&= \lambda Y.\ \left\{ \sigma_\dashv \;\middle|\; \sigma \in \left\{ \eta\delta \;\middle|\; \delta \in X^+,\ \delta_\vdash \in Y,\ \eta \in \Sigma^+ \right\} \right\} \\
&= \lambda Y.\ \left\{ \sigma_\dashv \;\middle|\; \sigma \in X^+,\ \sigma_\vdash \in Y \right\}
\end{aligned}$$

Now it is immediate to verify that this abstraction is such that $\mathcal{D}^+ \cong \alpha^{D^+}(\tau^\infty)$. □

Theorem 5.7. *The angelic denotational semantics is the set of all the monotone functions between the elements of the forward potential termination, namely* $Pot^{\vdash ?} \overset{\frown}{\longleftarrow} Pot^{\vdash ?} = Ang^{\mathcal{D}}$.

Proof. We prove that the two functions $\alpha^{D^+}(X) = \lambda Y.\ \{\sigma_\dashv \mid \sigma \in X^+,\ \sigma_\vdash \in Y\}$ and $\gamma^{D^+}(f) = \{\sigma \in \Sigma^+ \mid \sigma_\dashv \in f(\sigma_\vdash)\} \cup \Sigma^\omega$ form a Galois connection. The monotonicity is trivial, so consider the following relations, where $f : \wp(\Sigma) \to \wp(\Sigma)$ and $X \in \wp(\Sigma^\infty)$:

$$\begin{aligned}
\alpha^{D^+}\gamma^{D^+}(f) &= \alpha^{D^+}\left(\{\sigma \in \Sigma^+ \mid \sigma_\dashv \in f(\sigma_\vdash)\} \cup \Sigma^\omega\right)\\
&= \lambda Y.\ \{\sigma_\dashv \mid \sigma \in \{\delta \in \Sigma^+ \mid \delta_\dashv \in f(\delta_\vdash)\}, \sigma_\vdash \in Y\}\\
&= \lambda Y.\ \{\sigma_\dashv \mid \sigma_\dashv \in f(Y)\}\\
&= \lambda Y.\ f(Y)\\
&= f
\end{aligned}$$

$$\begin{aligned}
\gamma^{D^+}\alpha^{D^+}(X) &= \gamma^{D^+}\left(\lambda Y.\ \{\delta_\dashv \mid \delta \in X^+,\ \delta_\vdash \in Y\}\right)\\
&= \{\sigma \in \Sigma^+ \mid \sigma_\dashv \in \{\delta_\dashv \mid \delta \in X^+,\ \delta_\vdash = \sigma_\vdash\}\} \cup \Sigma^\omega\\
&= \{\sigma \in \Sigma^+ \mid \exists\, \delta \in X^+ .\ \delta_\dashv = \sigma_\dashv,\ \delta_\vdash = \sigma_\vdash\} \cup \Sigma^\omega\\
&\supseteq X
\end{aligned}$$

Now, we consider the $Pot^{\vdash ?} \overset{\frown}{\longleftarrow} Pot^{\vdash ?} \overset{\text{def}}{=} \{\sigma \in \Sigma^+ \mid \exists\, \delta \in X^+ .\ \delta_\dashv = \sigma_\dashv,\ \delta_\vdash = \sigma_\vdash\} \cup \Sigma^\omega$ closure, that is clearly equal to the angelic denotational closure. □

This result tells us that the set of monotone functions between the terminating states of finite traces is exactly the set of the functions of the denotational angelic semantics of the transition system.

5.4. Optimality of the angelic denotational semantics

We can prove that the denotational angelic semantics is the most abstract semantics, more concrete than $Pot^{\vdash ?}$, which observes the final states of terminating computations. In order to show this fact, we can prove that the angelic denotational semantics is the solution of the abstract domain equation $X = Pot^{\vdash ?} \sqcap (X \overset{\frown}{\longleftarrow} X)$. This allows us to prove a result of optimality of the closure $Pot^{\vdash ?} \overset{\frown}{\longleftarrow} Pot^{\vdash ?}$. Namely, we prove that this semantics is the most abstract semantics which observe $Pot^{\vdash ?}$, and which is closed as regards $\overset{\frown}{\longleftarrow}$. In the following, we denote by $\vec{s}$ the constant function $\lambda x.\ s$.

Theorem 5.8. $(Pot^{\vdash ?} \overset{\frown}{\longleftarrow} Pot^{\vdash ?}) \overset{\frown}{\longleftarrow} (Pot^{\vdash ?} \overset{\frown}{\longleftarrow} Pot^{\vdash ?}) = Pot^{\vdash ?} \overset{\frown}{\longleftarrow} Pot^{\vdash ?}$.

Proof. We characterize $(Pot^{\vdash ?} \overset{\frown}{\longleftarrow} Pot^{\vdash ?}) \overset{\frown}{\longleftarrow} (Pot^{\vdash ?} \overset{\frown}{\longleftarrow} Pot^{\vdash ?})$ as a closure operator. We use the backward reduced relative power, over the domain $Pot^{\vdash ?} \overset{\frown}{\longleftarrow} Pot^{\vdash ?}$, in order to

build the function $\alpha : \wp(\Sigma^\infty) \to ((\wp(\Sigma) \to \wp(\Sigma)) \to (\wp(\Sigma) \to \wp(\Sigma)))$:

$$\begin{aligned}
\alpha(X) &= \lambda f.\, \alpha^{D^+}(\gamma^{D^+}(f)^\frown X) \\
&= \lambda f.\, \alpha^{D^+}\left(\left(\left\{\sigma \in \Sigma^+ \,\middle|\, \sigma_\dashv \in f(\sigma_\vdash)\ \right\}^\frown X\right) \cup \Sigma^\omega\right) \\
&= \lambda f.\, \alpha^{D^+}(\left\{\eta\delta \,\middle|\, \delta \in X,\ \eta \in \Sigma^+,\ \delta_\vdash \in f(\eta_\vdash)\ \right\} \cup\ \Sigma^\omega) \\
&= \lambda f.\lambda Y.\, \left\{\sigma_\dashv \,\middle|\, \sigma \in \left\{\eta\delta \,\middle|\, \begin{array}{l}\delta \in X^+,\ \eta \in \Sigma^+, \\ \delta_\vdash \in f(\eta_\vdash)\end{array}\right\}, \sigma_\vdash \in Y\right\} \\
&= \lambda f.\lambda Y.\, \left\{\sigma_\dashv \,\middle|\, \sigma \in X^+,\ \sigma_\vdash \in f(\eta_\vdash),\ \eta \in \Sigma^+,\ \eta_\vdash \in Y\ \right\} \\
&= \lambda f.\lambda Y.\, \left\{\sigma_\dashv \,\middle|\, \sigma \in X^+,\ \sigma_\vdash \in f(Y)\ \right\}
\end{aligned}$$

We know that this function is the left adjoint of a Galois insertion. Therefore let us consider the concretization $\gamma : ((\wp(\Sigma) \to \wp(\Sigma)) \to (\wp(\Sigma) \to \wp(\Sigma))) \to \wp(\Sigma^\infty)$, defined as

$$\gamma(g) = \left\{\sigma \in \Sigma^+ \,\middle|\, \forall\, X \in \wp(\Sigma)\,.\, \sigma_\dashv \in (g(\vec{\sigma}_\vdash))(X)\ \right\} \cup \Sigma^\omega$$

We prove that $\wp(\Sigma^\infty) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \alpha(\wp(\Sigma^\infty))$. Consider $g \in \alpha(\wp(\Sigma^\infty))$ defined as $g \stackrel{\text{def}}{=} \alpha(X)$ for some $X \in \wp(\Sigma^\infty)$.

$$\begin{aligned}
\alpha\gamma(g) &= \alpha\left(\left\{\delta \in \Sigma^+ \,\middle|\, \forall\, Z \in \wp(\Sigma)\,.\ \delta_\dashv \in (g(\vec{\delta}_\vdash))(Z)\ \right\} \cup \Sigma^\omega\right) \\
&= \lambda f.\lambda Y.\, \left\{\sigma_\dashv \,\middle|\, \begin{array}{l}\sigma \in \left\{\delta \in \Sigma^+ \,\middle|\, \begin{array}{l}\forall\, Z \in \wp(\Sigma)\,. \\ \delta_\dashv \in (g(\vec{\delta}_\vdash))(Z)\end{array}\right\} \\ \sigma_\vdash \in f(Y)\end{array}\right\} \\
&= \lambda f.\lambda Y.\, \left\{\sigma_\dashv \,\middle|\, \begin{array}{l}\sigma \in \Sigma^+,\ \forall Z \in \wp(\Sigma)\,.\, \sigma_\dashv \in (g(\vec{\sigma}_\vdash))(Z), \\ \sigma_\vdash \in f(Y)\end{array}\right\} \\
&= \lambda f.\lambda Y.\, \left\{\sigma_\dashv \,\middle|\, \begin{array}{l}\sigma \in \Sigma^+,\ \sigma_\vdash \in f(Y), \\ \sigma_\dashv \in \left\{\delta_\dashv \,\middle|\, \delta \in X^+,\ \delta_\vdash = \sigma_\vdash\ \right\}\end{array}\right\} \\
&= \lambda f.\lambda Y.\, \left\{\sigma_\dashv \,\middle|\, \exists\, \delta \in X^+\,.\, \sigma_\dashv = \delta_\dashv,\ \delta_\vdash = \sigma_\vdash \in f(Y)\ \right\} \\
&= \lambda f.\lambda Y.\, \left\{\sigma_\dashv \,\middle|\, \sigma \in X^+,\ \sigma_\vdash \in f(Y)\ \right\} \\
&= g
\end{aligned}$$

$$\begin{aligned}
\gamma\alpha(X) &= \gamma\left(\lambda f.\lambda Y.\, \left\{\sigma_\dashv \,\middle|\, \sigma \in X^+,\ \sigma_\vdash \in f(Y)\ \right\}\right) \\
&= \left\{\sigma \in \Sigma^+ \,\middle|\, \sigma_\dashv \in \left\{\delta_\dashv \,\middle|\, \delta \in X^+,\ \delta_\vdash = \sigma_\vdash\ \right\}\ \right\} \cup \Sigma^\omega \\
&= \left\{\sigma \in \Sigma^+ \,\middle|\, \exists\, \delta \in X^+\,.\, \delta_\dashv = \sigma_\dashv,\ \delta_\vdash = \sigma_\vdash\ \right\} \cup \Sigma^\omega \\
&\supseteq X
\end{aligned}$$

We can note that this closure is exactly the angelic denotational semantics. Hence we have $(Pot^{\vdash?} \overset{\frown}{\longleftarrow} Pot^{\vdash?}) \overset{\frown}{\longleftarrow} (Pot^{\vdash?} \overset{\frown}{\longleftarrow} Pot^{\vdash?})$ is the closure $Pot^{\vdash?} \overset{\frown}{\longleftarrow} Pot^{\vdash?}$. □

It is possible to conclude that the domain $Pot^{\vdash?} \overset{\frown}{\longleftarrow} Pot^{\vdash?}$ is the most abstract solution of the equation $X = Pot^{\vdash?} \sqcap X \overset{\frown}{\longleftarrow} X$, because it is trivial to prove that $Pot^{\vdash?} \sqsubseteq Pot^{\vdash?} \overset{\frown}{\longleftarrow}$

$Pot^{\vdash ?}$. This fact tells us that the domain $Pot^{\vdash ?} \overset{\frown}{\longleftarrow} Pot^{\vdash ?}$, which is the denotational angelic semantics, is the fixed point of refining process, starting from $Pot^{\vdash ?}$, by using $\overset{\frown}{\longleftarrow}$. So, this closure, is the most abstract one which observe the final states of finite traces and which is closed as regards the functional relations between these states.

5.5. Systematic construction of the liberal weakest precondition semantics

An analogous construction can be made for the $\mathcal{W}lp$ semantics, which is isomorphic to the denotational angelic one. This semantics can be defined as a closure operator [10] on the natural trace semantics, and it is equal to the angelic denotational semantics.

$$Wlp(X) = \left\{\sigma \in \Sigma^+ \;\middle|\; \exists\, \delta \in X^+ \,.\, \sigma_\vdash = \delta_\vdash,\ \sigma_\dashv = \delta_\dashv \right\} \cup \Sigma^\omega$$

As we have done for the denotational angelic semantics, we can build the set of the monotone functions between the states which lead to termination. In particular, we will use the forward reduced relative power on the concrete domain $\wp(\Sigma^\infty)$, with the trace concatenation. The involved abstract domains coincide, both, with the backward potential termination semantics, therefore the idea is to build the closure operator $Pot^{\dashv ?} \overset{\frown}{\longrightarrow} Pot^{\dashv ?}$.

Proposition 5.9. *Let $\alpha^{W^+} : \wp(\Sigma^\infty) \to (\wp(\Sigma) \leftarrow \wp(\Sigma))$ be the function obtained as reduced power of forward potential termination semantics, $\alpha^{W^+}(X) = \lambda Y.\, \alpha^{\dashv ?}(X \frown \gamma^{\dashv ?}(Y)) = \lambda Y.\ \left\{\sigma_\vdash \;\middle|\; \sigma \in X^+,\ \sigma_\dashv \in Y \right\}$. Then $\mathcal{W}lp \cong \alpha^{W^+}(\tau^\infty)$.*

Proof. First of all, we find the abstraction by using the forward reduced relative power. Analogous to Proposition 5.6 we can show that, if $X \in \wp(\Sigma^\infty)$, then:

$$\alpha^{W^+}(X) = \lambda Y.\, \alpha^{\dashv ?}(X \frown \gamma^{\dashv ?}(Y)) = \lambda Y.\ \left\{\sigma_\vdash \;\middle|\; \sigma \in X^+,\ \sigma_\dashv \in Y \right\}$$

Now we can prove that

$$\begin{aligned}
\mathcal{W}lp &= \alpha^{g^{\mathcal{W}p}}(\alpha^{\mathcal{D}}(\tau^+)) \\
&= \alpha^{g^{\mathcal{W}p}}\left(\left\{f \;\middle|\; f = \lambda\sigma_\vdash.\ \left\{\sigma_\dashv \;\middle|\; \sigma \in X\right\},\ X \in \wp(\Sigma^+)\right\}\right) \\
&= \left\{\varphi \;\middle|\; \varphi = \lambda P.\ \left\{\sigma_\vdash \;\middle|\; \left\{\sigma_\dashv \;\middle|\; \sigma \in X\right\} \subseteq P\right\},\ X \in \wp(\Sigma^+)\right\} \\
&= \left\{\varphi \;\middle|\; \varphi = \lambda P.\ \left\{\sigma_\vdash \;\middle|\; \sigma \in X,\ \sigma_\dashv \in P\right\},\ X \in \wp(\Sigma^+)\right\}
\end{aligned}$$

from which the thesis follows. □

Theorem 5.10. *The weakest-liberal precondition semantics is the set of the monotone function between the elements of the backward potential termination semantics, namely we have $Pot^{\dashv ?} \overset{\frown}{\longrightarrow} Pot^{\dashv ?} = Wlp$.*

Proof. Analogous to Theorem 5.7 we can prove that the function α^{W^+} defined in Proposition 5.9 and $\gamma^{W^+}(f) = \left\{\sigma \in \Sigma^+ \;\middle|\; \sigma_\vdash \in f(\sigma_\dashv)\right\} \cup \Sigma^\omega$ form a Galois insertion.

Moreover, the closure $Pot^{\dashv ?} \overset{\frown}{\longleftarrow} Pot^{\dashv ?} = \left\{\sigma \in \Sigma^+ \;\middle|\; \exists\, \delta \in X^+ \,.\, \delta_\dashv = \sigma_\dashv,\ \delta_\vdash = \sigma_\vdash \right\} \cup \Sigma^\omega$ is exactly the angelic denotational closure. □

5.6. Optimality of the weakest-liberal precondition semantics

The same property of optimality, which holds for the angelic denotational semantics, holds also for the liberal weakest precondition semantics.

Theorem 5.11. $(Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?}) \mathbin{\overset{\frown}{\longrightarrow}} (Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?}) = Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?}$.

Proof. We characterize $(Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?}) \mathbin{\overset{\frown}{\longrightarrow}} (Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?})$ as a closure operator by using the forward reduced relative power, on the just defined domain, similar to Theorem 5.8. First of all we obtain the function $\alpha : \wp(\Sigma^\infty) \to ((\wp(\Sigma) \to \wp(\Sigma)) \to (\wp(\Sigma) \to \wp(\Sigma)))$ as

$$\alpha(X) = \lambda f.\, \alpha^{W^+}(X^\frown\gamma^{W^+}(f)) = \lambda f.\lambda Y.\, \left\{\sigma_\vdash \,\middle|\, \sigma \in X^+,\ \sigma_\dashv \in f(Y)\right\}$$

We define the concretization function, and prove that it is the right adjoint of the abstraction α just defined. Let $\gamma : ((\wp(\Sigma) \to \wp(\Sigma)) \to (\wp(\Sigma) \to \wp(\Sigma))) \to \wp(\Sigma^\infty)$ be the function

$$\gamma(g) = \left\{\sigma \in \Sigma^+ \,\middle|\, \forall\, X \in \wp(\Sigma)\,.\, \sigma_\vdash \in (g(\vec{\sigma}_\dashv))(X)\right\} \cup \Sigma^\omega$$

and analogous to Theorem 5.8 we can prove that $\wp(\Sigma^\infty) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \alpha(\wp(\Sigma^\infty))$.

Finally note that $\gamma\alpha(X) = \left\{\sigma \in \Sigma^+ \,\middle|\, \exists\, \delta \in X^+\,.\, \delta_\vdash = \sigma_\vdash,\ \delta_\dashv = \sigma_\dashv\right\} \cup \Sigma^\omega$ is exactly the weakest-liberal precondition semantics. Hence, the closure $(Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?})$ is equal to $(Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?}) \mathbin{\overset{\frown}{\longrightarrow}} (Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?})$. □

This theorem tells us that the domain $Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?}$ is the most abstract solution of the equation $X = Pot^{\dashv?} \sqcap X \mathbin{\overset{\frown}{\longrightarrow}} X$, as it happens for $\mathcal{D}^+$. Even the liberal weakest precondition semantics is the fixed point of the refining process starting from $Pot^{\dashv?}$, by using $\mathbin{\overset{\frown}{\longrightarrow}}$. Namely also this semantics is the most abstract semantics which observes the initial states that can lead to termination, and which is closed as regards the functional relations between these states.

We can conclude that the semantics, obtained starting from the backward potential termination semantics, and starting from the forward potential termination, are the same closure operator, i.e.,

$$Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?} = Pot^{\vdash?} \mathbin{\overset{\frown}{\longleftarrow}} Pot^{\vdash?}$$

Remark 5.12. As shown in Example 5.3, the attribute independent composition of observables does not lead to compositional semantics. It is clear that $Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?} \sqsubseteq Pot^{\vdash?} \sqcap Pot^{\dashv?}$, namely that $Pot^{\dashv?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\dashv?}$ it is not the most abstract semantics, more concrete of both $Pot^{\vdash?}$ and $Pot^{\dashv?}$. Moreover, it can be easily verified that, by inverting the direction of the arrow in $Pot^{\vdash?} \mathbin{\overset{\frown}{\longleftarrow}} Pot^{\vdash?}$, we obtain the identity, namely it is immediate to prove the relation $Pot^{\vdash?} \mathbin{\overset{\frown}{\longrightarrow}} Pot^{\vdash?} = \iota_{\wp(\Sigma)}$. Intuitively, this happens because the forward reduced relative power of the forward termination semantics encodes how a given set X of concrete traces behaves when these are extended with any possible trace, ending in a given set of observable states Y. Hence, by observing the final states of these extended traces we get back Y. Instead, if we consider the initial states, in this construction, we can observe the

set of initial states of concrete traces that will have final states in Y. This is precisely Dijkstra's weakest precondition semantics *Wlp*, as proved in Theorem 5.10. A similar reasoning holds if we dualize $Pot^{\dashv?} \xrightarrow{\frown} Pot^{\dashv?}$.

5.7. Compositional angelic semantics

In this section, we prove that both the denotational and the liberal weakest precondition semantics, are the most abstract semantics on $\wp(\Sigma^\infty)$ observing, respectively, final and initial states, and which are compositional, i.e., solutions of the equation (COMP) above.

Theorem 5.13 (*Giacobazzi et al. [31]*). *The most abstract solution of* $\rho(X^\frown Y) = \rho(\rho(X)^\frown \rho(Y))$ *on* $uco(C)$ *is* $\rho = \rho \sqcap (\rho \xrightarrow{\frown} \iota_C) \sqcap (\iota_C \xleftarrow{\frown} \rho) \sqcap ((\iota_C \xrightarrow{\frown} \rho) \xleftarrow{\frown} \iota_C)$.

We have to prove that the closure $Pot^{\vdash?} \xleftarrow{\frown} Pot^{\vdash?} = Pot^{\dashv?} \xrightarrow{\frown} Pot^{\dashv?}$ is the most abstract compositional semantics definable on the set of maximal traces, which includes respectively, $Pot^{\vdash?}$ and $Pot^{\dashv?}$, as an abstract interpretation. In the following, we will denote by ι the identical closure $\iota_{\wp(\Sigma^\infty)}$, and by Theorems 5.7 and 5.10 $Ang^{\mathcal{D}} = Pot^{\vdash?} \xleftarrow{\frown} Pot^{\vdash?} = Pot^{\dashv?} \xrightarrow{\frown} Pot^{\dashv?}$.

Lemma 5.14. (i) $\iota \xrightarrow{\frown} Ang^{\mathcal{D}} = Ang^{\mathcal{D}}$.
(ii) $Ang^{\mathcal{D}} \xleftarrow{\frown} \iota = Ang^{\mathcal{D}}$.

Proof. (i) We can use the forward reduced relative power for building the closure corresponding to the semantics $\iota \xrightarrow{\frown} Ang^{\mathcal{D}}$. Let $X \in \wp(\Sigma^\infty)$, and consider, in particular $Ang^{\mathcal{D}} = Pot^{\dashv?} \xrightarrow{\frown} Pot^{\dashv?}$:

$$\begin{aligned}\alpha(X) &= \lambda Y.\, \alpha^{W^+}(X^\frown Y)\\ &= \lambda Y.\lambda Z.\, \left\{\sigma_\vdash \,\middle|\, \sigma \in X^{+\frown} Y^+,\ \sigma_\dashv \in Z\right\}\end{aligned}$$

note that $\alpha : \wp(\Sigma^\infty) \to ((\wp(\Sigma^\infty) \times \wp(\Sigma))) \to \wp(\Sigma))$, and that it is the left adjoint of a GI. Consider the function $\gamma(g) : ((\wp(\Sigma^\infty) \times \wp(\Sigma))) \to \wp(\Sigma^\infty)$ defined as

$$\gamma(g) = \left\{\sigma \in \Sigma^+ \,\middle|\, \sigma_\vdash \in g(\Sigma, \sigma_\dashv)\right\} \cup \Sigma^\omega$$

we prove that $\wp(\Sigma^\infty) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \alpha(\wp(\Sigma^\infty))$. Let $g \in \alpha(\wp(\Sigma^\infty))$ be such that $g = \alpha(X)$, we can compute

$$\begin{aligned}\alpha\gamma(g) &= \alpha(\left\{\sigma \in \Sigma^+ \,\middle|\, \sigma_\vdash \in g(\Sigma, \sigma_\dashv)\right\} \cup \Sigma^\omega)\\ &= \lambda Y.\lambda Z.\, \left\{\sigma_\vdash \,\middle|\, \sigma \in \left\{\delta \in \Sigma^+ \,\middle|\, \delta_\vdash \in g(\Sigma, \delta_\dashv)\right\}^\frown Y^+,\ \sigma_\dashv \in Z\right\}\\ &= \lambda Y.\lambda Z.\, \left\{\sigma_\vdash \,\middle|\, \sigma \in \left\{\delta\eta \,\middle|\, \begin{array}{l}\delta_\vdash \in g(\Sigma, \delta_\dashv),\\ \delta_\dashv\eta \in Y^+\end{array}\right\},\ \sigma_\dashv \in Z\right\}\\ &= \lambda Y.\lambda Z.\, \left\{\delta_\vdash \,\middle|\, \delta_\vdash \in g(\Sigma, \delta_\dashv),\ \delta_\dashv\eta \in Y^+,\ \eta_\dashv \in Z\right\}\end{aligned}$$

$$= \lambda Y.\lambda Z.\ \left\{ \delta_{\vdash} \;\middle|\; \begin{array}{l} \delta_{\vdash} \in \left\{ \sigma_{\vdash} \;\middle|\; \sigma \in X^{+},\ \sigma_{\dashv} = \delta_{\dashv} \right\}, \\ \delta_{\dashv}\eta \in Y^{+},\ \eta_{\dashv} \in Z \end{array} \right\}$$

$$= \lambda Y.\lambda Z.\ \left\{ \delta_{\vdash} \;\middle|\; \begin{array}{l} \exists\, \sigma \in X^{+}\,.\, \sigma_{\vdash} = \delta_{\vdash},\ \sigma_{\dashv} = \delta_{\dashv}, \\ \delta_{\dashv}\eta \in Y^{+},\ \eta_{\dashv} \in Z \end{array} \right\}$$

$$= \lambda Y.\lambda Z.\ \left\{ \sigma_{\vdash} \;\middle|\; \sigma \in X^{+},\ \sigma_{\dashv}\eta \in Y^{+},\ \eta_{\dashv} \in Z \right\}$$

$$= \lambda Y.\lambda Z.\ \left\{ \delta_{\vdash} \;\middle|\; \delta \in X^{+\frown}Y^{+},\ \delta_{\dashv} \in Z \right\} = g$$

$$\gamma\alpha(X) = \gamma\left(\lambda Y.\lambda Z.\ \left\{ \delta_{\vdash} \;\middle|\; \delta \in X^{+\frown}Y^{+},\ \delta_{\dashv} \in Z \right\}\right)$$

$$= \left\{ \sigma \in \Sigma^{+} \;\middle|\; \sigma_{\vdash} \in \left\{ \delta_{\vdash} \;\middle|\; \delta \in X^{+},\ \delta_{\dashv} = \sigma_{\dashv} \right\} \right\} \cup \Sigma^{\omega}$$

$$= \left\{ \sigma \in \Sigma^{+} \;\middle|\; \exists\, \delta \in X^{+}\,.\, \sigma_{\vdash} = \delta_{\vdash},\ \delta_{\dashv} = \sigma_{\dashv} \right\} \cup \Sigma^{\omega} \supseteq X$$

It is immediate to prove that $\iota \xrightarrow{\frown} Ang^{\mathcal{D}} = \gamma\alpha = Ang^{\mathcal{D}}$.

(ii) Consider $X \in \wp(\Sigma^{\infty})$, and $Ang^{\mathcal{D}} = Pot^{\vdash?} \xleftarrow{\frown} Pot^{\vdash?}$

$$\alpha(X) = \lambda Y.\ \alpha^{D^{+}}(Y^{\frown}X)$$

$$= \lambda Y.\lambda Z.\ \left\{ \sigma_{\dashv} \;\middle|\; \sigma \in Y^{+\frown}X^{+},\ \sigma_{\vdash} \in Z \right\}$$

where $\alpha : \wp(\Sigma^{\infty}) \to ((\wp(\Sigma^{\infty}) \times \wp(\Sigma))) \to \wp(\Sigma))$. Consider the function $\gamma(g) : ((\wp(\Sigma^{\infty}) \times \wp(\Sigma))) \to \wp(\Sigma^{\infty})$, defined as

$$\gamma(g) = \left\{ \sigma \in \Sigma^{+} \;\middle|\; \sigma_{\dashv} \in g(\Sigma, \sigma_{\vdash}) \right\} \cup \Sigma^{\omega}$$

We can prove that the two functions form Galois insertion showing that the following fact holds: $\wp(\Sigma^{\infty}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \alpha(\wp(\Sigma^{\infty}))$. Consider $g \in \alpha(\wp(\Sigma^{\infty}))$, such that $g = \alpha(X)$, and

$$\alpha\gamma(g) = \alpha\left(\left\{ \sigma \in \Sigma^{+} \;\middle|\; \sigma_{\dashv} \in g(\Sigma, \sigma_{\vdash}) \right\} \cup \Sigma^{\omega}\right)$$

$$= \lambda Y.\lambda Z.\ \left\{ \sigma_{\dashv} \;\middle|\; \begin{array}{l} \sigma \in Y^{+\frown}\left\{ \delta \in \Sigma^{+} \;\middle|\; \delta_{\dashv} \in g(\Sigma, \delta_{\vdash}) \right\}, \\ \sigma_{\vdash} \in Z \end{array} \right\}$$

$$= \lambda Y.\lambda Z.\ \left\{ \sigma_{\dashv} \;\middle|\; \sigma \in \left\{ \eta\delta \;\middle|\; \eta\delta_{\vdash} \in X^{+},\ \delta_{\dashv} \in g(\Sigma, \delta_{\vdash}) \right\}, \sigma_{\vdash} \in Z \right\}$$

$$= \lambda Y.\lambda Z.\ \left\{ \delta_{\dashv} \;\middle|\; \delta_{\dashv} \in g(\Sigma, \delta_{\vdash}),\ \eta\delta_{\vdash} \in Y^{+},\ \eta_{\vdash} \in Z \right\}$$

$$= \lambda Y.\lambda Z.\ \left\{ \delta_{\dashv} \;\middle|\; \begin{array}{l} \delta_{\dashv} \in \left\{ \sigma_{\dashv} \;\middle|\; \sigma \in X^{+},\ \sigma_{\vdash} = \delta_{\vdash} \right\}, \\ \eta\delta_{\vdash} \in Y^{+},\ \eta_{\vdash} \in Z \end{array} \right\}$$

$$= \lambda Y.\lambda Z.\ \left\{ \delta_{\dashv} \;\middle|\; \begin{array}{l} \exists\, \sigma \in X^{+}\,.\, \sigma_{\vdash} = \delta_{\vdash},\ \sigma_{\dashv} = \delta_{\dashv}, \\ \eta\delta_{\vdash} \in Y^{+},\ \eta_{\vdash} \in Z \end{array} \right\}$$

$$= \lambda Y.\lambda Z.\ \left\{ \sigma_{\dashv} \;\middle|\; \sigma \in X^{+},\ \eta\sigma_{\vdash} \in Y^{+},\ \eta_{\vdash} \in Z \right\}$$

$$= \lambda Y.\lambda Z.\ \left\{ \delta_{\dashv} \;\middle|\; \delta \in Y^{+\frown}X^{+},\ \delta_{\vdash} \in Z \right\} = g$$

$$\begin{aligned}\gamma\alpha(X) &= \gamma\left(\lambda Y.\lambda Z.\ \left\{\delta_{\dashv} \,\middle|\, \delta \in Y^{+}\frown X^{+},\ \delta_{\vdash} \in Z\ \right\}\right)\\ &= \left\{\sigma \in \Sigma^{+} \,\middle|\, \sigma_{\dashv} \in \left\{\delta_{\dashv} \,\middle|\, \delta \in X^{+},\ \delta_{\vdash} = \sigma_{\vdash}\ \right\}\right\} \cup \Sigma^{\omega}\\ &= \left\{\sigma \in \Sigma^{+} \,\middle|\, \exists\, \delta \in X^{+}\,.\,\sigma_{\vdash} = \delta_{\vdash},\ \delta_{\dashv} = \sigma_{\dashv}\ \right\} \cup \Sigma^{\omega} \supseteq X\end{aligned}$$

It is clear that $Ang^{\mathcal{D}} \overset{\frown}{\longleftarrow} \iota = \gamma\alpha = Ang^{\mathcal{D}}$. □

By Theorem 5.13 and Lemma 5.14, the following result is straightforward, and implies the optimality of the denotational and weakest precondition semantics. Namely, they are the most abstract semantics which are compositional as regards the trace concatenation.

Theorem 5.15. *For any* $X, Y \in \Sigma^{\infty}$:
- $Ang^{\mathcal{D}}(X\frown Y) = Ang^{\mathcal{D}}(Ang^{\mathcal{D}}(X)\frown Ang^{\mathcal{D}}(Y))$;
- $\rho(X\frown Y) = \rho(\rho(X)\frown\rho(Y)) \ \wedge\ \rho \sqsubseteq Pot^{\vdash ?} \ \Rightarrow\ \rho \sqsubseteq Ang^{\mathcal{D}}$.

6. The equational hierarchy of semantics

In the previous sections we derived the angelic compositional semantics as solutions of domain equations. Note that, by the definition of the semantics in the Cousot's hierarchy of semantics [10], while all the closures representing all the semantics more abstract than the relational one are all the same, the abstractions are different and considers different aspects of computation. For this reason we have to use the backward reduced relative power for obtaining the denotational abstraction, which is isomorphic to the relational one. While we have to use the forward reduced relative power for deriving the weakest-liberal precondition, and therefore the isomorphic partial correctness semantics. In particular, we obtain the angelic denotational/relational semantics as the backward reduced relative power of the semantics observing terminating states. In the same way, we derived the weakest-liberal precondition/partial-correctness semantics as forward reduced relative power of the semantics observing states that potentially lead to termination.

Moreover, in [26] we derived, in a very similar way, the equational representation of all the natural compositional semantics, i.e., denotational/relational and weakest precondition/Hoare's axiomatic. More precisely, the equational representation is obtained in a more concrete level, of the hierarchy of semantics: the transfinite one. In this level of abstraction, all the semantics are able to observe the transfinite behavior of programs, namely computations whose length is a generic ordinal. In this way we can distinguish also traces that leads to non-termination, characterizing which ordinal characterizes the infinity of the computation. Only by using this concrete semantics, we can use the reduced relative power operation in order to derive compositional semantics:

$$\begin{aligned} Den &= \alpha^{\propto}(X) \qquad \text{s.t. } X = Pot^{\vdash} \sqcap X \overset{\frown}{\longleftarrow} X\\ gWp &= \alpha^{\propto}(X) \qquad \text{s.t. } X = Pot^{\dashv} \sqcap X \overset{\frown}{\longrightarrow} X\end{aligned}$$

where $\alpha^{\propto}$ forgets the transfinite behavior collecting all the computation leading to non-termination, by abstracting non-terminating traces to $\bot$, while $Pot^{\vdash}$ and $Pot^{\dashv}$ are the transfinite version of, respectively, $Pot^{\vdash ?}$ and $Pot^{\dashv ?}$ [26], i.e., they observe, respectively, initial and final states of traces with a fixed ordinal length.

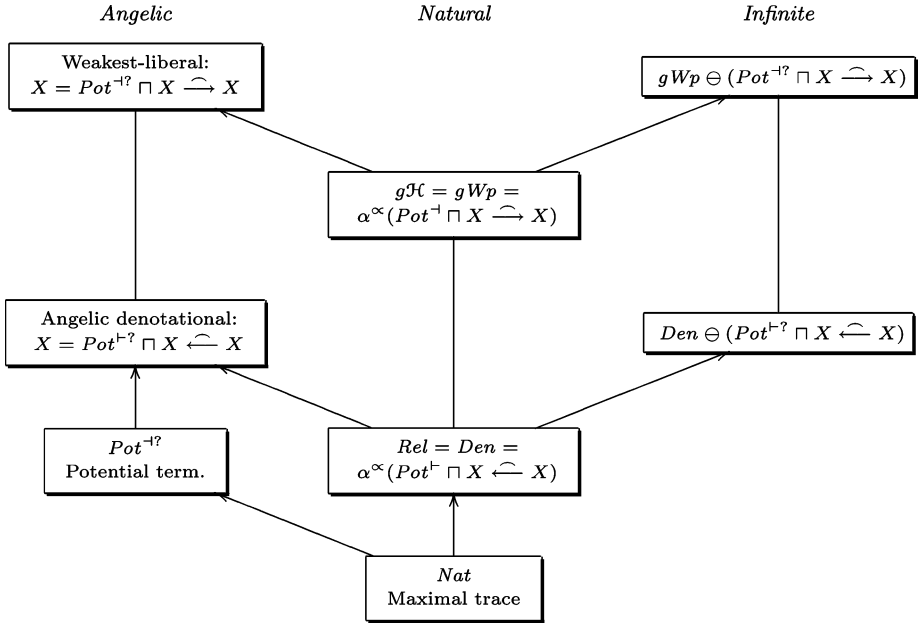

Fig. 9. Semantics as abstract domain equations.

Finally, we can combine the results described in Section 4.4 with the results described in Section 5.7, in order to obtain the equational representation also of the infinite semantics, as complements of the angelic semantics in the natural ones. In this way we derive the global picture depicted in Fig. 9.

7. Systematic design of semantics for concurrent constraint languages

In this section, we consider, as example, the case of *concurrent constraint programming* languages [40] and we derive their denotational closure-based [41], and axiomatic semantics [16], as an abstract interpretations of the maximal traces of constraints in a transition system semantics.

Concurrent constraint programming (*ccp* for short) is a well known concurrent programming paradigm where processes interact through a common store [40]. This leads to a computational model based on the notion of *store-as-constraint*. The main features of a concurrent constraint process is to refine the store (tell-constraints) or synchronize itself with other processes (ask-constraints). The *ask–tell* paradigm, which is the basis of *ccp* languages, is an extension of constraint logic programming: In addition to satisfiability (tell), *entailment* (ask) is introduced. A store is a constraint representing the global state of the computation. Synchronization is achieved through *blocking ask*: a process is suspended when the store does not entail the ask constraint, and it remains suspended until the store entails it.

The constraint system represents the basic algebraic notion behind *ccp*. The construction, in [41], is an extension of Scott's partial information systems [1]. Informally, we have a countable set D of elementary assertions (containing distinct elements **1** and **0** representing the least informative assertion and the contradiction, respectively), and a finite entailment relation $\vdash \subseteq \wp_f(D \times D)$. A *simple constraint system* is $\Sigma \stackrel{\text{def}}{=} \langle \wp(D), \dashv \rangle /_\sim$, which is a complete ω-algebraic lattice [1] where $X \sim Y$ iff $(X)^\dashv = (Y)^\dashv$, being $(X)^\dashv$ the entailment closure of a set of assertions X. An arbitrary element of Σ is called a *constraint*. Compact elements are called *finite constraints*, since they are equivalent to a finite subset of D. In order to treat the hiding operator of the language, Saraswat et al. [41] introduce a family of unary operations called *cylindrifications* (see [33]). Intuitively, given a constraint c, the cylindrification operation $\exists_x(c)$ yields the constraint obtained by "projecting out" from c the information about the variable x. *Diagonal elements* (equational constraints between variables [33]) are considered as a way to provide parameter passing. Note that special variables (not accessible to the user) together with a suitable use of cylindrification and diagonal elements make variable renaming no longer needed [41].

Definition 7.1. A *constraint system* $\langle \Sigma, \vdash, \mathit{false}, \mathit{true}, \wedge, V, \exists_x, d_{xy} \rangle$ is a structure where: $\langle \Sigma, \dashv \rangle$ is a simple constraint system, $\mathit{true} = [\mathbf{1}]_\sim$ and $\mathit{false} = [\mathbf{0}]_\sim$, $\wedge$ is the *glb*, V is a denumerable set of variables, and $\forall x, y \in V, \forall c, c' \in \Sigma$, the operator $\exists_x : \Sigma \to \Sigma$ satisfies
1. $c \vdash \exists_x c$,
2. if $c \vdash c'$ then $\exists_x c \vdash \exists_x c'$,
3. $\exists_x(c \wedge \exists_x c') = \exists_x c \wedge \exists_x c'$,
4. $\exists_x(\exists_y c) = \exists_y(\exists_x c)$.

$\forall x, y, z \in V, \forall c \in \Sigma$, the diagonal element d_{xy} satisfies
1. $d_{xx} = true$.
2. if $z \neq x, y$ then $d_{xy} = \exists_z(d_{xz} \wedge d_{zy})$,
3. if $x \neq y$ then $d_{xy} \wedge \exists_x(c \wedge d_{xy}) \vdash c$.

The semantic operators of concurrent constraint languages are: elementary actions (**ask** and **tell**), hiding ($\exists$), parallel composition ($\|$), guarded non-deterministic choice ($\sum$) and recursion. The semantics of *ccp* programs well fits into Cousot's hierarchy being easily described as maximal *consistent* traces of a transition system, i.e., maximal traces $\langle A_0, c_0 \rangle \longrightarrow_\tau \langle A_1, c_1 \rangle \longrightarrow_\tau \ldots$ where A_i are agents and $c_i \vdash c_{i-1}$ are constraints. We denote by $(\mathit{Agent} \times \Sigma)^\infty_\Gamma$ this set of traces. The standard syntax and transition-system semantics is in Table 3. The maximal-trace semantics of a *ccp* program $P = D.A$ is immediately defined as the set of finite and infinite consistent traces of constraints generated from P in an initial store $c \in \Sigma$.

$$\mathcal{O}(D.A)(c) = \left\{ \delta \in (\mathit{Agent} \times \Sigma)^\infty_\Gamma \;\middle|\; \begin{array}{l} \delta_\vdash = \langle A, c \rangle, \forall i \in [0, |\delta|) \\ \delta_i \longrightarrow_\tau \delta_{i+1} \end{array} \right\}$$

We define the maximal trace semantics of a *ccp* program P as follows:

$$[\![P]\!]_\infty = \beta\left(\left\{ \mathcal{O}(P)(c) \;\middle|\; c \in \Sigma \right\}\right)$$

where the function $\beta \in \wp((\mathit{Agent} \times \Sigma)^\infty) \xrightarrow{a} \wp(\Sigma^\infty)$ abstracts away the agent information from traces: $\beta(X) = \left\{ \sigma \in \Sigma^\infty \;\middle|\; \delta \in X, \delta_i = \langle A, \sigma_i \rangle \right\}$. The following result characterizes

Table 3
The syntax and operational semantics of *ccp*

$$
\begin{array}{lll}
\text{Program} & ::= & \text{Dec} \,.\, \text{Agent} \\
\text{Dec} & ::= & \varepsilon \\
 & | & p(\vec{x}) \ :- \ \text{Agent} \,.\, \text{Dec} \\
\text{Agent} & ::= & \mathbf{tell}(c) \\
 & | & \exists \vec{x}.\text{Agent} \\
 & | & \text{Agent} \parallel \text{Agent} \\
 & | & \sum_{i=1}^{n} (\mathbf{ask}(c_i) \rightarrow \text{Agent}_i) \\
 & | & p(\vec{y})
\end{array}
$$

R1 $\langle \mathbf{tell}(c)\,,\ \sigma\rangle \longrightarrow_{\mathcal{T}} \langle \varepsilon\,,\ \sigma \wedge c\rangle$

R2 $$\frac{\sigma \vdash c_i}{\left\langle \sum_{i=1}^{n} (\mathbf{ask}(c_i) \rightarrow A_i)\,,\ \sigma \right\rangle \longrightarrow_{\mathcal{T}} \langle A_i\,,\ \sigma\rangle}$$

R3 $$\frac{\langle A\,,\ \sigma\rangle \longrightarrow_{\mathcal{T}} \langle A'\,,\ \sigma'\rangle}{\begin{array}{l}\langle A \| B\,,\ \sigma\rangle \longrightarrow_{\mathcal{T}} \langle A' \| B\,,\ \sigma'\rangle \\ \langle B \| A\,,\ \sigma\rangle \longrightarrow_{\mathcal{T}} \langle B \| A'\,,\ \sigma'\rangle\end{array}}$$

R4 $$\frac{\langle A\,,\ d \wedge \exists_{\vec{x}}\sigma\rangle \longrightarrow_{\mathcal{T}} \langle B\,,\ e\rangle}{\langle \exists(\vec{x}, d).A\,,\ \sigma\rangle \longrightarrow_{\mathcal{T}} \langle \exists(\vec{x}, e).B\,,\ \sigma \wedge \exists_{\vec{x}} e\rangle}$$

R5 $$\frac{p(\vec{x}) \ :- \ A \in P}{\langle p(\vec{y})\,,\ \sigma\rangle \longrightarrow_{\mathcal{T}} \langle \exists(\vec{x}, d_{\vec{x},\vec{y}}).A\,,\ \sigma\rangle}$$

both, the closure-based denotational [41], and the predicate-transformer semantics of nondeterministic *ccp* programs $P \in \mathit{Program}$, in [16], as abstract interpretations of the maximal trace semantics $[\![P]\!]_\infty$. In particular these semantics can be both systematically derived from a non-compositional semantics observing, respectively, final and initial constraints in computational traces. In this case, $\alpha^{\vdash?}([\![P]\!]_\infty)$ and $\alpha^{\dashv?}([\![P]\!]_\infty)$ are, respectively, the forward and backward potential termination semantics of P.

Theorem 7.2. *Let P be a ccp program.*
- $\mathit{Ang}^{\mathcal{D}}([\![P]\!]_\infty) = \lambda X.\ \bigcup \{\alpha^{\mathcal{D}}(\alpha^{\mathcal{R}}([\![P]\!]_\infty))(c) \mid c \in X\}$ *is a linear continuous closure operator on Smith's powerdomain* $\wp(\Sigma_\perp)$*, with* $\perp$ *representing divergence.*
- $\mathit{Wlp}([\![P]\!]_\infty) = \lambda X.\ \bigcup \{c \mid \alpha^{\mathcal{D}}(\alpha^{\mathcal{R}}([\![P]\!]_\infty))(c) \subseteq X\}$ *is a co-additive function on* $\wp(\Sigma_\perp)$ *and its left adjoint function is* $\lambda X.\ \bigcup \{[\![P]\!]_{\mathcal{D}}(c) \mid c \in X\}$.

Proof. By a straightforward inductive argument, it is easy to prove that any trace in $\beta([\![P]\!]_\infty)$ is consistent, i.e., it refines constraints. This proves that $\mathit{Ang}^{\mathcal{D}}([\![P]\!]_\infty) = \lambda X.\ \bigcup \{\alpha^{\mathcal{D}}(\alpha^{\mathcal{R}}([\![P]\!]_\infty))(c) \mid c \in X\}$ is reductive on the Smith's powerdomain $\wp(\Sigma_\perp)$, ordered by $\vdash$. Monotonicity is trivial, while idempotence comes directly because only final terminating constraints (i.e., resting points [41]) are considered in $\alpha^{\vdash?}([\![P]\!]_\infty)$ and $\mathit{Ang}^{\mathcal{D}} = \mathit{Pot}^{\vdash?} \overleftarrow{\ } \mathit{Pot}^{\vdash?}$. As far as the strongest postcondition semantics is concerned, it is immediate to prove, by construction, that the weakest-liberal precondition semantics $\mathit{Wlp}([\![P]\!]_\infty) = \lambda X.\ \bigcup \{c \mid \alpha^{\mathcal{D}}(\alpha^{\mathcal{R}}([\![P]\!]_\infty))(c) \subseteq X\}$ is co-additive with left adjoint function $\lambda X.\ \bigcup \{[\![P]\!]_{\mathcal{D}}(c) \mid c \in X\}$. □

We define $[\![P]\!]_{\mathcal{D}} \stackrel{\text{def}}{=} \mathit{Ang}^{\mathcal{D}}([\![P]\!]_\infty)$ and $[\![P]\!]_{\mathit{Wlp}} \stackrel{\text{def}}{=} \mathit{Wlp}([\![P]\!]_\infty)$. By Theorem 7.2, they correspond, respectively, to the closure-based denotational semantics, in [41], and to the strongest postcondition semantics, in [16].

We close this section by considering examples of programs, with their denotational semantics, for the different observable behaviors, corresponding to the different complemen-

tary semantics in Cousot's hierarchy, namely the angelic, demonic, slothful, and infinite semantics. These semantics are all abstractions of $[\![\cdot]\!]_{\mathcal{D}}$. Consider the following programs, where $\mathbf{ask}(true) \to A$ is denoted A and the starting agent is underlined.

$$\begin{array}{ll} P: & p(x) :- \mathbf{tell}(x = 2) + q(x). \\ & q(x) :- p(x).\ \underline{p(x)} \\ Q: & p(x) :- \mathbf{tell}(x = 2).\ \underline{p(x)} \\ U: & p(x) :- \mathbf{tell}(x = 1) + q(x). \\ & q(x) :- p(x).\ \underline{p(x)} \end{array}$$

It is immediate that $[\![P]\!]_{\mathcal{D}^+} = [\![Q]\!]_{\mathcal{D}^+}$, since the two programs have the same set of finite output constraints, although P generates an infinite sequence from the same input constraint, indeed, $[\![P]\!]_{\mathcal{D}^\omega} \neq [\![Q]\!]_{\mathcal{D}^\omega}$. These facts imply that $[\![P]\!]_{\mathcal{D}^\partial} \neq [\![Q]\!]_{\mathcal{D}^\partial}$ and that $[\![P]\!]_{\mathcal{D}^\wp} \neq [\![Q]\!]_{\mathcal{D}^\wp}$. Consider now the programs P and U, then $[\![P]\!]_{\mathcal{D}^+} \neq [\![U]\!]_{\mathcal{D}^+}$, since they have different finite output constraints, moreover, $[\![P]\!]_{\mathcal{D}^\omega} = [\![U]\!]_{\mathcal{D}^\omega}$, since they have also the same infinite sequence starting from the same input constraint. This fact implies that $[\![P]\!]_{\mathcal{D}^\partial} = [\![U]\!]_{\mathcal{D}^\partial}$, since the demonic closure adds all the possible finite output from the same initial constraint, and for the same reason we have $[\![P]\!]_{\mathcal{D}^\wp} = [\![U]\!]_{\mathcal{D}^\wp}$. As far as the demonic, slothful and infinite semantics are concerned, consider the following programs:

$$\begin{array}{ll} H: & p(x) :- \mathbf{ask}(x = 1) \to q(x). \\ & q(x) :- q(x).\ \underline{p(x)} \\ K: & p(x) :- \mathbf{ask}(x = 1) \to q(x) + \\ & \qquad \mathbf{ask}(x > 1) \to \mathbf{tell}(y = 2 * x). \\ & q(x) :- q(x).\ \underline{p(x)} \end{array}$$

In this case, $[\![H]\!]_{\mathcal{D}^\omega} = [\![K]\!]_{\mathcal{D}^\omega}$, because they generate the same infinite sequence starting from the same input constraint, but $[\![H]\!]_{\mathcal{D}^\partial} \neq [\![K]\!]_{\mathcal{D}^\partial}$ because H can only stop or generate an infinite sequence, therefore chaos, while there exist input constraints ($x > 1$) such that K terminates without generating chaos. Finally, consider the following programs, where we assume that Σ is a finite domain **FD** constraint system (see [34]), and A is a terminating agent that, given $c \in \Sigma$, generates all the $c' \in \Sigma$ such that $c' \vdash c$:

$$\begin{array}{ll} G: & p(x) :- p(x) + A.\ \underline{p(x)} \\ R: & p(x) :- A.\ \underline{p(x)} \end{array}$$

then $[\![G]\!]_{\mathcal{D}^\partial} \neq [\![R]\!]_{\mathcal{D}^\partial}$, since G generates an infinite trace, while R cannot, but $[\![G]\!]_{\mathcal{D}^\wp} = [\![R]\!]_{\mathcal{D}^\wp}$, because both the programs generate the whole chaos, starting from the same initial point, G generates it for the infinite sequence and R, by its definition, generates a chaotic computation from the initial constraint.

8. Conclusions

In this paper, we have shown that standard semantics for programming languages can be systematically designed as solutions of abstract domain equations involving the basic operations known for designing abstractions for program analysis. In particular, we have

shown that complementary semantics of transition systems, in this hierarchy of semantics, can be systematically constructed by domain complementation in abstract interpretation. This provides both a better insight on semantics designed for characterizing complementary observable properties of programs, and the possibility to decompose semantics into most abstract factors involving possibly new semantics (e.g. the slothful semantics). In this context, we have shown a correspondence between logic and algebraic complementation in the $\mathcal{W}p$ semantics. This means that, in *uco*, we have an element that belongs also to a Boolean algebra, it would be interesting to identify in *uco* a maximal Boolean sub-algebra of known semantics. Then we have shown a strong connection between the structure of relational abstract domains for program analysis, and compositionality of the underlying semantics. Both can be systematically designed by solving the same abstract domain equation by means of the same domain refinement: the reduced power operation. This provides an equational presentation of semantics and abstract domains for program analysis in a unique formal setting. All these results prove that standard concrete semantics and abstract domains for program analysis share a common pattern, which is designed in terms of the same basic operators for domain transformation and depends upon the property of the semantics or analysis we want to achieve. The construction of either a semantics, or a program analysis tool, can therefore be unified in a common algebraic structure, where both can be seen as solutions of simple and basic domain equations (see Fig. 9), which can be made parametric on the observable property: complete final or initial states for concrete semantics or approximated final/initial states for abstract semantics or program analysis.

References

[1] S. Abramsky, A. Jung, Domain theory, in: S. Abramsky, D.M. Gabbay, T.S.E. Maibaum (Eds.), Handbook of Logic in Computer Science, Vol. 3, Clarendon Press, Oxford, U.K., 1994, pp. 1–168.
[2] B. Alpern, F.B. Schneider, Recognizing safety and liveness, Distributed Comp. 2 (1987) 117–126.
[3] K.R. Apt, G.D. Plotkin, Countable nondeterminism and random assignment, J. ACM 33 (4) (1986) 724–767.
[4] G. Birkhoff, Lattice Theory, 3rd Ed., AMS Colloquium Publication, AMS, Providence, RI, 1967.
[5] A. Bossi, M. Gabbrielli, G. Levi, M.C. Meo, A compositional semantics for logic programs, Theoret. Comput. Sci. 122 (1–2) (1994) 3–47.
[6] M. Comini, G. Levi, An algebraic theory of observables, in: M. Bruynooghe (Ed.), Proc. 1994 Internat. Logic Programming Symp. (ILPS '94), MIT Press, Cambridge, MA, 1994, pp. 172–186.
[7] A. Cortesi, G. Filé, R. Giacobazzi, C. Palamidessi, F. Ranzato, Complementation in abstract interpretation, ACM Trans. Program. Lang. Syst. 19 (1) (1997) 7–47.
[8] P. Cousot, Abstract interpretation, ACM Comput. Surveys 28 (2) (1996) 324–328.
[9] P. Cousot, Constructive design of a hierarchy of semantics of a transition system by abstract interpretation (invited paper), in: S. Brookes, M. Mislove (Eds.), Proc. 13th Internat. Symp. on Mathematical Foundations of Programming Semantics (MFPS '97), Electronic Notes in Theoretical Computer Science, Vol. 6, Elsevier, Amsterdam, 1997, URL: http://www.elsevier.nl/locate/entcs/volume6.html.
[10] P. Cousot, Constructive design of a hierarchy of semantics of a transition system by abstract interpretation, Theoret. Comput. Sci. 277 (1–2) (2002) 47,103.
[11] P. Cousot, R. Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, in: Proc. Conf. Record of the 4th ACM Symp. on Principles of Programming Languages (POPL '77), ACM Press, New York, 1977, pp. 238–252.
[12] P. Cousot, R. Cousot, Constructive versions of Tarski's fixed point theorems, Pacific J. Math. 82 (1) (1979) 43–57.

[13] P. Cousot, R. Cousot, Systematic design of program analysis frameworks, in: Proc. Conf. Record of the 6th ACM Symp. on Principles of Programming Languages (POPL '79), ACM Press, New York, 1979, pp. 269–282.
[14] P. Cousot, R. Cousot, Abstract interpretation and application to logic programs, J. Logic Program. 13 (2–3) (1992) 103–179.
[15] P. Cousot, R. Cousot, Inductive definitions semantics, and abstract interpretation, in: Proc. Conf. Record of the 19th ACM Symp. on Principles of Programming Languages (POPL '92), ACM Press, New York, 1992, pp. 83–94.
[16] F. de Boer, M. Gabbrielli, E. Marchiori, C. Palamidessi, Proving concurrent constraint programs correct, in: Proc. Conf. Record of the ACM Symp. on Principles of Programming Languages (POPL '94), ACM Press, New York, 1994, pp. 35–35.
[17] J. Desharnais, B. Möller, F. Tchier, Kleene under a demonic star, Proc. 9th Internat. Conf. on Algebraic Methodology and Software Technology (AMAST '00), Lecture Notes in Computer Science, Vol. 1816, Springer, Berlin, 2000, pp. 355–370.
[18] E.W. Dijkstra, Guarded commands, nondeterminism and formal derivation of programs, Comm. ACM 18 (8) (1975) 453–457.
[19] E.W. Dijkstra, A discipline of programming, Series in Automatic Computation, Prentice-Hall, Englewood Cliffs, NJ, 1976.
[20] P. Dwinger, On the closure operators of a complete lattice, Indag. Math. 16 (1954) 560–563.
[21] G. Filé, R. Giacobazzi, F. Ranzato, A unifying view of abstract domain design, ACM Comput. Survey 28 (2) (1996) 333–336.
[22] G. Filé, F. Ranzato, Complementation of abstract domains made easy, in: M. Maher (Ed.), Proc. 1996 Joint Internat. Conf. and Symp. on Logic Programming (JICSLP '96), MIT Press, Cambridge, MA, 1996, pp. 348–362.
[23] H. Gaifman, E. Shapiro, Fully abstract compositional semantics for logic programs, in: Proc. Conf. Record of the 16th ACM Symp. on Principles of Programming Languages (POPL '89), ACM Press, New York, 1989, pp. 134–142.
[24] R. Giacobazzi, "Optimal" collecting semantics for analysis in a hierarchy of logic program semantics, in: C. Puech, R. Reischuk (Eds.), Proc. 13th Internat. Symp. on Theoretical Aspects of Computer Science (STACS '96), Lecture Notes in Computer Science, Vol. 1046, Springer, Berlin, 1996, pp. 503–514.
[25] R. Giacobazzi, I. Mastroeni, A characterization of symmetric semantics by domain complementation, in: Proc. 2nd Internat. Conf. in Principles and Practice of Declarative Programming PPDP'00, ACM Press, New York, 2000, pp. 115–126.
[26] R. Giacobazzi, I. Mastroeni, Non-standard semantics for program slicing, (Special issue on partial evaluation and semantics-based program manipulation), Higher-Order Symbol. Comput. 16 (4) (2003) 297–339.
[27] R. Giacobazzi, C. Palamidessi, F. Ranzato, Weak relative pseudo-complements of closure operators, Algebra Universalis 36 (3) (1996) 405–412.
[28] R. Giacobazzi, F. Ranzato, Complementing logic program semantics, in: M. Hanus, M. Rodríguez Artalejo (Eds.), Proc. 5th Internat. Conf. on Algebraic and Logic Programming (ALP '96), Lecture Notes in Computer Science, Springer, Berlin, 1996, pp. 238–253.
[29] R. Giacobazzi, F. Ranzato, Refining and compressing abstract domains, in: P. Degano, R. Gorrieri, A. Marchetti-Spaccamela (Eds.), Proc. 24th Internat. Colloq. on Automata, Languages and Programming (ICALP '97), Lecture Notes in Computer Science, Vol. 1256, Springer, Berlin, 1997, pp. 771–781.
[30] R. Giacobazzi, F. Ranzato, The reduced relative power operation on abstract domains, Theoret. Comput. Sci. 216 (1999) 159–211.
[31] R. Giacobazzi, F. Ranzato, F. Scozzari, Building complete abstract interpretations in a linear logic-based setting, in: G. Levi (Ed.), Proc. 5th Internat. Static Analysis Symp. (SAS'98), Vol. 1503, 1998, pp. 215–229.
[32] R. Giacobazzi, F. Scozzari, A logical model for relational abstract domains, ACM Trans. Program. Lang. Syst. 20 (5) (1998) 1067–1109.
[33] L. Henkin, J.D. Monk, A. Tarski, Cylindric Algebras, Part I, North-Holland, Amsterdam, 1971.
[34] P. Van Hentenryck, V. Saraswat, Y. Deville, Constraint processing in cc(FD), in: A. Podelski (Ed.), Constraint Programming: Basics and Trends, Lecture Notes in Computer Science, Vol. 910, Springer, Berlin, 1995.
[35] C.A.R. Hoare, An axiomatic basis for computer programming, Comm. ACM 12 (10) (1969) 576–580.
[36] J. Morgado, Note on complemented closure operators of complete lattices, Portugal. Math. 21 (3) (1962) 135–142.

[37] F. Nielson, Tensor products generalize the relational data flow analysis method, in: M. Arató, I. Kátai, L. Varga (Eds.), Proc. 4th Hungarian Computer Science Conf., 1985, pp. 211–225.
[38] G. Plotkin, A Structural Approach to Operational Semantics, DAIMI-19 Aarhus University, Denmark, 1981.
[39] K.I. Rosenthal, Quantales and their applications, in: Pitman Research Notes in Mathematics, Longman Scientific & Technical, London, 1990.
[40] V. Saraswat, Concurrent Constraint Programming Languages, MIT Press, Cambridge, MA, 1993.
[41] V. Saraswat, V.A. Rinard, P. Panangaden, Semantic foundations of concurrent constraint programming, in: Proc. Conf. Record of the 18th ACM Symp. on Principles of Programming Languages (POPL '91), ACM Press, New York, 1991, pp. 333–353.
[42] F. Scozzari, Logical optimality of groundness analysis, in: P. Van Hentenryck (Ed.), Proc. 4th Internat. Static Analysis Symp. (SAS'97), Lecture Notes in Computer Science, Vol. 1302, Springer, Berlin, 1997, pp. 83–97.
[43] Z. Shmuely, The structure of Galois connections, Pacific J. Math. 54 (2) (1974) 209–225.