Hyperhierarchy of Semantics - A Formal Framework for Hyperproperties Verification

Isabella Mastroeni and Michele Pasqua(✉)

Dipartimento di Informatica, University of Verona,
Strada le Grazie 15, 37134 Verona, Italy
{isabella.mastroeni,michele.pasqua}@univr.it

Abstract. Hyperproperties are becoming the, de facto, standard for reasoning about systems executions. They differ from classical trace properties since they are represented by *sets of sets* of executions instead of sets of executions. In this paper, we extend and lift the hierarchy of semantics developed in 2002 by P. Cousot in order to cope with verification of hyperproperties. In the standard hierarchy, semantics at different levels of abstraction are related with each other by abstract interpretation. In the same spirit, we propose an *hyper*hierarchy of semantics adding a new, more concrete, hyper level. The semantics defined at this hyper level are suitable for hyperproperties verification. Furthermore, all the semantics in the hyperhierarchy (the standard and the hyper ones) are still related by abstract interpretation.

1 Introduction

Since its origin in 1977, abstract interpretation [8] 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 analysis frameworks to more recent applications in model checking, program verification, comparative semantics, data and SW security, malware detection, code obfuscation, etc. When reasoning about systems executions a key point is the degree of approximation given by the choice of the semantics used to represent computations. In this direction, comparative semantics consists in comparing semantics at different levels of abstraction, always by abstract interpretation [7,18]. The choice of the semantics is a key point, not only for finding the desirable trade-off between precision and decidability of program analysis in terms, for instance, of property verification, but also because not all the semantics are suitable for proving any possible property of interest. This means that the property to verify necessarily affect the semantics we have to choose for modeling the system to analyze. For instance, if we are interested in a property which is not a safety property [2], then we have necessarily to consider a semantics able to approximate the whole computation (not only the past of a computation), as static analysis does. While, when we are interested in safety property then we have to consider a *safety abstraction* of the semantics [13,19]. Analogously, if we have to characterize slices (extraction of executable

F. Ranzato (Ed.): SAS 2017, LNCS 10422, pp. 232–252, 2017.
DOI: 10.1007/978-3-319-66706-5_12

code sub-fragments of a program [21]) of potentially non-terminating programs then we need a semantics able to characterize also what happens after loops [17].

These were only examples, but in general new (classes of) properties of interest may induce the necessity of defining new semantics, i.e., new semantic models for computational systems. In particular, we observed that *hyperproperties*, namely sets of properties, recently gained more and more interest due to their capability to capture program features that cannot be caught by classical properties, namely features that cannot be characterized by a predicate defined on single computations. For instance, information flow properties can be verified only by comparing *sets* of computations, hence they are hyperproperties, and not properties in the standard sense. Hence, what we propose here is a general formal framework for comparing semantics including the so-called *hypersemantics*, modeling programs as sets of sets of computations, since we need such a more concrete observation of systems computations in order to verify, potentially by using approximation, hyperproperties. The framework we propose is indeed an extension of the Cousot hierarchy of semantics [7] enriched with an hyper level, where still all the semantics are compared by abstract interpretation. Moreover, we show that at least two existing program analysis approaches (one recent approach for information flow analysis [3] and standard program static analysis [9]) can be included or compared in our framework.

2 Transition Systems, Semantics and Approximations

In this section, we introduce the hierarchy of semantics (both definition and construction of semantics) proposed by Cousot [7], from which we move towards the hyperlevel. In this way, while providing a formal framework for hypersemantics we can formally prove its relation with the standard semantics framework.

2.1 Trace Semantics of Systems

We reason about semantics of systems independently from systems themselves. Let $\mathcal{S}$ be the set of possible denotations of states of (computational) systems. The *concrete* semantics of a system P is given by the transition system $\langle \varSigma, \varUpsilon, \varOmega, \tau \rangle$, where $\varSigma \subseteq \mathcal{S}$ is the set of possible states of P, $\varUpsilon \subseteq \varSigma$ is the set of *all* initial states of P, $\tau \subseteq \varSigma \times \varSigma$ is the transition relation between states of P, and $\varOmega \subseteq \varSigma$ is the set of blocking/final states of P, i.e., those states σ such that $\forall \sigma' \in \varSigma \,.\, \langle \sigma, \sigma' \rangle \notin \tau$. For instance, a system could be any program written in a programming language, the state denotations could be any possible mappings from program variables to values and the transition system is given by the operational semantics of the language.

The executions of a system are modeled by sequences of transitions [7]. The set $\mathcal{S}^{\vec{n}} \overset{\text{def}}{=} [0, n) \mapsto \mathcal{S}$, $n \in \mathbb{N}$, is the set of finite sequences $s = s_0 s_1 \dots s_{n-1}$ of length $|s| = n$ over $\mathcal{S}$. The set of finite non-empty sequences is $\mathcal{S}^{\vec{+}} \overset{\text{def}}{=} \bigcup_{0<n<\omega} \mathcal{S}^{\vec{n}}$. The set $\mathcal{S}^{\vec{\omega}} \overset{\text{def}}{=} \mathbb{N} \mapsto \mathcal{S}$ contains infinite sequences $s = s_0 s_1 \dots$ of length $|s| = \omega$ over $\mathcal{S}$. The set of non-empty sequences is $\mathcal{S}^{\vec{\infty}} \overset{\text{def}}{=} \mathcal{S}^{\vec{+}} \cup \mathcal{S}^{\vec{\omega}}$. The empty sequence is ϵ. Given $s, s' \in \mathcal{S}^{\vec{\infty}}$, s' can be appended to s iff $s_{|s|-1} = s'_0$ and their append is $s \frown s' \overset{\mathsf{def}}{=} s_0 s_1 \ldots s_{|s|-1} s'_1 s'_2 \ldots s'_{|s'|-1}$ [7]. Given a system P, $\Sigma^{\vec{\infty}} \subseteq \mathcal{S}^{\vec{\infty}}$ is the set of all sequences on the states Σ of P, analogous for $\Sigma^{\vec{+}} \subseteq \mathcal{S}^{\vec{+}}$ and $\Sigma^{\vec{\omega}} \subseteq \mathcal{S}^{\vec{\omega}}$.

An execution (*trace*) of a system P is a sequence of states in Σ where adjacent elements are in τ. $\tau^{\dot{\vec{n}}} \overset{\mathsf{def}}{=} \{\sigma \in \Sigma^{\vec{n}} \mid \forall i \in [0, n-1) \,.\, \langle \sigma_i, \sigma_{i+1} \rangle \in \tau\}$ are the finite traces of length n, while the set of finite blocking traces of length n is $\tau^{\vec{n}} \overset{\mathsf{def}}{=} \{\sigma \in \Sigma^{\vec{n}} \mid \sigma_{n-1} \in \Omega \wedge \forall i \in [0, n-1) \,.\, \langle \sigma_i, \sigma_{i+1} \rangle \in \tau\}$.

The *maximal finite trace semantics* (set of blocking/terminating executions) is $\tau^{\vec{+}} \overset{\mathsf{def}}{=} \bigcup_{0<n<\omega} \tau^{\vec{n}}$. The *infinite trace semantics* (set of non-blocking/non-terminating executions) is $\tau^{\vec{\omega}} \overset{\mathsf{def}}{=} \{\sigma \in \Sigma^{\vec{\omega}} \mid \forall i \in \mathbb{N} \,.\, \langle \sigma_i, \sigma_{i+1} \rangle \in \tau\}$. The *maximal trace semantics* is $\tau^{\vec{\infty}} \overset{\mathsf{def}}{=} \tau^{\vec{+}} \cup \tau^{\vec{\omega}}$ [7]. In the following, in order to avoid ambiguity, we can make explicit the system, e.g., we can write $\tau^{\vec{\infty}}[P]$ instead of just $\tau^{\vec{\infty}}$ in order to denote the maximal trace semantics of P.

2.2 Fixpoint Semantics Approximation

A semantics $\mathcal{T}$ is said to be *constructive*, i.e., expressible in fixpoint form, if there exists a *fixpoint semantic specification* $\langle F, D, \preccurlyeq \rangle$, where $\langle D, \preccurlyeq, \vee, \bot \rangle$ is a partially ordered set with (partially defined) least upper bound $\vee$ and minimum $\bot$ (usually at least a DCPO[1]), $F : D \to D$ is $\preccurlyeq$-monotone and iteratable[2] and $\mathcal{T} = \mathit{lfp}_{\bot}^{\preccurlyeq} F = F^{\delta}$, where δ is the least ordinal such that $F^{\delta} = F(F^{\delta})$ and F^{δ} is equal to $\bigvee_{n \leq \delta} F^n(\bot)$ [14].

Consider now the semantic specifications $\langle F, D, \preccurlyeq \rangle, \langle \bar{F}, \bar{D}, \bar{\preccurlyeq} \rangle$, and suppose that $\langle D, \preccurlyeq \rangle, \langle \bar{D}, \bar{\preccurlyeq} \rangle$ form a *Galois connection*[3], by means of the functions $\alpha : D \xrightarrow{m} \bar{D}$ (abstraction) and $\gamma : \bar{D} \xrightarrow{m} D$ (concretization), namely α and γ are adjoint functions. When the semantics is expressed in fixpoint form, we can derive an abstract fixpoint semantics by abstraction of a concrete one, or vice versa. The Kleenian fixpoint approximation theorem [7], requires abstraction soundness, i.e., $\alpha \circ F \mathrel{\bar{\preccurlyeq}} \bar{F} \circ \alpha$, guaranteeing fixpoint approximation, i.e., $\alpha(\mathit{lfp}_{\bot}^{\preccurlyeq} F) \mathrel{\bar{\preccurlyeq}} \mathit{lfp}_{\bar{\bot}}^{\bar{\preccurlyeq}} \bar{F}$. The (in the following called *backward*) Kleenian fixpoint transfer theorem [7] requires completeness, i.e., $\alpha \circ F = \bar{F} \circ \alpha$, guaranteeing the fixpoint transfer from concrete to abstract domain, i.e., $\alpha(\mathit{lfp}_{\bot}^{\preccurlyeq} F) = \mathit{lfp}_{\bar{\bot}}^{\bar{\preccurlyeq}} \bar{F}$.

Suppose now we are interested in transferring the fixpoint from an abstract domain to the concrete one[4]. Unfortunately, the completeness requirement observed in the abstract domain (called *backward*), i.e., $\alpha \circ F = \bar{F} \circ \alpha$, is not

[1] A *DCPO* is a poset where it exists the least upper bound of every directed subset.

[2] A function F over D is said *iteratable* if the transfinite iterates of F from $\bot$ are well defined. The *transfinite iterates* of F from $\bot$ are $F^0 = \bot$ and $F^{\delta+1} = F(F^{\delta})$ for successor ordinals $\delta + 1$ and $F^{\zeta} = \bigvee_{\delta<\zeta} F^{\delta}$ for limit ordinals ζ.

[3] α, γ form a Galois connection between concrete $\langle D, \preccurlyeq \rangle$ and abstract $\langle \bar{D}, \bar{\preccurlyeq} \rangle$ domains, denoted $\langle D, \preccurlyeq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \bar{D}, \bar{\preccurlyeq} \rangle$, if $\forall c \in D, a \in \bar{D} \,.\, \alpha(c) \mathrel{\bar{\preccurlyeq}} a \Leftrightarrow c \preccurlyeq \gamma(a)$. If $\alpha \circ \gamma = id_{\bar{D}}$ then they form a Galois insertion, denoted $\langle D, \preccurlyeq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \bar{D}, \bar{\preccurlyeq} \rangle$.

[4] This direction does not change anything in the approximation case, since the soundness requirement is equivalent also when we check it on the concrete, i.e., $\alpha \circ F \mathrel{\bar{\preccurlyeq}} \bar{F} \circ \alpha$ iff $F \circ \gamma \preccurlyeq \gamma \circ \bar{F}$.

the same as checking completeness on the concrete domain (called *forward*), i.e., $F \circ \gamma = \gamma \circ \bar{F}$. In order to transfer fixpoints from abstract to concrete we need precisely the latter direction. In this case, we provide the forward version of the Kleenian fixpoint transfer theorem.

Theorem 1 (*Forward* Kleenian fixpoint transfer). *Suppose that $\langle F, D, \preccurlyeq \rangle$ and $\langle \bar{F}, \bar{D}, \bar{\preccurlyeq} \rangle$ are concrete and abstract fixpoint semantics specifications. Let $\gamma : \bar{D} \to D$ be a strict Scott-continuous*[5] *concretization function. If $\gamma \circ \bar{F} = F \circ \gamma$ (forward completeness) then $\gamma(\mathit{lfp}^{\bar{\preccurlyeq}}_{\bar{\bot}} \bar{F}) = \mathit{lfp}^{\preccurlyeq}_{\bot} F$.*

In the abstract interpretation framework, it is well known that the Kleenian fixpoint approximation trivially hold when $\bar{F}$ is the best correct approximation (bca) of F, i.e., $\bar{F} = \alpha \circ F \circ \gamma$. Hence, we look for a similar characterization in the dual case. In particular, we look for a systematic way to retrieve a concrete semantics which best represents a given abstract function. Exploiting the "duality principle" of abstract interpretation [10] we can obtain the best correct concretization as $F \overset{\mathsf{def}}{=} \gamma \circ \bar{F} \circ \alpha$. Then we still trivially have that $\gamma(\mathit{lfp}^{\bar{\succcurlyeq}}_{\bar{\bot}} \bar{F}) \succcurlyeq \mathit{lfp}^{\succcurlyeq}_{\bot} F$ and $\mathit{lfp}^{\bar{\succcurlyeq}}_{\bar{\bot}} \bar{F} \mathrel{\bar{\succcurlyeq}} \alpha(\mathit{lfp}^{\succcurlyeq}_{\bot} F)$. Moreover, in a Galois insertion settings, it is always possible to derive a complete (backward and forward) concretisation, called *best complete concretisation*, of a given abstract semantics:

Theorem 2 (Best Complete Concretization). *Let $\langle D, \preccurlyeq \rangle$ and $\langle \bar{D}, \bar{\preccurlyeq} \rangle$ be partially ordered sets such that $\langle D, \preccurlyeq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \bar{D}, \bar{\preccurlyeq} \rangle$. Let $\bar{F} : \bar{D} \xrightarrow{m} \bar{D}$ and $F^{\mathrm{bcc}} = \gamma \circ \bar{F} \circ \alpha$. Then $\bar{D}$ is both backward and forward complete for F^{bcc}.*

Note that $\bar{F}$ is exactly the bca of F^{bcc} in $\bar{D}$, indeed ${F^{\mathrm{bcc}}}^{\mathrm{bca}} = \alpha \circ F^{\mathrm{bcc}} \circ \gamma = \alpha \circ \gamma \circ \bar{F} \circ \alpha \circ \gamma = \bar{F}$. Hence, given an abstract function $\bar{F}$ it is possible to derive a concrete function F, for which $\bar{F}$ is an approximation, such that $\alpha(\mathit{lfp}^{\preccurlyeq}_{\bot} F) = \mathit{lfp}^{\bar{\preccurlyeq}}_{\bar{\bot}} F^{\sharp}$ and $\mathit{lfp}^{\preccurlyeq}_{\bot} F = \gamma(\mathit{lfp}^{\bar{\preccurlyeq}}_{\bar{\bot}} \bar{F})$.

2.3 Standard Hierarchy of Semantics

In [7] the author showed that many well-known semantics can be computed as abstract interpretations of the maximal trace semantics, and they can be organized in a hierarchy. For instance, the *relational semantics* τ^{∞} associates an input/output relation with system traces by using the $\bot$ symbol to denote non-termination, while *denotational semantics* $\tau^{\natural}$ gives semantics by considering input/output functions. Each semantics (said to be in *natural style*) have three different abstractions, for instance the *angelic* abstraction, which observes only finite computations, e.g., the *angelic trace semantics* $\tau^{\vec{+}}$ observes only finite traces, while the *angelic relational semantics* τ^{+} and the *angelic denotational semantics* $\tau^{\flat}$ the corresponding relations and functions. In [7] the author consider also several other semantics but, in sake of simplicity, we focus only in the subset of the hierarchy depicted in Fig. 1, on the left. Another useful semantics is *partial trace semantics* (finite prefixes of computations, starting from initial states): $\tau^{\vec{\propto}} = \bigcup_{0<n<\omega} \{\sigma \in \tau^{\dot{\vec{n}}} \mid \sigma_0 \in \mathit{\Upsilon}\}$ [12].

[5] A function f is said *Scott-continuous* if preserves the least upper bound of directed subsets of X and it is said *strict* if $f(\bot) = \bot$.

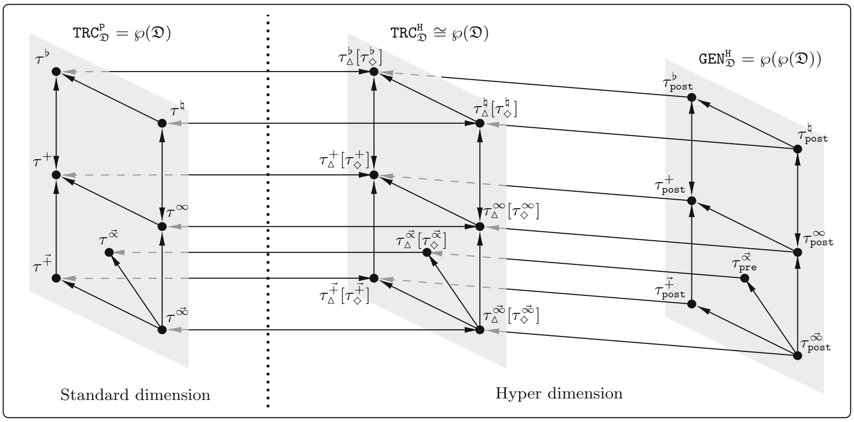
$\mathtt{TRC}^{\mathtt{P}}_{\mathfrak{D}} = \wp(\mathfrak{D})$
$\tau^{\flat}$
$\tau^{\natural}$
τ^{+}
τ^{∞}
$\tau^{\vec{\propto}}$
$\tau^{\vec{+}}$
$\tau^{\vec{\infty}}$
Standard dimension
$\mathtt{TRC}^{\mathtt{H}}_{\mathfrak{D}} \cong \wp(\mathfrak{D})$
$\tau^{\flat}_{\vartriangle}[\tau^{\flat}_{\diamond}]$
$\tau^{\natural}_{\vartriangle}[\tau^{\natural}_{\diamond}]$
$\tau^{+}_{\vartriangle}[\tau^{+}_{\diamond}]$
$\tau^{\infty}_{\vartriangle}[\tau^{\infty}_{\diamond}]$
$\tau^{\vec{\propto}}_{\vartriangle}[\tau^{\vec{\propto}}_{\diamond}]$
$\tau^{\vec{+}}_{\vartriangle}[\tau^{\vec{+}}_{\diamond}]$
$\tau^{\vec{\infty}}_{\vartriangle}[\tau^{\vec{\infty}}_{\diamond}]$
$\mathtt{GEN}^{\mathtt{H}}_{\mathfrak{D}} = \wp(\wp(\mathfrak{D}))$
$\tau^{\flat}_{\mathtt{post}}$
$\tau^{\natural}_{\mathtt{post}}$
$\tau^{+}_{\mathtt{post}}$
$\tau^{\infty}_{\mathtt{post}}$
$\tau^{\vec{\propto}}_{\mathtt{pre}}$
$\tau^{\vec{+}}_{\mathtt{post}}$
$\tau^{\vec{\infty}}_{\mathtt{post}}$
Hyper dimension

Fig. 1. A part of the standard hierarchy of semantics with its hyper counterparts

Furthermore, these semantics can all be computed by fixpoint of a monotone operator over an ordered domain [7,12]. In this case, it is not always possible to obtain semantics by fixpoint w.r.t. the standard inclusion order ($\subseteq$), also called the approximation order. In fact, in some cases the fixpoint operator is not monotone on the approximation order, and therefore we have to define a computational order forcing monotonicity, and therefore convergence of the fixpoint operator. For instance, the maximal trace semantics of P can be computed as: $\tau^{\vec{\infty}} \underset{\cdot}{=} \mathit{lfp}^{\sqsubseteq^{\vec{\infty}}}_{\perp^{\vec{\infty}}} F^{\vec{\infty}}$, where $F^{\vec{\infty}} : \wp(\Sigma^{\vec{\infty}}) \rightarrow \wp(\Sigma^{\vec{\infty}})$ is defined as $F^{\vec{\infty}} \overset{\mathsf{def}}{=} \lambda X \,.\, \tau^{\vec{1}} \cup (\tau^{\vec{2}} \frown X)$, which is monotone on the computational order $X \sqsubseteq^{\vec{\infty}} Y \overset{\mathsf{def}}{=} ((X \cap \Sigma^{\vec{+}}) \subseteq (Y \cap \Sigma^{\vec{+}})) \wedge (X \cap \Sigma^{\vec{\omega}}) \supseteq (Y \cap \Sigma^{\vec{\omega}}))$ (the corresponding lub is $\bigsqcup^{\vec{\infty}} X_i \overset{\mathsf{def}}{=} \bigcup (X_i \cap \Sigma^{\vec{+}}) \cup \bigcap (X_i \cap \Sigma^{\vec{\omega}})$ and $\perp^{\vec{\infty}} \overset{\mathsf{def}}{=} \Sigma^{\vec{\omega}}$). As far as the partial semantics is concerned, the semantics operator is computed as: $\tau^{\vec{\propto}} \underset{\cdot}{=} \mathit{lfp}^{\subseteq}_{\varnothing} F^{\vec{\propto}}$, where $F^{\vec{\propto}} : \wp(\Sigma^{\vec{+}}) \rightarrow \wp(\Sigma^{\vec{+}})$ is defined as $F^{\vec{\propto}} \overset{\mathsf{def}}{=} \lambda X \,.\, \Upsilon \cup (X \frown \tau^{\vec{2}})$, which is monotone on the standard approximation order ($\subseteq$) [12].

Example 1. Let $P \overset{\mathsf{def}}{=} l := 4;\ \textbf{if}\,(h = 1)\,\textbf{then}\, l := 2h\, \textbf{else}\ \textbf{while}\,(\mathsf{true})\,\textbf{do}\,\{l := 6\}$, and let us denote states as maps between variables to values ($[n, m]$ means $l \mapsto n$, $h \mapsto m$). Maximal trace semantics $\tau^{\vec{\infty}}[\![P]\!]$ and relational semantics $\tau^{\infty}[\![P]\!]$ are:

$$\tau^{\vec{\infty}}[\![P]\!] = \{\, [n,1][4,1][2,1], [4,1][2,1], [2,1], [n,m][4,m][6,m]^{\omega} \;\mid n \in \mathbb{N}, m \in \mathbb{N} \setminus \{1\}\}$$
$$\tau^{\infty}[\![P]\!] = \{\; \langle [n,1],[2,1]\rangle,\; \langle [4,1][2,1]\rangle,\; \langle [2,1][2,1]\rangle,\; \langle [n,m],\perp\rangle \;\mid n \in \mathbb{N}, m \in \mathbb{N} \setminus \{1\}\}$$

3 *Hyper*properties

In the security context, there are policies that can be expressed as trace properties, like access control, and others which cannot, like non-interference. In this latter case, it is necessary to specify it as an hyperproperty. Intuitively, a property is defined exclusively in terms of individual executions and, in general, do not specify a relation between different executions of the system. Instead, an hyperproperty specifies the set of *sets of system executions* allowed by the security policy, therefore expressing relations between executions. In [5] it is stated that in order to formalize security policies, it is sufficient to consider hyperproperties. This means that hyperproperties are able to define every possible security policy (this is true for systems modeled as set of states traces).

In this section, we introduce the notion of *hyperproperty* [5], i.e., a set of sets of executions. In the original formulation, systems are modeled by non-empty sets of infinite traces, where terminating executions are modeled by repeating the final state of the trace an infinite number of times [5]. In our work, we will reason about hyperproperties keeping generality, so we are not restricted to only infinite sequences.

Safety Hyperproperties [5]. In the context of trace properties, a particular kind of properties are *safety* ones [2], expressing the fact that "nothing bad happens". These properties are interesting because they depend only on the *history/past* of single executions, meaning that safety properties are dynamically monitorable [2]. Similarly, *safety hyperproperties* (or hypersafety) are the lift to sets of safety properties. This means that, for each set of executions that is not in a safety hyperproperty, there exists a finite prefix set of finite executions (the "bad thing") which cannot be extended for satisfying the property.

Another particular class of hyperproperties are the k-safety hyperproperties (or k-hypersafety). They are safety hyperproperties in which the "bad thing" never involves more than k executions [5]. This means that it is possible to check the violation of a k-hypersafety just observing a set of k executions (note that 1-hypersafeties are exactly safety properties). This is important for verification, in fact, it is possible to reduce the verification of a k-hypersafety on system P to the verification of a safety on the self-composed systems P^k [5]. Furthermore, lots of interesting security policies can be formalized as k-hypersafety; for instance, some definitions of non-interference are 2-hypersafety.

The topic of hyperproperties verification is quite new. Besides the reduction to safety, in [1] the authors introduce a runtime refutation methods for k-safety, based on a three-valued logic. Similarly, [4,15] define hyperlogics, i.e., extensions of temporal logic able to quantify over multiple traces. The use of abstract interpretation in hyperproperties verification is limited to [3], analyzed in Sect. 7.

4 Verifying Hyperproperties

In this section, we deal with hyperproperties *verification*. Here, by verification we mean both *validation*, i.e., checking whether a system fulfills the property, and *confutation*, i.e., checking whether a system does not fulfill the property. It is well known that we cannot always answer to both these problems precisely.

Consider the set of state denotations $\mathcal{S}$ and a set $\mathfrak{D}$ of all possible executions of any system P on states $\mathcal{S}$. The execution of a system could be a sequence (finite or infinite), a pair, etc., of elements in $\mathcal{S}$, depending on how we mean to represent computations. In the following, given a system P, we denote by $[\![P]\!] \subseteq \mathfrak{D}$ a generic semantics of P, parametric on the executions domain $\mathfrak{D}$. For instance, if

$\mathfrak{D} = \mathcal{S}^{\vec{\infty}}$ then we consider the maximal trace semantics of P, i.e., $[\![P]\!] = \tau^{\vec{\infty}}$, while if $\mathfrak{D} = \mathcal{S} \times \mathcal{S}$ then we consider the angelic relational semantics of P, i.e., $[\![P]\!] = \tau^{+}$. Usually, a trace property is modeled as the set of all executions satisfying it. Hence, let $\mathfrak{P} \subseteq \mathfrak{D}$ be such a property, then it is well known that a system P *satisfies* $\mathfrak{P}$, denoted as $P \models \mathfrak{P}$, iff $[\![P]\!] \subseteq \mathfrak{P}$. Hence, by definition, $\mathfrak{P}$ is fulfilled for a system P iff $\mathfrak{P}$ is fulfilled for each one of its executions, i.e., $P \models \mathfrak{P}$ iff $\forall s \in [\![P]\!] \, . \, s \in \mathfrak{P}$ (validation). This is quite useful because in order to disprove that a system fulfills a trace property we just need one counterexample, i.e., $P \not\models \mathfrak{P}$ iff $\exists s \in [\![P]\!] \, . \, s \notin \mathfrak{P}$ (confutation). We denote by $\mathtt{TRC}^{\mathtt{P}}_{\mathfrak{D}}$ the set of all trace properties, i.e., $\wp(\mathfrak{D})$. For instance, trace properties in $\wp(\mathfrak{D})$, for $\mathfrak{D} = \mathcal{S}^{\vec{\infty}}$, are *termination* $\mathtt{Term} \overset{\text{def}}{=} \mathcal{S}^{\vec{+}}$ and $\mathtt{Even}^{l} \overset{\text{def}}{=} \{s \in \mathcal{S}^{\vec{\infty}} \mid \forall i > 0 \, . \, s_i(l) \text{ even}\}$ (saying that variable l is always even after initialization). Note that, the program in Example 1 satisfies $\mathtt{Even}^{l}$ but not $\mathtt{Term}$, since $\tau^{\vec{\infty}}[P] \subseteq \mathtt{Even}^{l}$, while $\tau^{\vec{\infty}}[P] \nsubseteq \mathtt{Term}$.

For hyperproperties, the satisfiability relation changes from set-inclusion to set-membership [5], namely $P \models \mathfrak{Hp}$ iff $[\![P]\!] \in \mathfrak{Hp}$.

4.1 Hyperproperties Verification

As introduced in Sect. 2, hyperproperties are sets of sets of executions, hence the domain of hyperproperties is $\wp(\wp(\mathfrak{D}))$. We denote by $\mathtt{GEN}^{\mathtt{H}}_{\mathfrak{D}}$ the set of all (generic) hyperproperties, i.e., $\wp(\wp(\mathfrak{D}))$. Similarly to what happens for trace properties, we characterize hyperproperty validation as:

$$P \models \mathfrak{Hp} \in \mathtt{GEN}^{\mathtt{H}}_{\mathfrak{D}} \;\Leftrightarrow\; [\![P]\!] \in \mathfrak{Hp} \;\Leftrightarrow\; \{[\![P]\!]\} \subseteq \mathfrak{Hp}$$

This means that the strongest hyperproperty of a system P is $[\![P]\!]_{\diamond} \overset{\text{def}}{=} \{[\![P]\!]\}$ [6], since every hyperproperty of P is implied by, i.e., include, $[\![P]\!]_{\diamond}$. An example of a generic hyperproperty for $\mathfrak{D} = \mathcal{S}^{\vec{\infty}}$ is *generalized non-interference* $\mathtt{GNI} \overset{\text{def}}{=} \{X \subseteq \wp(\mathfrak{D}) \mid \forall s, s' \in X \, \exists \bar{s} \in X \, . \, (\bar{s}_{\vdash} =_{\mathtt{H}} s_{\vdash} \wedge \bar{s} \approx_{\mathtt{L}} s')\}$ [5], stating that, for each pair s, s' of executions there exists an interleaving one $\bar{s}$ which agrees with s on private variables ($\mathtt{H}$) in input ($\vdash$) and with s' on public variables ($\mathtt{L}$)[6]. The program in Example 1 do not satisfy $\mathtt{GNI}$, since $\tau^{\vec{\infty}}[P] \notin \mathtt{GNI}$.

At this point, we wonder whether we can use standard semantics for verifying, at least, a subset of hyperproperties. Let us consider the following restriction.

Definition 1 (Trace hyperproperty). $\mathfrak{tHp} \in \mathtt{GEN}^{\mathtt{H}}_{\mathfrak{D}}$ *is called* trace *hyperproperty if* $\mathfrak{tHp} = \wp(\bigcup \mathfrak{tHp})$*, i.e., if* $\langle \mathfrak{tHp}, \subseteq, \cup, \cap, \varnothing, \bigcup \mathfrak{tHp} \rangle$ *is a boolean algebra*[7]*.*

We denote with $\mathtt{TRC}^{\mathtt{H}}_{\mathfrak{D}}$ the set of all trace hyperproperties, i.e., $\mathtt{TRC}^{\mathtt{H}}_{\mathfrak{D}}$ is the set $\{\mathfrak{tHp} \in \mathtt{GEN}^{\mathtt{H}}_{\mathfrak{D}} \mid \wp(\bigcup \mathfrak{tHp}) = \mathfrak{tHp}\}$. Hence, we have validation as

$$P \models \mathfrak{tHp} \in \mathtt{TRC}^{\mathtt{H}}_{\mathfrak{D}} \;\Leftrightarrow\; \{\{s\} \mid s \in [\![P]\!]\} \subseteq \mathfrak{tHp} \;\Leftrightarrow\; \forall s \in [\![P]\!] \, . \, \{s\} \in \mathfrak{tHp}$$

[6] Note that $=_{\mathtt{H}}$ is an equivalence on states while $\approx_{\mathtt{L}}$ is on traces.

[7] A *boolean algebra* is a complemented (each $x \in X$ has complement $y \in X$: $x \wedge y = \bot$, $x \vee y = \top$) and distributive ($\forall x, y, z \in X \, . \, x \wedge (y \vee y) = (x \vee y) \wedge (x \vee z)$) lattice.

This means that, exactly as it happens for properties, we can check this kind of hyperproperties on single executions: if we find at least one execution not satisfying the hyperproperty, then the whole system does not satisfy it. For example, $\mathtt{Even}^l_{\mathcal{H}} \overset{\text{def}}{=} \wp(\mathtt{Even}^l)$ is the trace hyperproperty equivalent to trace property $\mathtt{Even}^l$.

The hyperproperties which we can verify with standard trace semantics are all and only the trace hyperproperties, as stated by the following theorem.

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

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

Direction ($\Rightarrow$) holds since, by definition, $\mathfrak{Hp} \in \mathtt{TRC}^{\mathtt{H}}_{\mathfrak{D}}$ implies $\mathfrak{Hp} = \wp(\bigcup \mathfrak{Hp})$, and setting $\mathfrak{P} = \bigcup \mathfrak{Hp}$ we have $[\![P]\!] \subseteq \bigcup \mathfrak{Hp} \Leftrightarrow \wp([\![P]\!]) \subseteq \wp(\bigcup \mathfrak{Hp}) \Leftrightarrow [\![P]\!] \in \mathfrak{Hp}$. For the converse ($\Leftarrow$) we give only an intuition. Take, for instance, $\mathfrak{Hp} = \{\{a\},\{b\}\} \notin \mathtt{TRC}^{\mathtt{H}}_{\mathfrak{D}}$, so $\forall \mathfrak{P} \in \mathtt{TRC}^{\mathtt{P}}_{\mathfrak{D}}\, \exists P \in$ systems such that $P \models \mathfrak{P} \Leftrightarrow P \models \mathfrak{Hp}$ do not hold. Indeed, if $\mathfrak{P} \cap \bigcup \mathfrak{Hp} \supseteq \{a, b\}$ consider $[\![P]\!] = \{a, b\}$, then we have $[\![P]\!] \subseteq \mathfrak{P}$ but $[\![P]\!] \notin \mathfrak{Hp}$. Otherwise, if $\mathfrak{P} \cap \bigcup \mathfrak{Hp} \supseteq \{a\}$ take $[\![P]\!] = \{b\}$, otherwise take $[\![P]\!] = \{a\}$, in any case we can show that $[\![P]\!] \in \mathfrak{Hp}$ but $[\![P]\!] \not\subseteq \mathfrak{P}$.

We can further generalize this restriction, allowing us to preserve the possibility of verifying hyperproperty on trace semantics at least for confutation. It should be clear that, in the general case, we have to compute the whole semantics $[\![P]\!]$ in order to verify (both validate and confute) the hyperproperty $\mathfrak{Hp}$. However, it is worth noting that there is a particular kind of hyperproperties that generalizes hypersafety and whose verification test can be simplified.

Definition 2 (Subset-closed hyperproperty). *$\mathfrak{cHp} \in \mathtt{GEN}^{\mathtt{H}}_{\mathfrak{D}}$ is called a* subset-closed *hyperproperty if $\mathfrak{cHp}$ is such that $X \in \mathfrak{cHp} \Rightarrow (\forall Y \subseteq X\,.\,Y \in \mathfrak{cHp})$.*

We denote with $\mathtt{SSC}^{\mathtt{H}}_{\mathfrak{D}}$ the set of all subset-closed hyperproperties, i.e., $\mathtt{SSC}^{\mathtt{H}}_{\mathfrak{D}}$ is the set $\{\mathfrak{cHp} \in \mathtt{GEN}^{\mathtt{H}}_{\mathfrak{D}} \mid X \in \mathfrak{cHp} \Rightarrow (\forall Y \subseteq X\,.\,Y \in \mathfrak{cHp})\}$. Note that all trace hyperproperties are subset-closed but not vice-versa (one example is observational determinism [22]). In particular, a subset-closed hyperproperty $\mathfrak{cHp}$ is also a trace hyperproperty if, in addition, it holds: $X, Y \in \mathfrak{cHp} \Rightarrow X \cup Y \in \mathfrak{cHp}$. It turns out that lots of interesting hyperproperties are subset-closed, e.g., all hypersafety and some hyperliveness [5]. In this case, validation becomes

$$P \models \mathfrak{cHp} \in \mathtt{SSC}^{\mathtt{H}}_{\mathfrak{D}} \;\Leftrightarrow\; \wp([\![P]\!]) \subseteq \mathfrak{cHp} \;\Leftrightarrow\; \forall X \subseteq [\![P]\!]\,.\,X \in \mathfrak{cHp}$$

where $[\![P]\!]_{\vartriangle} \overset{\text{def}}{=} \wp([\![P]\!])$ is the strongest subset-closed hyperproperty of P. It is clear that this does not change the validation of $\mathfrak{cHp}$, but it may in general simplify the confutation, since we do not need the whole semantics $[\![P]\!]$: it is sufficient to find a $X \subseteq [\![P]\!]$ such that $X \notin \mathfrak{cHp}$ in order to imply $\{[\![P]\!]\} \not\subseteq \mathfrak{cHp}$. A subset-closed hyperproperty for $\mathfrak{D} = \mathcal{S} \times \mathcal{S}_{\perp}$ which is not a trace hyperproperty is *termination insensitive non-interference* $\mathtt{TINI} \overset{\text{def}}{=} \{X \subseteq \wp(\mathfrak{D}) \mid \forall s, s' \in X\,.\,s_{\vdash} =_{\mathtt{L}} s'_{\vdash} \Rightarrow (s_{\dashv} = \perp \vee s'_{\dashv} = \perp \vee s_{\dashv} =_{\mathtt{L}} s'_{\dashv})\}$ [5], stating that, each pair of executions agreeing on public variables ($\mathtt{L}$) in input ($\vdash$), must terminate agreeing on public variables in output ($\dashv$). The program in Example 1, with typing $\Gamma(l) = \mathtt{L}, \Gamma(h) = \mathtt{H}$, satisfies

TINI since all terminating traces provides the same value for l, i.e., $\tau^{\infty}[\![P]\!] \in \mathtt{TINI}$. In [5], the authors proved that TINI is 2-hypersafety, hence it is subset-closed, and, conversely, they proved that GNI is not subset-closed.

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

Proposition 1. *Every subset-closed hyperproperty* $\mathfrak{cHp}$ *can be decomposed in a conjunction of trace hyperproperties, namely:*

$$\mathfrak{cHp} = \bigcup_{Y \in \max_{\subseteq}(\mathfrak{cHp})} \wp(Y) \quad \textit{with} \quad \max_{\subseteq}(\mathcal{X}) \overset{\text{def}}{=} \left\{ X \in \mathcal{X} \;\middle|\; \begin{array}{l} \forall X' \in \mathcal{X} \,. \\ X \subseteq X' \Rightarrow X = X' \end{array} \right\}$$

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

Clearly, for all Y in $\max_{\subseteq}(\mathfrak{cHp})$, it holds $\wp(\bigcup \wp(Y)) = \wp(Y)$ so $\wp(Y)$ is a trace hyperproperty. Hence any subset-closed hyperproperty can be characterized as $\mathfrak{cHp} = \bigcup_{i \in \Delta} \mathfrak{tHp}_i$ (for a set $\Delta \subseteq \mathbb{N}$). This implies that, in order to validate $\mathfrak{cHp}$ on standard trace semantics it is sufficient to validate just one of these $\mathfrak{tHp}_i$. In fact, if $P \models \mathfrak{tHp}_i$, i.e., $[\![P]\!] \in \mathfrak{tHp}_i$, then $[\![P]\!] \in \mathfrak{cHp}$ and hence $P \models \mathfrak{cHp}$.

4.2 Hyperproperties Relations and Algebraic Structures

In this section, we show the relations existing among the notions of hyperproperties we have introduced. Moreover, we describe the algebraic structures of hyperproperties domains. In the following, we omit the subscript of properties/hyperproperties domain when it is clear from the context or not relevant.

It is straightforward to note that $\mathtt{TRC}^{\mathtt{H}} \subsetneq \mathtt{SSC}^{\mathtt{H}} \subsetneq \mathtt{GEN}^{\mathtt{H}}$ and that $\mathtt{SSC}^{\mathtt{H}}$ (and therefore $\mathtt{TRC}^{\mathtt{H}}$) do not contain $\varnothing$. Indeed the empty set has no members, so it cannot be subset-closed. In addition, the unique singleton subset-closed is $\{\varnothing\}$.

Now let $\rho_{\star}$ be the function $\lambda\mathcal{X} \,.\, \gamma_{\star} \circ \alpha_{\star}(\mathcal{X})$, where $\alpha_{\star} \overset{\text{def}}{=} \lambda X \,.\, \bigcup X$ and $\gamma_{\star} \overset{\text{def}}{=} \lambda X \,.\, \wp(X)$, and let $\rho_{\vartriangle}$ be the function $\lambda\mathcal{X} \,.\, \{X \mid \exists Y \in \mathcal{X} \,.\, X \subseteq Y\}$. It is easy to note that they are both upper closure operators of $\mathtt{GEN}^{\mathtt{H}}$ (i.e., monotone operators in $\wp(\wp(\mathfrak{D})) \to \wp(\wp(\mathfrak{D}))$ which are extensive and idempotent)[8].

Proposition 2. $\mathtt{SSC}^{\mathtt{H}} = \rho_{\vartriangle}(\mathtt{GEN}^{\mathtt{H}})$ *and* $\mathtt{TRC}^{\mathtt{H}} = \rho_{\star}(\mathtt{GEN}^{\mathtt{H}}) = \rho_{\star}(\mathtt{SSC}^{\mathtt{H}})$.

Note that $\langle \mathtt{SSC}^{\mathtt{H}}, \subseteq, \cup, \cap, \{\varnothing\}, \wp(\mathfrak{D}) \rangle$ is a complete lattice, where the bottom is $\{\varnothing\}$ because $\varnothing$ is contained in every subset-closed set and the top is $\wp(\mathfrak{D})$ because it is the top of $\mathtt{GEN}^{\mathtt{H}}$ and it is subset-closed. For the same reasons they are the bottom and the top of the complete lattice $\langle \mathtt{TRC}^{\mathtt{H}}, \subseteq, \cup, \cap, \{\varnothing\}, \wp(\mathfrak{D}) \rangle$, which is the sublattice of $\mathtt{SSC}^{\mathtt{H}}$ (and $\mathtt{GEN}^{\mathtt{H}}$) comprising its boolean algebras. Finally, it is straightforward to note that $\mathtt{TRC}^{\mathtt{H}}$ is isomorphic, through $\langle \alpha_{\star}, \gamma_{\star} \rangle$, to $\mathtt{TRC}^{\mathtt{P}}$. The big picture is depicted by the commutative diagram in Fig. 2. Recall that the approximation order plays the role of implication. So the strongest hyperproperty, i.e., the one which implies any other hyperproperty, is $\varnothing$ for $\mathtt{GEN}^{\mathtt{H}}$ and

[8] The adjunction $\langle \alpha_{\star}, \gamma_{\star} \rangle$ and its link with systems properties were already introduced in [3] (their $\langle \alpha_{\text{hpp}}, \gamma_{\text{hpp}} \rangle$) and even before in [13] (their $\langle \alpha_{\Theta}, \gamma_{\Theta} \rangle$).

$\{\varnothing\}$ for $\mathtt{SSC}^{\mathtt{H}}, \mathtt{TRC}^{\mathtt{H}}$. Conversely, the weakest hyperproperty, i.e., the one which is implied by any other one, is $\wp(\mathfrak{D})$ for $\mathtt{GEN}^{\mathtt{H}}, \mathtt{SSC}^{\mathtt{H}}, \mathtt{TRC}^{\mathtt{H}}$. For what concerns $\mathtt{TRC}^{\mathtt{P}}$, it is isomorphic to $\mathtt{TRC}^{\mathtt{H}}$ hence the strongest trace property is $\alpha_\star(\{\varnothing\}) = \varnothing$ and the weakest is $\alpha_\star(\wp(\mathfrak{D})) = \mathfrak{D}$, as expected.

$$\begin{tikzcd}
 & \langle \mathtt{TRC}^{\mathtt{P}}, \subseteq, \cup, \cap, \varnothing, \mathfrak{D} \rangle \arrow[dl, shift right, "\gamma^{*}"'] \arrow[d, shift right, "\gamma_{\star}"'] \arrow[dr, shift left, two heads, "\gamma_{\star}"'] & \\
\langle \mathtt{GEN}^{\mathtt{H}}, \subseteq, \cup, \cap, \varnothing, \wp(\mathfrak{D}) \rangle \arrow[ur, shift right, two heads, "\alpha^{*}"'] \arrow[r, shift right, two heads, "\rho_{\Delta}"'] & \langle \mathtt{SSC}^{\mathtt{H}}, \subseteq, \cup, \cap, \{\varnothing\}, \wp(\mathfrak{D}) \rangle \arrow[l, shift right, "id"'] \arrow[u, shift right, two heads, "\alpha_{\star}"'] \arrow[r, shift right, two heads, "\rho_{\star}"'] & \langle \mathtt{TRC}^{\mathtt{H}}, \subseteq, \cup, \cap, \{\varnothing\}, \wp(\mathfrak{D}) \rangle \arrow[ul, shift left, two heads, "\alpha_{\star}"'] \arrow[l, shift right, "id"']
\end{tikzcd}$$

Fig. 2. Relations between hyperproperties

5 Approximating Hyperproperties Verification

In this section, we investigate how we can approximate hyperproperty verification. Let us briefly recall how we can approximate standard property verification. In order to cope with the potential non decidability of trace properties verification, approximation of systems semantics is necessary. In the standard framework of abstract interpretation [8,9] we can compute a sound over-approximation $O \supseteq [\![P]\!]$ of a system semantics allowing sound validation of trace properties (Fig. 3, part [a]). This is obtained by means of an abstraction of the concrete domain, where the abstract semantics plays the role of the over-approximation. Let P be a system, $\hat{A} \subseteq \mathtt{TRC}^{\mathtt{P}}$ an abstract domain, $\mathfrak{P} \in \mathtt{TRC}^{\mathtt{P}}$ a trace property and $[\![P]\!]^\sharp$ an abstract interpretation of $[\![P]\!]$ in $\hat{A}$, i.e., $[\![P]\!] \subseteq \hat{\gamma}([\![P]\!]^\sharp)$, then:

$$\langle \mathtt{TRC}^{\mathtt{P}}, \subseteq \rangle \underset{\hat{\alpha}}{\overset{\hat{\gamma}}{\leftrightarrows}} \langle \hat{A}, \preccurlyeq \rangle \quad \text{and} \quad \hat{\gamma}([\![P]\!]^\sharp) \subseteq \mathfrak{P} \quad \text{implies} \quad P \models \mathfrak{P}$$

Recall that, by under-approximation we can improve decidability of the confutation of a property, since if $U \subseteq [\![P]\!]$ and $U \not\subseteq \mathfrak{P}$ then we have that $[\![P]\!] \not\models \mathfrak{P}$. At this point, we can show that trace hyperproperties can be verified in the standard analysis framework based on abstract interpretation.

Proposition 3. *Let P be a system, $\hat{A} \subseteq \mathtt{TRC}^{\mathtt{P}}$ be an abstract domain, $\mathfrak{tHp} \in \mathtt{TRC}^{\mathtt{H}}$ be a trace hyperproperty and $[\![P]\!]^\sharp$ be an abstraction of $[\![P]\!]$ in $\hat{A}$, i.e., $[\![P]\!] \subseteq \hat{\gamma}([\![P]\!]^\sharp)$, then $\langle \mathtt{TRC}^{\mathtt{P}}, \subseteq \rangle \underset{\hat{\alpha}}{\overset{\hat{\gamma}}{\leftrightarrows}} \langle \hat{A}, \preccurlyeq \rangle$ and $\hat{\gamma}([\![P]\!]^\sharp) \subseteq \bigcup \mathfrak{tHp}$ implies $P \models \mathfrak{tHp}$.*

Hence, we can still use standard analysis based on over-approximation for verifying trace hyperproperties. Moreover, when dealing with confutation of properties, also in this case we can use under-approximation in the standard way, since if we have $U \subseteq [\![P]\!]$ and $U \not\subseteq \bigcup \mathfrak{tHp}$ then still we can derive that $P \not\models \mathfrak{tHp}$.

Unfortunately, when we do not have restrictions on hyperproperties, standard trace semantics, in general, does not provide enough information for approximating

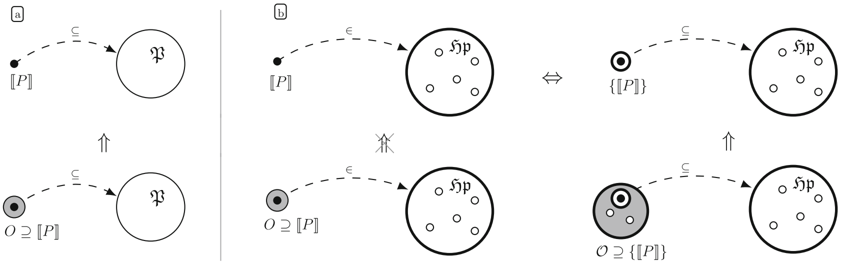
a
$\subseteq$
$\llbracket P \rrbracket$
$\mathfrak{P}$
$\Uparrow$
$\subseteq$
$O \supseteq \llbracket P \rrbracket$
$\mathfrak{P}$
b
$\in$
$\llbracket P \rrbracket$
$\mathfrak{Hp}$
$\Leftrightarrow$
$\subseteq$
$\{\llbracket P \rrbracket\}$
$\mathfrak{Hp}$
$\in$
$O \supseteq \llbracket P \rrbracket$
$\mathfrak{Hp}$
$\Uparrow$
$\subseteq$
$\mathcal{O} \supseteq \{\llbracket P \rrbracket\}$
$\mathfrak{Hp}$

Fig. 3. Over-approximation of trace properties [a] and hyperproperties [b]

verification, since $O \supseteq \llbracket P \rrbracket \wedge O \in \mathfrak{Hp} \;\not\Rightarrow\; \llbracket P \rrbracket \in \mathfrak{Hp}$ (Fig. 3, part [b] on the left). Over-approximations do not work properly because we are approximating on the wrong domain. Indeed, if we move towards $\mathtt{GEN}^{\mathtt{H}}$ (or $\mathtt{SSC}^{\mathtt{H}}$), then $\mathcal{O} \supseteq \{\llbracket P \rrbracket\} \wedge \mathcal{O} \subseteq \mathfrak{Hp} \;\Rightarrow\; \{\llbracket P \rrbracket\} \subseteq \mathfrak{Hp}$, i.e., $\llbracket P \rrbracket \in \mathfrak{Hp}$ (Fig. 3, part [b] on the right). The problem is due to the fact that the property is defined on the domain $\mathtt{GEN}^{\mathtt{H}}$, different from the domain $\mathtt{TRC}^{\mathtt{P}}$, where the system semantics is computed.

The idea we propose in the following sections, consists in moving the systems semantics on a more concrete domain, i.e., we build the semantics at the same level of the properties, namely at the *hyper* level. In this way, we can exploit the abstract interpretation framework even for approximating hyperproperties verification. Our goal is to define the system P semantics on the hyper level, i.e., we define the *hyper semantics* $\llbracket P \rrbracket_{\mathcal{H}}$ such that $\{\llbracket P \rrbracket\} \subseteq \llbracket P \rrbracket_{\mathcal{H}}$.

An over-approximation of $\llbracket P \rrbracket_{\mathcal{H}}$ clearly leads to a sound verification mechanism for hyperproperties. In fact, let P be a system, $\tilde{\mathcal{A}} \subseteq \mathtt{GEN}^{\mathtt{H}}$ be an abstract domain, $\mathfrak{Hp} \in \mathtt{GEN}^{\mathtt{H}}$ be an hyperproperty, $\llbracket P \rrbracket_{\mathcal{H}}$ be a semantics on $\mathtt{GEN}^{\mathtt{H}}$ and $\llbracket P \rrbracket^{\sharp}_{\mathcal{H}}$ be an abstract interpretation of $\llbracket P \rrbracket_{\mathcal{H}}$ in $\tilde{\mathcal{A}}$, i.e., $\llbracket P \rrbracket_{\mathcal{H}} \subseteq \tilde{\gamma}(\llbracket P \rrbracket^{\sharp}_{\mathcal{H}})$, then:

$$\langle \mathtt{GEN}^{\mathtt{H}}, \subseteq \rangle \underset{\tilde{\alpha}}{\overset{\tilde{\gamma}}{\leftrightarrows}} \langle \tilde{\mathcal{A}}, \precsim \rangle \quad \text{and} \quad \tilde{\gamma}(\llbracket P \rrbracket^{\sharp}_{\mathcal{H}}) \subseteq \mathfrak{Hp} \quad \text{imply} \quad P \models \mathfrak{Hp}$$

Hence, we build an hyper semantics of the system, and then we can over-approximate it in some abstraction of the hyper domain. This is depicted in Fig. 4, where in [a] we have the standard case and in [b] the hyper case.

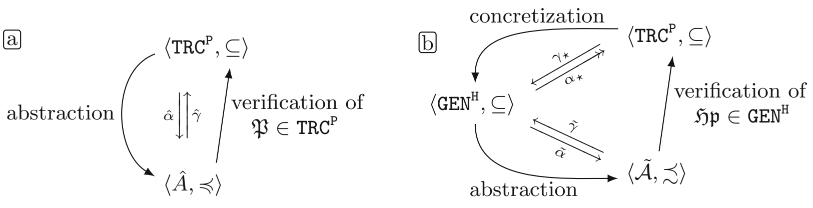
a
$\langle \mathtt{TRC}^{\mathtt{P}}, \subseteq \rangle$
abstraction
$\hat{\alpha}$
$\hat{\gamma}$
verification of
$\mathfrak{P} \in \mathtt{TRC}^{\mathtt{P}}$
$\langle \hat{A}, \preccurlyeq \rangle$
b
concretization
$\langle \mathtt{TRC}^{\mathtt{P}}, \subseteq \rangle$
$\gamma\star$
$\alpha\star$
$\langle \mathtt{GEN}^{\mathtt{H}}, \subseteq \rangle$
verification of
$\mathfrak{Hp} \in \mathtt{GEN}^{\mathtt{H}}$
$\tilde{\gamma}$
$\tilde{\alpha}$
abstraction
$\langle \tilde{\mathcal{A}}, \precsim \rangle$

Fig. 4. Verification (abstract interpretation) of properties [a] and hyperproperties [b]

6 Hyperhierarchy of Semantics

In Sect. 2.3 we introduced the hierarchy of semantics proposed in [7], where most well known semantics have been related by Galois insertions. In this section, we aim at extending this hierarchy in order to include an hyper level of semantics suitable for hyperproperties verification. The intuition of lifting the classical hierarchy of semantics to sets of sets was already present in [3], where it was just sketched. Here we analyze the problem in a deeper and comprehensive way. Note that, as observed in Sect. 4.2, we have different notions of hyperproperties, implying different possible approaches for verification. We do not have precisely the same distinction when dealing with systems semantics.

6.1 Defining Hypersemantics

In the following, we indicate with $[\![P]\!]$ a generic standard semantics of the system P, namely an element of the standard hierarchy, as we have done in Sect. 4. So, for instance, $[\![P]\!]$ can stand for $\tau^{\vec{\infty}}[P]$, or it can stand for $\tau^{+}[P]$, etc.

Subset-Closed and Generic Hypersemantics. The first level comprises subset-closed systems semantics. This means that every element of this hierarchy, which is parametric by systems denotations ($\mathfrak{D}$) as in the standard case, is in the set $\mathtt{SSC}^{\mathtt{H}}$. It turns out that, given a system P, its subset-closed hypersemantics is $[\![P]\!]_{\triangle} = \wp([\![P]\!])$, which is indeed its strongest subset-closed hyperproperty. This happens because any semantics have a maximal set of computations, therefore an $\mathtt{SSC}^{\mathtt{H}}$ semantics is in particular a boolean algebra.

The second level comprises generic systems hypersemantics. This means that every element of this hierarchy, which is again parametric on systems denotations ($\mathfrak{D}$), is in $\mathtt{GEN}^{\mathtt{H}}$. It turns out that, given a system P, its generic hypersemantics is $[\![P]\!]_{\diamond} = \{[\![P]\!]\}$, which is indeed its strongest generic hyperproperty.

It is worth nothing that, $[\![P]\!]_{\triangle} \in \mathtt{SSC}^{\mathtt{H}}$ and $[\![P]\!]_{\diamond} \in \mathtt{GEN}^{\mathtt{H}}$ do not give us more information on the executions of P than $[\![P]\!]$, being isomorphic to $[\![P]\!]$. Namely these parallel hierarchies does not provide different observables, but only new verification methods for hyperproperties. In particular, over-approximations of hypersemantics on these more expressive semantic levels, provide verification methods for subset-closed and generic hyperproperties. We cannot verify these hyperproperties within the standard hierarchy of semantics.

Post/Pre Hypersemantics. In the previous sections, we considered only hypersemantics isomorphic to standard ones. It is clear, that the hyper level is indeed strictly more concrete than the standard level, hence we aim at defining hyper semantics strictly more expressive than standard ones. In particular, we can extends to the hyper levels both the maximal trace semantics and the partial trace semantics and we observe how we can exploit the expressiveness of these semantics when dealing with hyperproperties verification.

The *Post hypersemantics* $\tau^{\vec{\infty}}_{\texttt{post}}$ is defined as:

$$\tau^{\vec{\infty}}_{\texttt{post}} \overset{\text{def}}{=} \Big\{ \{ \textstyle\bigcup_{n>0} \tau^{\vec{n}}_{X} \cup \tau^{\vec{\omega}} \} \;\Big|\; X \subseteq \varOmega \Big\} \qquad \text{where } \; \tau^{\vec{n}}_{X} \overset{\text{def}}{=} \{\sigma \in \tau^{\vec{n}} \mid \sigma_{n-1} \in X\}$$

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

$$\tau^{\vec{\propto}}_{\texttt{pre}} \overset{\text{def}}{=} \Big\{ \{ \textstyle\bigcup_{n>0} \tau^{\dot{\vec{n}}}_{X} \} \;\Big|\; X \subseteq \varUpsilon \wedge X \neq \varnothing \Big\} \qquad \text{where } \; \tau^{\dot{\vec{n}}}_{X} \overset{\text{def}}{=} \{\sigma \in \tau^{\dot{\vec{n}}} \mid \sigma_{0} \in X\}$$

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

Example 2. As example, consider the transition system with $\Sigma = \{a, b, c, d, e\}$, $\tau = \{\langle a, b\rangle, \langle a, c\rangle, \langle b, d\rangle, \langle c, c\rangle, \langle e, b\rangle, \langle e, e\rangle\}$, $\varUpsilon = \{a, e\}$ and $\varOmega = \{d\}$. Then

$$\begin{aligned} \tau^{\vec{\infty}} &= \{d, bd, abd\} \cup \{e^n bd\}_{n\geq 1} \cup \{c^\omega, ac^\omega, e^\omega\} \\ \tau^{\vec{\propto}} &= \{a, ab, abd\} \cup \{ac^n\}_{n\geq 1} \cup \{e^n\}_{n\geq 1} \cup \{e^n b\}_{n\geq 1} \cup \{e^n bd\}_{n\geq 1} \end{aligned}$$

The hyper versions are

$$\begin{aligned} \tau^{\vec{\infty}}_{\texttt{post}} &= \big\{\tau^{\vec{\infty}}, \{c^\omega, ac^\omega, e^\omega\}\big\} \\ \tau^{\vec{\propto}}_{\texttt{pre}} &= \big\{\tau^{\vec{\propto}}, \{a, ab, abd\} \cup \{ac^n\}_{n\geq 1}, \{e^n\}_{n\geq 1} \cup \{e^n b\}_{n\geq 1} \cup \{e^n bd\}_{n\geq 1}\big\} \end{aligned}$$

being $\wp(\varOmega) = \big\{\{d\}, \varnothing\big\}$ and $\wp(\varUpsilon) \setminus \{\varnothing\} = \big\{\{a, e\}, \{a\}, \{e\}\big\}$.

These hypersemantics can be used for *partially verifying hyperproperties*, since they provide the semantics parametrically on the subsets of blocking/initial states. Suppose that, instead of checking *whether* a system fulfills an hyperproperty $\mathfrak{Hp}$, we want to check *when* a system fulfills it. The problem boils down to analyze the intersection $\tau^{\vec{\infty}}_{\texttt{post}} \cap \mathfrak{Hp}$ [or $\tau^{\vec{\propto}}_{\texttt{pre}} \cap \mathfrak{Hp}$]. If the intersection is $\varnothing$ then the answer is "never", if the answer is $\tau^{\vec{\infty}}_{\texttt{post}}$ [or $\tau^{\vec{\propto}}_{\texttt{pre}}$] then $P \models \mathfrak{Hp}$, otherwise we have that for particular final states [initial states] the system satisfies the hyperproperty. Hence we have a form of *partial satisfiability.* This is in practice useful, for example when we want to know under what conditions we can still use an unsafe system.

The Hyperhierarchy. Up to now, we simply reasoned on single semantics. Finally, we can show that the whole hierarchy of standard semantics can be lifted on the hyper levels, preserving all the abstraction relations between semantics. In the standard hierarchy, $\tau^{\vec{\infty}}$ and $\tau^{\vec{+}}$ (and hence all their relational/denotational abstractions) are *backward* semantics in the sense they are suffix-closed [11]. This means that they represents systems executions with complete traces and all their suffixes. Instead, the semantics $\tau^{\vec{\propto}}$ is *forward* in the sense it is prefix-closed [13]. This means that it represents systems executions with all the partial computations starting from initial states (i.e., trace prefixes).

Note that all the semantics in the standard hierarchy are abstractions of $\tau^{\vec{\infty}}$ and, analogously, every hypersemantics is an abstraction of $\tau^{\vec{\infty}}_{\mathtt{post}}$.

Proposition 4. *Let $\mathfrak{y} \in \{\vec{\infty}, \vec{+}, \vec{\propto}, \infty, +, \natural, \flat\}$, let α be such that $\tau^{\mathfrak{y}} = \alpha(\tau^{\vec{\infty}})$ in the standard hierarchy of semantics, and let $\alpha{\upharpoonleft} \overset{\mathsf{def}}{=} \lambda\mathcal{X} \,.\, \{\alpha(X) \mid X \in \mathcal{X}\}$, then:*

$$\tau^{\mathfrak{y}}_{\mathtt{post}} = \alpha{\upharpoonleft}(\tau^{\vec{\infty}}_{\mathtt{post}}) \textit{ and } \alpha \circ \alpha_\star = \alpha_\star \circ \alpha{\upharpoonleft}$$

The subset-closed ($\triangle$) and generic ($\diamond$) hypersemantics are isomorphic to the standard ones, trough $\langle \alpha_\star, \gamma_\star \rangle$ and $\langle \alpha_\star, \lambda X \,.\, \{X\} \rangle$ respectively. This means that for these hypersemantics the commutativity trivially holds. So, lifting to sets the abstraction function used to go from a semantics to another semantics, in the standard hierarchy, results in an abstraction between the respective hypersemantics at the hyper level. Proposition 4 justifies Fig. 1, where an arrow between semantics means that there is an abstraction relation, while a double arrow means that the semantics are isomorphic. On the left we have the standard hierarchy and on the right the hyper levels. The central level represents subset-closed ($\triangle$) and generic ($\diamond$) hypersemantics, which are isomorphic to standard semantics. This allows us, with the same information, to gain expressiveness in verification. On the right, we have the level of post/pre hypersemantics, namely semantics which contains strictly more information w.r.t. the standard ones and which can be used for partial verification. From these hypersemantics we obtain the standard ones through the abstraction $\langle \alpha_\star, \gamma_\star \rangle$ and hence, by composition with the isomorphism, also subset-closed ($\triangle$) and generic ($\diamond$) hypersemantics are abstractions of them.

6.2 Computing Hypersemantics

In this section, we show how we can compute the semantics at the hyper levels, similarly to what happens in the standard hierarchy of semantics [7], where each semantics is obtained as fixpoint of a monotone operator.

Computing Hypersemantics by Using bcc *and Additive Lift.* Suppose we are interested in computing the standard semantics at the hyper level. In this case, our aim is simply to emulate the standard semantics computation on the hyper level. This may be considered useful for approximating computation when dealing with hyperproperty verification, as explained in Sect. 5. In this case we have to *transfer* the fixpoint computation from the abstract domain of standard semantics, to the concrete domain of hypersemantics, and we can follow two possible ways: we can use the *best complete concretization* (bcc) of the standard semantic operator, or we can *lift* the operator to sets. Basically, we aim at computing by fixpoint a semantics $[\![P]\!]_{\mathcal{H}}$ (one of the semantics in Fig. 1, on the central level), namely we want to find a monotone operator $F_{\mathcal{H}} : \wp(\wp(\mathfrak{D})) \rightarrow \wp(\wp(\mathfrak{D}))$, such that $[\![P]\!]_{\mathcal{H}} = \mathit{lfp}\ F_{\mathcal{H}}$, built on top of the standard semantics operator F.

First, consider $F_{\mathcal{H}} \overset{\mathsf{def}}{=} F_{\vartriangle} = \gamma_\star \circ F \circ \alpha_\star$ (namely we apply Theorem 2 considering $F_{\vartriangle}$ as the best complete concretization of F). Since $\gamma_\star$ is a strict Scott-continuous concretization map between $\langle \mathtt{TRC}^{\mathtt{P}}, \subseteq, \cup, \cap, \varnothing, \mathfrak{D}\rangle$ and $\langle \mathtt{SSC}^{\mathtt{H}}, \subseteq, \cup, \cap, \{\varnothing\}, \wp(\mathfrak{D})\rangle$ and the forward completeness holds by definition, we can apply Theorem 1 and hence $\gamma_\star(\mathit{lfp}^{\subseteq}_{\varnothing} F) = \mathit{lfp}^{\subseteq}_{\{\varnothing\}} F_{\vartriangle}$, i.e., $\gamma_\star([\![P]\!]) = \wp([\![P]\!]) = [\![P]\!]_{\vartriangle} = \mathit{lfp}^{\subseteq}_{\{\varnothing\}} F_{\vartriangle}$. Indeed $F_{\vartriangle}$ is $\subseteq$-monotone and $F^0_{\vartriangle}(\{\varnothing\}) = \{\varnothing\} \subseteq F^1_{\vartriangle}(\{\varnothing\}) = \wp(F(\varnothing)) \subseteq F^2_{\vartriangle}(\{\varnothing\}) = \wp(F^2(\varnothing)) \subseteq \ldots F^n_{\vartriangle}(\{\varnothing\}) = \wp(F^n(\varnothing))$ since, for every n, $F^n(\varnothing) \subseteq F^{n+1}(\varnothing)$. It should be clear that, with this operator, we move inside elements of $\mathtt{TRC}^{\mathtt{H}}$, which is a strict subset of $\mathtt{SSC}^{\mathtt{H}}$.

The second choice consists in defining $F_{\mathcal{H}}$ as the additive lift of F, i.e., $F_{\mathcal{H}} \overset{\mathsf{def}}{=} F_{\diamond} = \lambda \mathcal{X} . \{F(X) \mid X \in \mathcal{X}\}$. Unfortunately, the lift does not guarantee monotonicity. Indeed the iterates of $F_{\diamond}$ from the bottom are: $F^0_{\diamond}(\varnothing) = \varnothing$, $F^1_{\diamond}(\varnothing) = \{\varnothing\}$, $F^2_{\diamond}(\varnothing) = \{F(\varnothing)\}$, $\ldots F^n_{\diamond}(\varnothing) = \{F^{n-1}(\varnothing)\}$. Clearly the iterates do not form an increasing $\subseteq$-chain and so $\langle F_{\diamond}, \mathtt{GEN}^{\mathtt{H}}, \subseteq \rangle$ is not a fixpoint semantics specification. In this case we need to change the computational domain. Let us consider the following computational order $\subseteq_\star$:

$$\mathcal{X} \subseteq_\star \mathcal{Y} \overset{\mathsf{def}}{=} \begin{array}{c} (\mathcal{X} = \varnothing \ \vee (\forall X \in \mathcal{X}\, \exists Y \in \mathcal{Y} . X \subseteq Y)) \wedge \\ (\mathcal{Y} = \varnothing \vee ((\forall Y \in \mathcal{Y}\, \exists X \in \mathcal{X} . Y \subseteq X) \Rightarrow \mathcal{X} = \mathcal{Y})) \end{array} \tag{1}$$

Namely, for each element $X \in \mathcal{X}$ there exists an element of $\mathcal{Y}$ in the $\subseteq$ relation with X (the second conjunction just forces antisymmetry). Furthermore, the equalities with the empty-set add the axiom $\varnothing \subseteq_\star \varnothing \subseteq_\star \mathcal{X}$, for any $\mathcal{X}$. The bottom is $\varnothing$ and the (partial) least upper bound is $\uplus_\star$ defined as:

$$\mathcal{X} \uplus_\star \mathcal{Y} \overset{\mathsf{def}}{=} \begin{array}{l} \{X \cup Y \mid X \in \mathcal{X} \wedge Y \in \mathcal{Y} \wedge (X \subseteq Y \vee Y \subseteq X)\} \cup \\ \{X \mid X \in \mathcal{X} \wedge (\mathcal{Y} = \varnothing \vee \forall Y \in \mathcal{Y} . (X \not\subseteq Y \wedge Y \not\subseteq X))\} \cup \\ \{Y \mid Y \in \mathcal{Y} \wedge (\mathcal{X} = \varnothing \vee \forall X \in \mathcal{X} . (Y \not\subseteq X \wedge X \not\subseteq Y))\} \end{array} \tag{2}$$

The lub makes the union of the elements of $\mathcal{X}$ and $\mathcal{Y}$ which are in relation $\subseteq$, and adds all the other elements of both sets, as they are. The domain $\langle \mathtt{GEN}^{\mathtt{H}}, \subseteq_\star, \uplus_\star, \varnothing\rangle$ is a pointed DCPO with (partial) lub and bottom, indeed we have $\varnothing \subseteq_\star \mathcal{X}$ for every $\mathcal{X} \in \mathtt{GEN}^{\mathtt{H}}$ and $\mathcal{X} \subseteq_\star \mathcal{Y}$ implies $\mathcal{X} \uplus_\star \mathcal{Y} = \mathcal{Y}$. Then we have that $\langle F_{\diamond}, \mathtt{GEN}^{\mathtt{H}}, \subseteq_\star \rangle$ is a fixpoint semantic specification, since $F_{\diamond}$ is $\subseteq_\star$-monotone.

Proposition 5. *If* $\langle F, \mathtt{TRC}^{\mathtt{P}}, \subseteq \rangle$ *and* $[\![P]\!] = \mathit{lfp}^{\subseteq}_{\varnothing} F = \bigcup_{n>0} F^n(\varnothing)$ *then we have:* $\langle F_{\diamond}, \mathtt{GEN}^{\mathtt{H}}, \subseteq_\star \rangle$ *and* $[\![P]\!]_{\diamond} = \mathit{lfp}^{\subseteq_\star}_{\varnothing} F_{\diamond} = \biguplus_{\star\, n>0} F^n_{\diamond}(\varnothing) = \{[\![P]\!]\}$.

Also in this case we simply compute standard semantics on the hyperlevel, but we do not really exploit the more concrete level at which we are computing the semantics. In other words, as before, we are emulating the standard computation on the generic hypersemantics domain. Indeed, the semantics $[\![P]\!]_{\diamond}$ is isomorphic to the standard semantics $[\![P]\!]$.

Computing Post/Pre Hypersemantics. Here, we aim at exploiting the concrete domain on which we are computing by defining new operators moving freely among elements of $\mathtt{GEN}^{\mathtt{H}}$ and not only on elements of $\mathtt{TRC}^{\mathtt{H}}$. We consider only one case for the backward hypersemantics, the most concrete, but the others are similar. We take $\mathfrak{D} = \mathcal{S}^{\vec{\infty}}$, so let

$$F^{\vec{\infty}}_{\mathtt{post}} \overset{\mathsf{def}}{=} \lambda \mathcal{X} \,.\, \{X \cup \Sigma^{\vec{\omega}} \mid X \subseteq \tau^{\vec{1}}\} \mathbin{\sqcup_{\star}}^{\vec{\infty}} \{X \sqcup^{\vec{\infty}} \tau^{\dot{\vec{2}}} \frown X \mid X \in \mathcal{X}\}$$

Then we have that $\tau^{\vec{\infty}}_{\mathtt{post}} = \mathit{lfp}^{\sqsubseteq^{\vec{\infty}}_{\star}}_{\{\Sigma^{\vec{\omega}}\}} F^{\vec{\infty}}_{\mathtt{post}} = \bigsqcup_{\star}{}^{\vec{\infty}}_{n>0} {F^{\vec{\infty}}_{\mathtt{post}}}^{n}(\{\Sigma^{\vec{\omega}}\})$. Where $\sqsubseteq^{\vec{\infty}}_{\star}$ is defined as in Eq. 1, substituting $\subseteq$ with $\sqsubseteq^{\vec{\infty}}$ in the definition, the lub $\sqcup_{\star}^{\vec{\infty}}$ is defined as in Eq. 2, substituting $\cup$ with $\sqcup^{\vec{\infty}}$ in the definition and the bottom is $\{\Sigma^{\vec{\omega}}\}$. Analogously, we can do the same for the forward case. Here we have only one case, hence we take $\mathfrak{D} = \mathcal{S}^{\vec{\propto}}$ and we have $\tau^{\vec{\propto}}_{\mathtt{pre}} = \mathit{lfp}^{\subseteq_{\star}}_{\varnothing} F^{\vec{\propto}}_{\mathtt{pre}} = \bigcup_{\star}{}_{n>0} {F^{\vec{\propto}}_{\mathtt{pre}}}^{n}(\varnothing)$, where

$$F^{\vec{\propto}}_{\mathtt{pre}} \overset{\mathsf{def}}{=} \lambda \mathcal{X} \,.\, (\wp(\varUpsilon) \setminus \{\varnothing\}) \mathbin{\cup_{\star}} \{X \cup X \frown \tau^{\dot{\vec{2}}} \mid X \in \mathcal{X}\}$$

We can show that the standard operator $F^{\vec{\infty}}$ is the fixpoint transfer (on the abstract domain of standard semantics), by means of the Galois insertion $\langle \alpha_{\star}, \gamma_{\star} \rangle$, of the concrete semantic operator $F^{\vec{\infty}}_{\mathtt{post}}$. Analogously, transferring the operator $F^{\vec{\propto}}_{\mathtt{pre}}$ on the standard semantic domain, we fall back on $F^{\vec{\propto}}$.

Theorem 4. *The following hold:*

1. $\mathit{lfp}^{\sqsubseteq^{\vec{\infty}}}_{\Sigma^{\vec{\omega}}} F^{\vec{\infty}} = \alpha_{\star}(\mathit{lfp}^{\sqsubseteq^{\vec{\infty}}_{\star}}_{\{\Sigma^{\vec{\omega}}\}} F^{\vec{\infty}}_{\mathtt{post}}) = \alpha_{\star}(\tau^{\vec{\infty}}_{\mathtt{post}})$ *and* $F^{\vec{\infty}} \circ \alpha_{\star} = \alpha_{\star} \circ F^{\vec{\infty}}_{\mathtt{post}}$.
2. $\mathit{lfp}^{\subseteq}_{\varnothing} F^{\vec{\propto}} = \alpha_{\star}(\mathit{lfp}^{\subseteq_{\star}}_{\varnothing} F^{\vec{\propto}}_{\mathtt{pre}}) = \alpha_{\star}(\tau^{\vec{\propto}}_{\mathtt{pre}})$ *and* $F^{\vec{\propto}} \circ \alpha_{\star} = \alpha_{\star} \circ F^{\vec{\propto}}_{\mathtt{pre}}$.

7 Concluding: Hypersemantics Around Us

In this work, we have introduced a formal framework for modeling system semantics at the same level of hyperproperties. These more expressive semantics not only allow us to provide weaker forms of satisfiability, as shown in Sect. 6, but provide a promising methodology allowing us to lift static analysis (for hyperproperties) directly at the hyper level. We believe that this approach could provide a deep insight and useful formal tools also for tackling the problem of *analyzing analyzers*, aiming at systematically analyzing static analyses [16].

Finally, we present two verification methods that, explicitly or implicitly, can be generalized in our work. The first is an ad-hoc hypersemantics of programs [3], made for the verification of information flow policies. The second is the classical framework of static analysis for program properties verification [9].

7.1 Hypercollecting Semantics

As observed in the previous sections, there is an hyper hierarchies of semantics that mimic the standard one in more expressive domains. This gain of expressiveness allows us to verify (by over-approximation) *hyper*properties.

To the best of our knowledge, the only work that perform verification by mean of abstract interpretation exploiting the full expressiveness of hyperproperties is [3]. They deal with information flow policies that are k-hypersafety and they focus on the definition of the abstract domains over sets of sets needed for the analysis. They proposed an ad-hoc hypersemantics (termed *hypercollecting semantics*) to show how to apply the abstract interpretation framework. This semantics is computed denotationally starting from the code of the program to analyze (their $\mathsf{systems}$ are programs of a toy programming language) and it is used to verify some information flow policies, such as some formulations of non-interference. In order to perform information flow verification, they consider the domain of finite relational traces, namely $\wp(\mathcal{S} \times \mathcal{S})$ (their $\wp(\mathbf{Trc})$), or better its hyper version, namely $\wp(\wp(\mathcal{S} \times \mathcal{S}))$ (their $\wp(\wp(\mathbf{Trc}))$). States are maps from variables to values, i.e., $\mathcal{S} = \mathrm{Var} \to \mathrm{Val}$ (their **States**). Their semantics computes, denotationally, the *angelic relational semantics* $\tau^{+}[P]$, in the Cousot hierarchy. More formally, for every program P, the collecting semantics $\{\!|P|\!\}\mathbf{IniTrc}$ of [3], where **IniTrc** is the set of all possible inputs[9], is $\tau^{+}[P]$ in the standard hierarchy of semantics ([3], Sect. 2). Then they propose the hypercollecting semantics $(\!|\cdot|\!)$ such that $\{\!|P|\!\}X \in (\!|P|\!)\{X\}$ (this implies $\{\tau^{+}[P]\} \subseteq (\!|P|\!)\{\mathbf{IniTrc}\}$).

Proposition 6. $(\!|P|\!)\wp(\mathbf{IniTrc}) = \tau^{+}_{\Delta}[P]$.

Hence, the hypercollecting semantics proposed in [3], starting from $\wp(\mathbf{IniTrc})$[10], is exactly the hyper angelic relational semantics $\tau^{+}_{\Delta}[P]$ in our hyper hierarchy.

Let us consider, now, the computation of the semantics for a program P for the verification of a given property. We can observe that Proposition 6 guarantees the equivalence of these two semantics for property verification only for subset-closed hyper property, while for general hyperproperty the two semantics are not comparable. In particular, let $\mathfrak{c}\mathfrak{H}\mathfrak{p} \in \mathtt{SSC}^{\mathtt{H}}$, we can observe that

$$P \models \mathfrak{c}\mathfrak{H}\mathfrak{p} \;\Leftrightarrow\; \tau^{+}_{\Delta}[P] \subseteq \mathfrak{c}\mathfrak{H}\mathfrak{p} \;\Leftrightarrow\; (\!|P|\!)\wp(\mathbf{IniTrc}) \subseteq \mathfrak{c}\mathfrak{H}\mathfrak{p} \;\Leftrightarrow\; (\!|P|\!)\{\mathbf{IniTrc}\} \subseteq \mathfrak{c}\mathfrak{H}\mathfrak{p}$$

where the first implication holds for our definition of verification, the second holds by Proposition 6 and the third one holds since the hyperproperty is subset-closed. On the other hand, if we consider a generic hyper property $\mathfrak{H}\mathfrak{p} \in \mathtt{GEN}^{\mathtt{H}}$ the last implication does not hold in general. In particular, the hypercollecting semantics is the additive lift of the standard semantics for all commands except the **while**. Indeed, as also the authors underline, when the program contains a loop their semantics adds the sets of traces that exit the loop at each iteration ([3], Sect. 4). For this reason, the hypercollecting semantics is not complete for generic hyperproperties verification.

[9] Precisely is the set of all pairs $\langle\sigma, \sigma\rangle$ where σ is an initial state.

[10] $\wp(\mathbf{IniTrc})$ is the concretization of **IniTrc** to set of sets, i.e., $\wp(\mathbf{IniTrc}) = \gamma_{\star}(\mathbf{IniTrc})$.

Example 3. Let $P \overset{\text{def}}{=} \textbf{while}\,(x < 2)\,\textbf{do}\,\{\, x := x + 1 \,\}$, with the unique variable x ranging over the values $\{0, 1, 2\}$. Then $\textbf{IniTrc} = \{\langle 0, 0\rangle, \langle 1, 1\rangle, \langle 2, 2\rangle\}$, where $\langle v, v'\rangle$ is a concise representation of the couple of mapping (i.e., **States**) $\langle x \mapsto v, x \mapsto v'\rangle$. The angelic relational semantics of c is $\tau^{+}[P] = \{\langle 0, 2\rangle, \langle 1, 2\rangle, \langle 2, 2\rangle\}$, which is exactly $\{\!|P|\!\}\textbf{IniTrc}$. The hypercollecting semantics $(\!|P|\!)_{\wp}(\textbf{IniTrc})$ is computed as follow. The least fixpoint of the while is the set of sets of traces:

$$\wp(\textbf{IniTrc}) \cup \left\{ \begin{array}{c} \{\langle 0,1\rangle\}, \{\langle 1,2\rangle\}, \{\langle 0,1\rangle, \langle 1,2\rangle\}, \{\langle 0,1\rangle, \langle 2,2\rangle\}, \{\langle 1,2\rangle, \langle 2,2\rangle\}, \{\langle 0,2\rangle\}, \\ \{\langle 0,1\rangle, \langle 1,2\rangle, \langle 2,2\rangle\}, \{\langle 0,2\rangle, \langle 2,2\rangle\}, \{\langle 0,2\rangle, \langle 1,2\rangle\}, \{\langle 0,2\rangle, \langle 1,2\rangle, \langle 2,2\rangle\} \end{array} \right\}$$

At the while exit we have to keep only the traces making false the guard [3], i.e.,

$$(\!|P|\!)_{\wp}(\textbf{IniTrc}) = \left\{ \begin{array}{c} \varnothing, \{\langle 2,2\rangle\}, \{\langle 1,2\rangle\}, \{\langle 1,2\rangle, \langle 2,2\rangle\}, \{\langle 0,2\rangle\}, \{\langle 0,2\rangle, \langle 2,2\rangle\}, \\ \{\langle 0,2\rangle, \langle 1,2\rangle\}, \{\langle 0,2\rangle, \langle 1,2\rangle, \langle 2,2\rangle\} \end{array} \right\}$$

which is exactly $\wp(\{\!|P|\!\}\textbf{IniTrc}) = \wp(\tau^{+}[P]) = \tau^{+}_{\vartriangle}[P]$.

7.2 Standard Static Program Analysis

In the literature, standard static program analysis has been modeled as reachability analysis, since the collected values are all the *reachable* values for a variable. Assume that $\langle \varSigma, \varUpsilon, \varOmega, \tau\rangle$ is the transition system associated to the program P, and $\varPsi \subseteq \varUpsilon$ is a subset of initial states. Static analysis can be seen as the characterization, potentially approximated, of the set of reachable states from initial $\varPsi$, i.e., $\tau^{\mathrm{r}}(\varPsi) = \{\varsigma \mid \exists \sigma \in \tau^{\overrightarrow{\infty}}, i \in \mathbb{N} \,.\, \sigma_0 \in \varPsi \wedge \sigma_i = \varsigma\}$, which provides a, potentially approximated, invariant of the program [9]. In order to properly model *flow-sensitive* static analysis, where we look for invariants for each program point, we can simply consider a more concrete definition of state, which is not simply a memory, i.e., an element of $\mathbb{M} = \mathrm{Var} \to \mathrm{Val}$, but it is a pair associating with each program point a memory [9]. Formally, given a program P, its possible states are $\varSigma_P \overset{\text{def}}{=} \mathbb{L}_P \times \mathbb{M}$, where $\mathbb{L}_P$ is the set of program points in P. When we move towards approximation, instead of manipulating states we manipulate sets of states, i.e., elements of $\wp(\varSigma_P)$, for which holds the following

$$\wp(\mathbb{L}_P \times \mathbb{M}) \cong \mathbb{L}_P \to \wp(\mathbb{M}) = \mathbb{L}_P \to \wp(\mathrm{Var} \to \mathrm{Val})$$

Let $\iota : \wp(\varSigma_P) \to (\mathbb{L}_P \to \wp(\mathrm{Var} \to \mathrm{Val}))$ be such an isomorphism, then $\iota(\tau^{\mathrm{r}}(\varPsi))$ is a map associating each program point with the set of all "reached" *memories*, in the computations starting from $\varPsi$. In [20] the author shows that this semantics corresponds to the solution of a system of equations generated from the program syntax. Static analysis abstracts this semantics considering the map associating with each variable all the *values* "reached", for each program point, in the computations starting in $\varPsi$. This abstraction is $\alpha_c = \lambda f \,.\, (\lambda l \,.\, \dot{\bigvee} f(l))$, where $\dot{\bigvee}\{g_i\} \overset{\text{def}}{=} \lambda x \,.\, \bigcup_i g_i(x)$. So the composition $\alpha_c \circ \iota$ is a function in $\wp(\varSigma_P) \to (\mathbb{L}_P \to (\mathrm{Var} \to \wp(\mathrm{Val}))$. We denote with $\alpha_{\iota,c}$ this composition.

Example 4. Consider a program with two variables, x and y, the memory is the association of a natural value to these variables, i.e., $[x \mapsto v_1, y \mapsto v_2]$, that we denote concisely with $(v_1; v_2)$. A state is an association between a program point and a memory, i.e., ${}^{\mathbf{i}}m_i$ meaning that with the **i**-th program point is associated the memory m_i. Hence, consider the following transition system: (suppose we have only three program points)

$$\Sigma = \left\{ \begin{array}{l} \overbrace{\langle {}^{\mathbf{1}}(1;2), {}^{\mathbf{2}}(1;3), {}^{\mathbf{3}}(2;3)\rangle}^{a}, \overbrace{\langle {}^{\mathbf{1}}(1;2), {}^{\mathbf{2}}(1;4), {}^{\mathbf{3}}(2;3)\rangle}^{b}, \overbrace{\langle {}^{\mathbf{1}}(1;2), {}^{\mathbf{2}}(1;4), {}^{\mathbf{3}}(3;4)\rangle}^{c} \\ \underbrace{\langle {}^{\mathbf{1}}(2;2), {}^{\mathbf{2}}(2;3), {}^{\mathbf{3}}(3;3)\rangle}_{d}, \underbrace{\langle {}^{\mathbf{1}}(2;2), {}^{\mathbf{2}}(2;4), {}^{\mathbf{3}}(3;3)\rangle}_{e}, \underbrace{\langle {}^{\mathbf{1}}(2;2), {}^{\mathbf{2}}(2;4), {}^{\mathbf{3}}(4;4)\rangle}_{f} \end{array} \right\}$$

$$\Upsilon = \{a, d\} \quad \Omega = \{c, f\} \quad \tau = \{\langle a, b\rangle, \langle b, c\rangle, \langle d, e\rangle, \langle e, f\rangle\}$$

Hence, $\alpha_{\iota,c}(\Sigma) = \langle {}^{\mathbf{1}}(\{1,2\};\{2\}), {}^{\mathbf{2}}(\{1,2\};\{3,4\}), {}^{\mathbf{3}}(\{2,3,4\};\{3,4\})\rangle$.

At this point, we can observe that the semantics of an (abstract) interpreter of a program P is an abstraction of the hypersemantics of P. First of all, note that $\tau^{\mathrm{r}}(\Psi)$ is an abstraction of $\tau^{\vec{\propto}}$, through the function $\lambda X \,.\, \alpha_r(\{\sigma \in X \mid \sigma_0 \in \Psi\})$, where $\alpha_r \overset{\mathsf{def}}{=} \lambda X \,.\, \{\varsigma \mid \exists \sigma \in X, i \in \mathbb{N} \,.\, \sigma_i = \varsigma\}$ [13]. Analogously, we show that the semantics of an abstract interpreter, associating with each possible subset of initial states, the corresponding reachable states, is an abstraction of $\tau^{\vec{\propto}}_{\mathtt{pre}} \subseteq \wp(\Sigma^{\vec{+}}_P)$. As usual, we obtain *abstract* invariants in the abstract domain $\mathcal{A}$ exploiting a Galois insertion $\langle \wp(\mathrm{Val}), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq \rangle$.

Proposition 7. *The semantics of the abstract interpreter w.r.t. abstract domain* $\mathcal{A}$ *is* $\alpha^{\mathcal{A}}_{\iota,c}{\restriction} \circ \alpha_r{\restriction}(\tau^{\vec{\propto}}_{\mathtt{pre}})$[11]*, i.e., it is an abstraction of the hypersemantics* $\tau^{\vec{\propto}}_{\mathtt{pre}}$.

Example 5. Consider Example 4. Then $\tau^{\vec{\propto}} = \{a, ab, abc, d, de, def\}$ and $\tau^{\vec{\propto}}_{\mathtt{pre}} = \{\{a, ab, abc\}, \{d, de, def\}, \tau^{\vec{\propto}}\}$. We do not consider any abstraction $\mathcal{A}$, then:

$$\begin{aligned} &\alpha_{\iota,c} \circ \alpha_r(\tau^{\vec{\propto}}) = \alpha_{\iota,c}(\{a, b, c, d, e, f\}) = \langle {}^{\mathbf{1}}(\{1,2\};\{2\}), {}^{\mathbf{2}}(\{1,2\};\{3,4\}), {}^{\mathbf{3}}(\{2,3,4\};\{3,4\})\rangle \\ &\alpha_{\iota,c} \circ \alpha_r(\{a, ab, abc\}) = \alpha_{\iota,c}(\{a, b, c\}) = \langle {}^{\mathbf{1}}(\{1\};\{2\}), {}^{\mathbf{2}}(\{1\};\{3,4\}), {}^{\mathbf{3}}(\{2,3\};\{3,4\})\rangle \\ &\alpha_{\iota,c} \circ \alpha_r(\{d, de, def\}) = \alpha_{\iota,c}(\{d, e, f\}) = \langle {}^{\mathbf{1}}(\{2\};\{2\}), {}^{\mathbf{2}}(\{2\};\{3,4\}), {}^{\mathbf{3}}(\{3,4\};\{3,4\})\rangle \end{aligned}$$

Hence, the set of invariants, depending on the set of initial states, is:

$$\alpha_{\iota,c}{\restriction} \circ \alpha_r{\restriction}(\tau^{\vec{\propto}}_{\mathtt{pre}}) = \left\{ \begin{array}{l} \langle {}^{\mathbf{1}}(\{1,2\};\{2\}), {}^{\mathbf{2}}(\{1,2\};\{3,4\}), {}^{\mathbf{3}}(\{2,3,4\};\{3,4\})\rangle, \\ \langle {}^{\mathbf{1}}(\{1\};\{2\}), {}^{\mathbf{2}}(\{1\};\{3,4\}), {}^{\mathbf{3}}(\{2,3\};\{3,4\})\rangle, \\ \langle {}^{\mathbf{1}}(\{2\};\{2\}), {}^{\mathbf{2}}(\{2\};\{3,4\}), {}^{\mathbf{3}}(\{3,4\};\{3,4\})\rangle, \end{array} \right\}$$

References

1. Agrawal, S., Bonakdarpour, B.: Runtime verification of k-safety hyperproperties in HyperLTL. In: IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, 27 June–1 July, 2016, pp. 239–252 (2016). http://dx.doi.org/10.1109/CSF.2016.24
2. Alpern, B., Schneider, F.B.: Defining liveness. Inf. Process. Lett. **21**(4), 181–185 (1985)

[11] $\alpha^{\mathcal{A}}_{\iota,c}$ returns abstract invariants maps, i.e., $\alpha^{\mathcal{A}}_{\iota,c} \in \wp(\Sigma_P) \to (\mathbb{L}_P \to (\mathrm{Var} \to \mathcal{A}))$.

3. Assaf, M., Naumann, D.A., Signoles, J., Totel, E., Tronel, F.: Hypercollecting semantics and its application to static analysis of information flow. In: Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, 18–20 January 2017, pp. 874–887 (2017). http://dl.acm.org/citation.cfm?id=3009889
4. Clarkson, M.R., Finkbeiner, B., Koleini, M., Micinski, K.K., Rabe, M.N., Sánchez, C.: Temporal logics for hyperproperties. In: Proceedings of the 3rd Conference on Principles of Security and Trust (POST 2014) (2014)
5. Clarkson, M.R., Schneider, F.B.: Hyperproperties. J. Comput. Secur. **18**(6), 1157–1210 (2010). http://dl.acm.org/citation.cfm?id=1891823.1891830
6. Cousot, P.: Abstract interpretation. ACM Comput. Surv. **28**(2), 324–328 (1996). http://doi.acm.org/10.1145/234528.234740
7. Cousot, P.: Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. Theor. Comput. Sci. **277**(1–2), 47–103 (2002)
8. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, POPL 1977, pp. 238–252. ACM, New York (1977)
9. Cousot, P., Cousot, R.: Systematic design of program analysis frameworks. In: Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, POPL 1979, NY, USA, pp. 269–282 (1979). http://doi.acm.org/10.1145/567752.567778
10. Cousot, P., Cousot, R.: Abstract interpretation frameworks. J. Log. Comput. **2**(4), 511–547 (1992). http://dx.doi.org/10.1093/logcom/2.4.511
11. Cousot, P., Cousot, R.: A case study in abstract interpretation based program transformation. Electron. Notes Theor. Comput. Sci. **45**, 41–64 (2001). http://www.sciencedirect.com/science/article/pii/S157106610480954X
12. Cousot, P., Cousot, R.: Systematic design of program transformation frameworks by abstract interpretation. In: Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2002, NY, USA, pp. 178–190. ACM, New York (2002)
13. Cousot, P., Cousot, R.: An abstract interpretation framework for termination. In: Conference Record of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Philadelphia, PA, pp. 245–258. ACM, New York, January 2012
14. Cousot, R., Cousot, P.: Constructive versions of Tarski's fixed point theorems. Pac. J. Math. **82**(1), 43–57 (1979)
15. Finkbeiner, B., Rabe, M.N., Sánchez, C.: Algorithms for model checking HyperLTL and HyperCTL*. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9206, pp. 30–48. Springer, Cham (2015). doi:10.1007/978-3-319-21690-4_3
16. Giacobazzi, R., Logozzo, F., Ranzato, F.: Analyzing program analyses. In: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, 15–17 January 2015, pp. 261–273 (2015)
17. Giacobazzi, R., Mastroeni, I.: Non-standard semantics for program slicing. High. Order Symbolic Comput. **16**(4), 297–339 (2003)
18. Giacobazzi, R., Mastroeni, I.: Transforming semantics by abstract interpretation. Theor. Comput. Sci. **337**(1–3), 1–50 (2005)
19. Mastroeni, I., Giacobazzi, I.: An abstract interpretation-based model for safety semantics. Int. J. Comput. Math. **88**(4), 665–694 (2011)

20. Miné, A.: Backward under-approximations in numeric abstract domains to automatically infer sufficient program conditions. Sci. Comput. Program. **93**(Part B), 154–182 (2014). Special Issue on Invariant Generation
21. Weiser, M.: Program slicing. IEEE Trans. Softw. Eng. **10**(4), 352–357 (1984)
22. Zdancewic, S., Myers, A.C.: Observational determinism for concurrent program security. In: Proceedings of IEEE Computer Security Foundations Workshop, Pacific Grove, CA, pp. 29–43, June 2003