International Journal of Computer Mathematics
Vol. 00, No. 00, Month 200x, 1–29

RESEARCH ARTICLE

An Abstract Interpretation-based Model for Safety Semantics

Isabella Mastroeni (Corresponding author) and Roberto Giacobazzi
Dipartimento di Informatica
Università di Verona
Strada Le Grazie, 37134 Verona (Italy)
isabella.mastroeni@univr.it and *roberto.giacobazzi@univr.it*
Phone: +39 045 802 7089/7995, Fax: +39 045 802 7928
(*Received 00 Month 200x; in final form 00 Month 200x*)

In this paper we describe safety semantics as abstract interpretation of a trace-based operational semantics of a transition system. Intuitively, a property is safety if "nothing bad will happen". Formally this is described by saying that a property is safety if it is maximal with respect to a given set of allowed partial executions. We show that this can be specified in the standard Cousot's framework of abstract interpretation. In particular, we show that this semantics can be derived as fixpoint of a semantic operator. This construction provides a formal characterization of the constructive nature of safety properties, that can be enforced by means of execution monitors. By using the same construction we show that while safety without stuttering preserves the constructive nature, safety properties allowing cancellation of states lose the constructive characterization. Finally, we characterize safety properties as the closed elements of a closure, and we show that in the abstract interpretation framework safety and liveness properties lose their complementary nature.

Keywords: Abstract interpretation, safety, semantics, program verification, closure operators.

1. Introduction

The traditional dualism between safety and liveness properties of a transition system has been widely studied in the literature. Since Lamport's seminal paper [28], a number of authors have studied the computational [4, 32], logical [8], algebraic [4], and topological [3, 6] aspects of safety and liveness properties of a computation. This dualism have been also studied in the framework of model checking and temporal logic [37, 38, 40] where safety is also known as *invariance*, saying that each partial computation of a possibly infinite trace meets some requirement. According to this intuitive definition, safety properties assert that "nothing bad happens"; whereas liveness properties ensure that "something good will eventually happen". Typical examples of safety properties are deadlock freedom, mutual exclusion, and partial correctness. In contrast, a typical liveness property is termination.
The importance of safety properties relies precisely on their standard constructive characterization. Indeed, Schneider [35] noted that safety properties correspond precisely to the enforceable properties. Namely, to those properties for which there exists a mechanism that works by monitoring execution steps of a program, terminating the programs that are about to violate the security property. The basic idea is that a safety property holds for a computation if it holds for each of its states, therefore by checking the property during the execution we are sure to enforce the property for the whole computation. Starting from this work, several papers have been written about execution monitors, analysing their power, in terms of the

ISSN: 0020-7160 print/ISSN 1029-0265 online

DOI: 10.1080/0020716YYxxxxxxxx
http://www.informaworld.com

information that they can recall [20], or trying to extend the class of properties that can be monitored [30]. Recently, a precise characterization of enforceable security properties has been given [26], providing a better characterization of those properties which are enforceable by execution monitors as well as a taxonomy of enforceable security policies.
A more theoretical aspect to consider is that the standard characterization of safety/liveness properties naturally leads also to the definition of safety properties as closure operators on the set of possible traces, and liveness as open sets. This corresponds to a well-known approach to safety/liveness in topological terms. According to Alpern and Schneider [3] safety properties are the closed sets in the Cantor's topology on infinite traces, while liveness properties are precisely the dense sets of the same topology. This dualism is justified by observing that with respect to liveness properties, any partial computation is always remediable. This corresponds to saying that for any finite (partial) trace σ, there exists an infinite completion $\sigma\eta$ of σ such that $\sigma\eta$ satisfies a given liveness property. Another theoretical approach for modelling safety and liveness is the one proposed by H.P. Gumm in [25]. In this work the author shows that all that is needed in order to characterize safety is a $\vee$-preserving map φ between complete Boolean algebras. This map extracts from a set of infinite traces all the corresponding partial executions and it can be interpreted as an abstraction of the infinite semantics, in the standard abstract interpretation framework [11]. This map is central in our approach since it provides the model for safety semantics necessary for establishing a formal connection between the standard approaches to safety and liveness and abstract interpretation.

Abstract Interpretation and the hierarchy of semantics. Abstract interpretation [11] is a general theory for semantics approximation, which includes static program analysis as a special case. The design of an approximate semantics is usually a step-by-step procedure which starts from a very concrete semantics, specifying the computational behavior at a great level of detail, and which leads to the definition of a more abstract semantics, where only the properties of interest about the computation can be observed. The abstraction is specified by an $\vee$-preserving map which represents the left adjoint in a pair of functions, relating the concrete and the abstract semantics, forming a Galois insertion. In the case of standard program analysis, the approximate semantics is a decidable approximation of the concrete one. The whole approach is systematically driven by abstract interpretation theory which provides a number of formal methods and tools to help the designer. This approach has several well-known advantages with respect to other methods: (1) The analysis is fully described and constructively derived by the way the concrete data and control flows are approximated; (2) The correctness with respect to the concrete semantics can be immediately proved formally by construction; (3) New and more advanced analyses can be systematically conceived by modifying the abstraction methods [13, 22].

Cousot [10] proposes an abstract interpretation-based formal structure where several well-known semantics are derived as abstract interpretations of a more concrete semantics, which is the maximal trace semantics. In Fig. 1 we have a picture of this hierarchy, in particular we can note that in the same structure we have also depicted several possible observables of the different semantics (e.g., finite, $+$, or infinite, ω, computations). All the abstraction relations depicted with plain lines (isomorphisms) or arrows (abstractions) are those present in the original hierarchy (see Sect. 2.3 for more details).

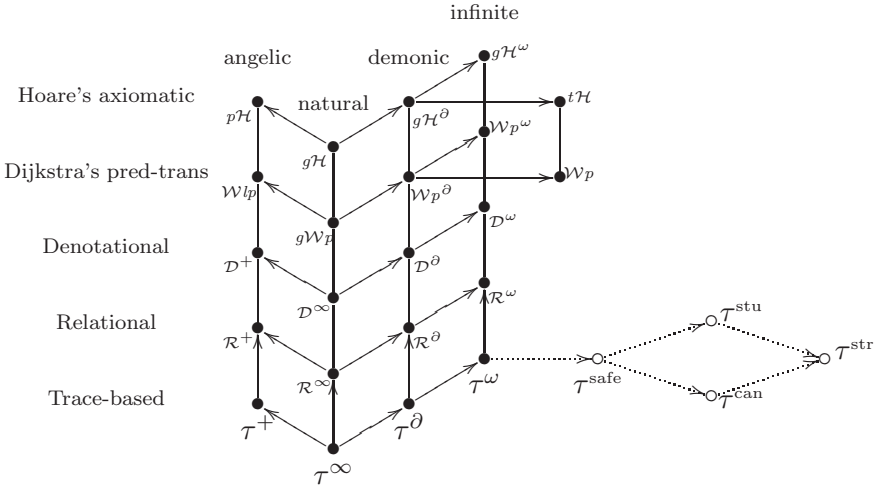

Figure 1. Cousot's hierarchy.

Main contribution. In this paper we use the abstract interpretation framework mainly for two reasons: first we want to insert safety semantics in the Cousot's hierarchy of semantics (Fig. 1); second, we aim to study whether the complementary relation between safety and liveness holds also in the abstract interpretation characterization. For the first task, the idea is that of deriving a semantics for safety by abstract interpretation, i.e., by abstracting the (infinite) operational trace-based semantics [10]. The derived semantics is the most abstract approximation of the concrete trace semantics of a transition system which preserves safe executions, i.e., modeling only safety properties. The interest in this semantics is twofold: (1) it provides a formal setting where safety semantics can be compared with respect to other semantics; (2) it provides a base semantics for designing static program analysis tools for safety properties, for proving their correctness, and for deriving new safety properties by abstract interpretation. In particular, we show that safety semantics can be obtained as the fixpoint of a semantic operator, which provides a formal characterization of how execution monitors works for enforcing safety properties. We use this characterization also for showing that not all the possible restrictions of safety properties preserve this constructive nature. In particular, safety without stuttering, allowing repetition of states, can still be obtained as fixpoint, while safety properties allowing cancellation of states (e.g., strong safety) lose the constructive nature, namely cannot be enforced like standard safety properties.

The second task concerns complementation in the abstract interpretation framework, hence we have first to characterize safety properties by means of a closure operator. We formally prove that this operator precisely captures safety properties in the Alpern-Schneider approach, modelling both safety and liveness properties. At this point we study the algebraic properties of the safety domain in order to compute its (pseudo-)complement in the infinite trace semantics, showing that, in the abstract interpretation framework, safety is not complemented, hence liveness cannot be characterized as the complement of safety.

Structure of the paper. The paper is structured as follows. In Sect. 2 we describe some basic notions that we will use in the paper. In particular we introduce abstract interpretation, and we describe the Cousot's hierarchy of semantics. In Sect. 3 we describe the safety semantics as Galois insertion, including it in the

hierarchy of semantics. The main task of this section is to use the Kleene fixpoint transfer theorem in order to obtain the safety semantics as fixpoint of a semantic operator, formalising its constructive nature. In Sect. 4 we introduce three restrictions of safety properties, we include them in the Cousot's hierarchy of semantics as abstract interpretations of the safety semantics. Hence we show that to allow repetition of states in safety properties preserves the constructive nature, while to allow cancellation of states makes safety properties lose the constructive characterization. Finally, in Sect. 5 we obtain safety semantics as an abstract domain and we characterize the algebraic structure of safety semantics in the abstract interpretation framework, in order to show that, in this context, liveness cannot be interpreted as the complement of safety semantics.

2. Preliminaries

2.1 *Basic notions*

If S and T are sets, then $\wp(S)$ denotes the powerset of S, $S \smallsetminus T$ 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 $g \circ f$ we denote the composition of the functions f and g, i.e., $g \circ f \stackrel{\text{def}}{=} \lambda x.g(f(x))$.

Lattices and meet-irreducible elements. The notation $\langle P, \leq\rangle$ denotes a poset P with ordering relation $\leq$, while $\langle P, \leq, \vee, \wedge, \top, \bot\rangle$ denotes a complete lattice P, with ordering $\leq$, *lub* $\vee$, *glb* $\wedge$, greatest element (top) $\top$, and least element (bottom) $\bot$. Often, $\leq_P$ 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.
$x \in C$ is *meet-irreducible* if $x = a \wedge b \;\Rightarrow\; x \in \{a, b\}$. The set of meet-irreducible elements in C is denoted $Mirr(C)$. A subset X of a lattice C is said to be *order generating* iff every element of C can be written as a glb of a subset of X.

Functions. $S \longrightarrow T$ 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 $x \in S$, $f(x) \leq_P g(x)$. 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 (nonempty) chains iff f preserves *lub*'s of directed subsets (co-continuity is dually defined), and $f : C \to A$ is (completely) additive if f preserves *lub*'s of all subsets of C (empty set included). Co-addittivity is defined by duality.

Fixpoints. We denote by $lfp^{\leq}_{\bot} f$ and $gfp^{\leq}_{\top} f$, respectively, the least and greatest fixpoint, when they exist, of an operator f on a poset. If $f \in C \xrightarrow{\text{c}} C$ then $lfp^{\leq_c}_{\bot_c} f = \vee_{i\in\mathbb{N}} f^i(\bot_C)$, where, for any $i \in \mathbb{N}$ and $x \in C$, the i-th 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 $gfp^{\leq_C}_{\top_C} f = \wedge_{i\in\mathbb{N}} f^i(\top_C)$. $\{f^i(\bot_C)\}_{i\in\mathbb{N}}$ and $\{f^i(\top_C)\}_{i\in\mathbb{N}}$ are called, respectively, the *upper* and *lower Kleene's iteration sequences of* f (see [12]). It is possible to transfer any fixpoint computation on a domain into another do-

main under suitable conditions. These results are known as *fixpoint transfer theorems* [10]. In the following we will use the *Kleene fixpoint transfer theorem* which is as follows: Let $\langle A, \leq_A \rangle$ and $\langle C, \leq_C \rangle$ be complete lattices and $f_C : C \xrightarrow{\mathrm{m}} C$, $f_A : A \xrightarrow{\mathrm{m}} A$, and $\alpha : C \xrightarrow{\mathrm{c}} A$ such that $\alpha(\bot_C) = \bot_A$ and $\alpha \circ f_C = f_A \circ \alpha$. Then $\alpha(lfp^{\leq_c}_{\bot_c} f_C) = lfp^{\leq_A}_{\bot_A} f_A$. The *closure iteration order* for $lfp f$ ($gfp f$) is the least ordinal β such that $f(f^\beta) = f^\beta$.

Topology. A *topology* on a set X, ΩX, is a family of subsets of X such that: If $S \subseteq \Omega X$ then $\bigcup S \in \Omega X$; If $S \subseteq \Omega X$ is finite then $\bigcap S \in \Omega X$. X is a *topological space* if it is equipped with a topology. The elements of ΩX are known as the *open* subsets of the space X. We say that a subset $F \subseteq X$ is *closed* if its complement is open. Let X be a topological space, then a *(Kuratowski) topological closure* is an operator $M : \wp(X) \longrightarrow \wp(X)$ which is extensive ($\forall A \subseteq X.\ A \subseteq M(A)$), idempotent and finitely additive (namely $M(\varnothing) = \varnothing$ and $M(A) \cup M(B) = M(A \cup B)$).

2.2 *Abstract interpretation*

Abstract interpretation is a general theory for specifying and designing approximate semantics of program languages [11].

Abstract domains individually. Approximation can be equivalently formulated either in terms of Galois connections or closure operators [11, 13].
A Galois connection is an adjoint relation between abstraction and concretization functions [11]. The abstraction identifies only some properties of interest, while the concretization associates with the abstract property the greatest set of concrete elements having the same abstract property. Consider for example the concrete domain of sets of integer values $\wp(\mathbb{Z})$, and suppose to consider the sign property, namely the sign abstract domain $\mathcal{S}$. Then a possible abstract domain is $\mathcal{S} = \{$ *"I don't know"*, $+, 0, -$, *"none"*$\}$ representing the possible sign of sets of integers, and the corresponding abstraction is $\alpha_{\mathcal{S}} : \wp(\mathbb{Z}) \to \mathcal{S}$ such that $\alpha_{\mathcal{S}}(\varnothing) =$ *"none"*, $\alpha_{\mathcal{S}}(X) = +$ if $\forall n \in X.\ n > 0$, $\alpha_{\mathcal{S}}(0) = 0$, $\alpha_{\mathcal{S}}(X) = -$ if $\forall n \in X.\ n < 0$ and $\alpha_{\mathcal{S}}(X) =$ *"I don't know"* otherwise. The corresponding concretization function maps each abstract value in the set of all the integers with the represented property: $\gamma_{\mathcal{S}} : \mathcal{S} \to \wp(\mathbb{Z})$ such that $\gamma_{\mathcal{S}}(\text{“none”}) = \varnothing$, $\gamma_{\mathcal{S}}(+) = \left\{ n \in \mathbb{Z} \,\middle|\, n > 0 \right\}$, $\gamma_{\mathcal{S}}(0) = \{0\}$, $\gamma_{\mathcal{S}}(-) = \left\{ n \in \mathbb{Z} \,\middle|\, n < 0 \right\}$ and $\gamma_{\mathcal{S}}(\text{“I don't know”}) = \mathbb{Z}$. At this point let us introduce the adjoint framework formally. If $\alpha : C \xrightarrow{\mathrm{m}} A$ and $\gamma : A \xrightarrow{\mathrm{m}} C$ 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 $\langle C, \leq_C \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq_A \rangle$. Note that in GC, for any $x \in C$ and $y \in A$: $\alpha(x) \leq_A y \Leftrightarrow x \leq_C \gamma(y)$ and $\gamma(y) = \bigvee \left\{ x \,\middle|\, \alpha(x) \leq y \right\}$ and $\alpha(x) = \bigwedge \left\{ y \,\middle|\, x \leq \gamma(y) \right\}$. If in addition $\alpha \circ \gamma = \lambda x.x$, then (A, α, γ, C) is a *Galois insertion* (GI) also denoted $\langle C, \leq_C \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq_A \rangle$ of A in C. Note that $A \cong C$ iff $\langle C, \leq_C \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq_A \rangle$. The concrete and abstract domains, C and A, are assumed to be complete lattices. 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, γ being 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.

The standard abstract interpretation framework can be also represented by means of upper closure operators [13]. An *upper closure operator* (uco for short) $\rho : C \to C$ on a poset C, representing concrete objects, is monotone, idempotent, and extensive: $\forall x \in C.\ x \leq_C \rho(x)$. The upper closure operator is the function that maps the concrete values to their abstract properties, namely with the best possible approximation of the concrete value in the abstract domain. More precisely, this means that the closure maps each concrete element in the greatest concrete element satisfying the same abstract property. For example, the operator for the property of signs is $\mathtt{Sign} : \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$, operating on the powerset of integers and associating each set of integers with the set of all the integers with the same sign, for instance $\mathtt{Sign}(S) = \left\{\ n \in \mathbb{Z} \,\middle|\, n > 0\ \right\}$ if $\forall n \in S.\, n > 0$. Let $\langle C, \leq, \vee, \wedge, \top, \bot\rangle$ be a complete lattice, then closure operators ρ are uniquely determined by the set of their fixpoints $\rho(C)$, for instance $\mathtt{Sign} = \{\mathbb{Z}, \mathbb{Z}^+, \{0\}, \mathbb{Z}^-, \varnothing\}$[1]. For upper closures, $X \subseteq C$ is the set of fixpoints of $\rho \in \mathtt{uco}(C)$ iff X is a *Moore-family* of C, i.e., $X = \mathcal{M}(X) \stackrel{\text{def}}{=} \{\wedge S \mid S \subseteq X\}$ — where $\wedge\varnothing = \top \in \mathcal{M}(X)$.

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

Note that any GI (A, α, γ, C) uniquely determines an upper closure operator $\gamma\circ\alpha \in uco(C)$ 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, Sect. 7] and [13, Sect. 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 with regard to 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 $\langle A_1, \leq_{A_1}\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A_2, \leq_{A_2}\rangle$. Let $\{A_i\}_{i\in I} \subseteq uco(C)$: $\sqcup_{i\in I}A_i$ is the most concrete among the domains in $\mathfrak{L}_C$ which are abstractions 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 among the domains in $\mathfrak{L}_C$ which are 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., $\sqcap_{i\in I}A_i = \mathcal{M}(\cup_{i\in I}A_i)$.

Computing abstract functions. If (A, α, γ, C) is a GI and $f_C : C \xrightarrow{c} C$, $f_A : A \xrightarrow{c} A$, then f_A is a *sound* approximation of f_C if $\alpha \circ f_C \leq_A f_A \circ \alpha$. Soundness naturally implies that $\alpha(lfp^{\leq_C}_{\bot_C} f_C) \leq_A lfp^{\leq_A}_{\bot_A} f_A$. If $\alpha \circ f_C = f_A \circ \alpha$ then we say that f_A is a *complete* approximation of f_C. In the case of completeness we have $\alpha(lfp^{\leq_C}_{\bot_C} f_C) = lfp^{\leq_A}_{\bot_A} f_A$ [23].

[1]Note that $\mathbb{Z}^+ \stackrel{\text{def}}{=} \left\{\ n \in \mathbb{Z} \,\middle|\, n > 0\ \right\}$ and $\mathbb{Z}^-$ is analogously defined.

Semantics	*Domain relation*	*Abstraction and Concretization*
$\tau^+ = \alpha^+(\tau^\infty)$	$\langle \wp(\Sigma^\infty), \subseteq\rangle \underset{\alpha^+}{\overset{\gamma^+}{\leftrightarrows}} \langle \wp(\Sigma^+), \subseteq\rangle$	$\alpha^+(X) = X \cap \Sigma^+ \stackrel{\text{def}}{=} X^+$ $\gamma^+(Y) = Y \cup \Sigma^\omega$
$\tau^\partial = \alpha^\partial(\tau^\infty)$	$\langle \wp(\Sigma^\infty), \subseteq\rangle \underset{\alpha^\partial}{\overset{\gamma^\partial}{\leftrightarrows}} \langle D^\partial, \subseteq\rangle$	$\alpha^\partial(X) = X \cup \bigcup \{\ \mathit{chaos}(\sigma_0) \,\big|\, \sigma \in X \cap \Sigma^\omega\ \}$ $\gamma^\partial(Y) = Y$
$\tau^\omega = \alpha^\omega(\tau^\infty)$	$\langle \wp(\Sigma^\infty), \subseteq\rangle \underset{\alpha^\omega}{\overset{\gamma^\omega}{\leftrightarrows}} \langle \wp(\Sigma^\omega), \subseteq\rangle$	$\alpha^\omega(X) = X \cap \Sigma^\omega \stackrel{\text{def}}{=} X^\omega$ $\gamma^\omega = X \cup \Sigma^+$

Table 1. Observable semantics as abstract interpretations

2.3 *Cousot's semantics hierarchy*

In this section, we recall Cousot's hierarchy of semantics [10, 14]. 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 nonempty 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 finite nonempty and infinite sequences of symbols in Σ. Given a sequence $\sigma \in \Sigma^\infty \stackrel{\text{def}}{=} \Sigma^+ \cup \Sigma^\omega$, its length is denoted $|\sigma| \in \mathbb{N} \cup \{\omega\}$ and its i-th element is denoted σ_i. 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 $i < |\sigma|$: $\langle \sigma_i, \sigma_{i+1}\rangle \in \tau$. In the following we will use Greek letters for denoting potentially infinite traces, we will use letters such as x, y for denoting finite traces of states. The *maximal trace semantics* of a transition system [14] is $\tau^\infty \stackrel{\text{def}}{=} \tau^+ \cup \tau^\omega$, where if $T \subseteq \Sigma$ is a set of final/blocking states $\tau^{\dot n} = \{\sigma \in \Sigma^+ |\ |\sigma| = n, \forall i \in [1, n)\,.\,\langle \sigma_{i-1}, \sigma_i\rangle \in \tau\}$, $\tau^\omega = \{\sigma \in \Sigma^\omega |\ \forall i \in \mathbb{N}\,.\,\langle \sigma_i, \sigma_{i+1}\rangle \in \tau\}$, $\tau^+ = \cup_{n>0}\{x \in \tau^{\dot n} |\ x_{n-1} \in T\}$, 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 $0 \leq l < |\eta|$, while $\sigma_{|\eta|-1+n} = \xi_n$ if $0 \leq n < |\xi|$. Moreover if $\eta \in \Sigma^\omega$ then for each $\xi \in \Sigma^\infty$ we have $\eta^\frown\xi = \eta$. For instance, if $\eta = ab$ and $\xi = bc$, then $\sigma = \eta^\frown\xi = abc$.

The semantics τ^∞ [14] is the fixpoint 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 fixpoint in a unique fixpoint presentation: finite (terminating) traces are obtained by induction (*least fixpoint*) of F^∞ on $\langle \wp(\Sigma^+), \subseteq\rangle$ and infinite traces are obtained by co-induction (*greatest fixpoint*) on $\langle \wp(\Sigma^\omega), \subseteq\rangle$, which corresponds to the *least fixpoint* of F^∞ on $\langle \wp(\Sigma^\omega), \supseteq\rangle$. In this case: $\tau^\infty = \mathit{lfp}^{\sqsubseteq^\infty}_{\Sigma^\omega} F^\infty$ (see [10, 14] for details).

The semantics in natural style may have a corresponding *angelic*, *demonic*, and *infinite* observable all of which are abstractions. All the observables are derived as fixpoints in the computational order by applying fixpoint transfer theorems.

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 sets of finite traces only, i.e., $\tau^+ = \alpha^+(\tau^\infty)$ (see Table 1). The angelic trace semantics is constructively derived as fixpoint in the computational order: $\tau^+ = lfp^{\subseteq}_{\varnothing} F^+$ where $F^+ : \wp(\Sigma^+) \to \wp(\Sigma^+)$ is defined as $F^+(X) = \tau^1 \cup \tau^{\dot{2}} \frown X$.

Demonic. The demonic trace semantics, denoted as τ^∂, 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 and this corresponds to allowing the worst possible behavior of the program [10, 16]. This semantics is obtained as an abstraction of the natural semantics by the function α^∂, i.e. $\tau^\partial = \alpha^\partial(\tau^\infty)$ (see Table 1). In this way the new observable is defined on the domain $D^\partial = \alpha^\partial(\wp(\Sigma^\infty))$[1] that is such that $X \in D^\partial$ if and only if

$$\sigma \in X^\omega \;\Rightarrow\; chaos(\sigma) \subseteq X^+$$

where $chaos(\sigma) \stackrel{\text{def}}{=} \{\, \delta \in \Sigma^+ \,|\, \delta_0 = \sigma_0 \,\}$. The demonic trace semantics is constructively derived as fixpoint in the computational order: $\tau^\partial = lfp^{\sqsubseteq^\partial}_{\Sigma^\infty} F^\partial$ where $X \sqsubseteq^\partial Y$ iff $\forall \sigma \in \Sigma^\omega.\, \sigma \in X \;\vee\; (\sigma \notin Y \;\wedge\; \forall \delta \in \Sigma^+.\, \sigma_0\delta \in X \;\Rightarrow\; \sigma_0\delta \in Y)$ and $F^\partial : D^\partial \to D^\partial$ is defined as $F^\partial(X) = \tau^1 \cup \tau^{\dot{2}} \frown X$ [10].

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 1). The infinite trace semantics is constructively derived as fixpoint in the computational order: $\tau^\omega = gfp^{\subseteq}_{\Sigma^\omega} F^\omega$ where $F^\omega : \wp(\Sigma^\omega) \to \wp(\Sigma^\omega)$ is defined as $F^\omega(X) = \tau^{\dot{2}} \frown X$.

Example 2.1 In this example, we show how the observable abstractions work. Consider $\tau = \{\langle a, a\rangle, \langle b, c\rangle, \langle a, b\rangle, \langle c, d\rangle, \langle c, e\rangle\}$ with d and e final states. Then $\tau^\infty = \{a^\omega\} \cup \{\, a^n bcd, a^n bce \,|\, n \in \mathbb{N} \,\} \cup \{cd, d, ce, e\}$. At this point, $\tau^\omega = \{a^\omega\}$, $\tau^+ = \{\, a^n bcd, a^n bce \,|\, n \in \mathbb{N} \,\} \cup \{cd, d, ce, e\}$ and $\tau^\partial = \{\, \sigma \in \Sigma^+ \,|\, \sigma_0 = a \,\} \cup \{\, a^n bcd, a^n bce \,|\, n \in \mathbb{N} \,\} \cup \{cd, d, ce, e\} = \{\, \sigma \in \Sigma^+ \,|\, \sigma_0 = a \,\} \cup \{bcd, bce, cd, d, ce, e\}$.

All semantics in the hierarchy are derived again as abstract interpretation of the trace-based semantics. Each semantics in natural style corresponds here to a suitable abstraction of the basic natural trace-based semantics τ^∞.

The *relational semantics* $\mathcal{R}^\infty$ associates an input-output relation with program traces 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 2. The relative observables are angelic $\mathcal{R}^+$ (the big-step relational semantics [34]), demonic $\mathcal{R}^\partial$ and infinite $\mathcal{R}^\omega$ relational.

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 leads to the denotational semantics by abstracting the relational one, i.e., $\mathcal{D}^\infty = \alpha^{\mathcal{D}}(\mathcal{R}^\infty)$ is given in Table 2. The relative observables are angelic $\mathcal{D}^+$, demonic $\mathcal{D}^\partial$ [5] and infinite $\mathcal{D}^\omega$ denotational.

[1]Note that, as explained in Sect. 2.2, in order to obtain a Galois insertion the abstraction has to be surjective and therefore, in this case, we have to restrict the co-domain of α^∂ precisely to the set of its images.

Semantics	*Domain relation*	*Abstraction and Concretization*
$\mathcal{R}^\infty = \alpha^{\mathcal{R}}(\tau^\infty)$	$\langle \wp(\Sigma^\infty), \subseteq\rangle \underset{\alpha^{\mathcal{R}}}{\overset{\gamma^{\mathcal{R}}}{\leftrightarrows}} \langle \wp(\Sigma \times \Sigma_\perp), \subseteq\rangle$	$\alpha^{\mathcal{R}}(X) = \{\langle x_0, x_{n-1}\rangle \mid x \in X^+\}$ $\cup\{\langle \sigma_0, \perp\rangle \mid \sigma \in X^\omega\}$ $\gamma^{\mathcal{R}}(Y) = \{x \in \Sigma^+ \mid \langle x_0, x_{n-1}\rangle \in Y\}$ $\cup\{\sigma \in \Sigma^\omega \mid \langle \sigma_0, \perp\rangle \in Y\}$
$\mathcal{D}^\infty = \alpha^{\mathcal{D}}(\mathcal{R}^\infty)$	$\langle \wp(\Sigma \times \Sigma_\perp), \subseteq\rangle \underset{\alpha^{\mathcal{D}}}{\overset{\gamma^{\mathcal{D}}}{\leftrightarrows}} \langle \Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq\rangle$	$\alpha^{\mathcal{D}}(X) = \lambda s.\{s' \in \Sigma_\perp \mid \langle s, s'\rangle \in X\}$ $\gamma^{\mathcal{D}}(f) = \{\langle x, y\rangle \mid y \in f(x)\}$
$g\mathcal{W}p = \alpha^{g\mathcal{W}p}(\mathcal{D}^\infty)$	$\langle \Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq\rangle \underset{\alpha^{g\mathcal{W}p}}{\overset{\gamma^{g\mathcal{W}p}}{\leftrightarrows}} \langle \wp(\Sigma_\perp) \xrightarrow{\text{coa}} \wp(\Sigma), \sqsupseteq\rangle$	$\alpha^{g\mathcal{W}p}(f) = \lambda P.\{s \in \Sigma \mid f(s) \subseteq P\}$ $\gamma^{g\mathcal{W}p}(\Phi) = \lambda s.\{s' \mid s \notin \Phi(\Sigma_\perp \smallsetminus \{s'\})\}$
$g\mathcal{H} = \alpha^{g\mathcal{H}}(g\mathcal{W}p)$	$\langle \wp(\Sigma_\perp) \xrightarrow{\text{coa}} \wp(\Sigma), \sqsupseteq\rangle \underset{\alpha^{g\mathcal{H}}}{\overset{\gamma^{g\mathcal{H}}}{\leftrightarrows}} \langle \wp(\Sigma) \otimes \wp(\Sigma_\perp), \supseteq\rangle$	$\alpha^{g\mathcal{H}}(\Phi) = \{\langle X, Y\rangle \mid X \subseteq \Phi(Y)\}$ $\gamma^{g\mathcal{H}}(H) = \lambda Y. \cup \{X \mid \langle X, Y\rangle \in H\}$

Table 2. Basic natural-style semantics as abstract interpretations

Dijkstra's predicate transformer $g\mathcal{W}p$ is represented as a set of co-additive functions, denoting the weakest-precondition predicate transformers [17]. In general, the weakest precondition semantics describes in an implicit way the semantics of a program. 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 *pre-condition*, namely the biggest set of possible initial states, which allows the program to terminate in a state which belongs to P. 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 2. The relative observables are angelic $\mathcal{W}lp$ (weakest-liberal precondition [16]), demonic $\mathcal{W}p^\partial$, infinite $\mathcal{W}p^\omega$ and weakest precondition for total correctness $\mathcal{W}p$ [15].

Similarly 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 [27]. Hoare's axiomatic semantics $g\mathcal{H}$ is represented as elements in tensor product domains, i.e., GC's, specifying the adjoint relation between weakest-precondition and strongest-postcondition in Hoare's triples $\{P\}\ C\ \{Q\}$. The abstraction function $\alpha^{g\mathcal{H}}$ that leads to the axiomatic semantics by abstracting the weakest precondition one, i.e., $g\mathcal{H} = \alpha^{g\mathcal{H}}(g\mathcal{W}p)$, is given in Table 2. The relative observables are angelic $p\mathcal{H}$ (Hoare's partial correctness semantics [27]), demonic $g\mathcal{H}^\partial$, infinite $g\mathcal{H}^\omega$ and total correctness semantics $t\mathcal{H}$ [27].

Example 2.2 Consider the transition system of Example 2.1. In this case we have $\mathcal{R}^\infty = \{\langle a, \perp\rangle, \langle a, d\rangle, \langle a, e\rangle, \langle b, d\rangle, \langle b, e\rangle, \langle c, d\rangle, \langle c, e\rangle, \langle d, d\rangle, \langle e, e\rangle\}$. $\mathcal{D}^\infty = \lambda s.\ X_s$ where $X_a = \{\perp, d, e\}$, $X_b = \{d, e\}$, $X_c = \{d, e\}, X_d = \{d\}$ and $X_e = \{e\}$. $g\mathcal{W}p = \lambda S.\ Y_S$ where $Y_{\{\perp\}} = \{a\}$, $Y_{\{d\}} = \{a, b, c, d\}$ and $Y_{\{e\}} = \{a, b, c, e\}$. Finally $g\mathcal{H}$ contains, for example, tuples of the kind $\langle\{a\}, \{\perp\}\rangle$ or $\langle\{a, b\}, \{d, \perp\}\rangle$.

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

abstractions (i.e., abstractions which are not isomorphisms) between semantics.

3. Safety semantics in the hierarchy

In this section we aim to characterize the safety semantics in the abstract interpretation framework in order to insert it in the Cousot's hierarchy of semantics and to formally characterize its constructive nature. In fact, as we have seen, all the semantics in the hierarchy are obtained as fixpoints of semantic operators. Our aim is to provide the same characterization also for safety semantics, showing that this fixpoint characterization precisely formalises the constructive nature of safety properties, which can be enforced by means of execution monitors.

Modelling safety in abstract interpretation. The abstract interpretation formalization of safety properties is given in terms of an abstraction of a set of infinite traces of a transition system modelling concurrent executions. The first definition of safety by means of a pair of adjoint functions was given in terms of the maps $\varphi_\omega : \wp(\Sigma^\omega) \to \wp(\Sigma^+)$ and $\gamma_\omega : \wp(\Sigma^+) \to \wp(\Sigma^\omega)$ [25] where:

$$\varphi_\omega(X) = \left\{ x \in \Sigma^+ \,\middle|\, \exists \delta \in X \,.\, x \preccurlyeq \delta \right\} \qquad \gamma_\omega(Y) = \left\{ \sigma \in \Sigma^\omega \,\middle|\, \varphi_\omega(\sigma) \subseteq Y \right\}$$

with $X \in \wp(\Sigma^\omega)$, $Y \in \wp(\Sigma^+)$. The relation $x \preccurlyeq y$ means that x is a prefix of y. In this case, while φ_ω extracts the set of finite prefixes of an (infinite) trace, γ_ω completes a set Y of finite traces into the least set of infinite traces whose prefixes are included in Y. The result is a Galois connection.

PROPOSITION 3.1 *$\langle \wp(\Sigma^+), \varphi_\omega, \gamma_\omega, \wp(\Sigma^\omega) \rangle$ is a Galois connection [25].*

At this point, we can define the safety domain, as the φ_ω abstraction of the infinite trace domain, namely

$$\mathcal{S} = \varphi_\omega(\wp(\Sigma^\omega)) = \left\{ X \in \wp(\Sigma^+) \,\middle|\, \exists Y \in \wp(\Sigma^\omega) \,.\, \varphi_\omega(Y) = X \right\}$$

This is a domain of infinite sets of finite traces, collecting all the sets of traces corresponding to safety properties. Note that, this domain, is closed under set union but not under set intersection, since the intersection of infinite sets can be finite.

PROPOSITION 3.2 *$\langle \mathcal{S}, \subseteq, \cup, \sqcap^{safe}, \Sigma^+, \varnothing \rangle$ is a complete lattice, where the greatest lower bound is the best correct approximation of the concrete one $\cap$, i.e., $\sqcap_i^{safe} X_i = \varphi_\omega(\cap_i \gamma_\omega(X_i))$.*

Hence, we can insert safety semantics in the hierarchy as shown in Fig. 1. by defining the safety semantics of a transition system $\langle \Sigma, \tau \rangle$ as $\tau^{\text{safe}} = \varphi_\omega(\tau^\omega)$.

Example 3.3 Consider the set of traces $X = \left\{ a^n c^\omega \,\middle|\, n \in \mathbb{N} \right\}$, then $\varphi_\omega(X) = \left\{ a^i c^j \,\middle|\, i, j \in \mathbb{N} \right\}$ takes all the finite prefixes of traces in X while $\gamma_\omega(\varphi_\omega(X)) = X \cup \{a^\omega\}$, namely γ_ω adds all the infinite traces whose prefixes are all in $\varphi_\omega(X)$, in this case a^ω.

Constructing safety by fixpoint. At this point, we aim to exploit the hierarchy of semantics in order to prove that also the safety semantics can be obtained as

the fixpoint of a semantic operator. This fixpoint characterization is important both in the security policies and in the semantic contexts since it provides a better understanding of the structure of the safety semantics. In the context of security policies, this construction provides, in some sense, a theoretical comprehension of why safety properties are enforceable by execution monitors. Indeed, execution monitors analyze the property step by step during the execution of programs, while the fixpoint operator we are going to define builds the safety semantics by keeping, at each step of computation, only the prefixes of those traces that at least for n steps (at the n^{th} iteration) are possible executions of the program to analyse. This is exactly the *constructive* characterization we can provide of safety semantics, in the semantic context, coherent with the constructive semantic characterization provided for several known semantics in the Cousot's hierarchy [10].

Hence, we follow the standard Cousot's construction by specifying safety semantics τ^{safe} as the fixpoint of a monotone operator defined on infinite traces. In particular, we show that this semantic operator is $\varphi_\omega(F^\omega)$, where we recall that the fixpoint of $F^\omega \stackrel{\text{def}}{=} \lambda X.\, \tau^{\dot{2}} \frown X$ is the infinite semantics τ^ω [10]. Note that, in the following, we use the function φ_ω applied also to sets of finite traces. This is a natural extension of the function previously defined: Let $X \in \Sigma^\infty$ then $\varphi_\omega(X) \stackrel{\text{def}}{=} \left\{\, y \in \Sigma^+ \,\middle|\, \exists x \in X \,.\, y \preccurlyeq x \,\right\}$. In order to specify safety semantics as fixpoints, we consider the semantic operator:

$$F^{\text{safe}}(X) = \varphi_\omega(\tau^{\dot{2}} \frown X)$$

The idea is to prove that the safety semantics is the fixpoint of this semantic operator by using the dual Kleene transfer theorem [10]. Consider a concrete domain C with an operation F, an abstract domain A with an abstract operator F_A, $\alpha : C \longrightarrow A$ co-continuous and $F_A \circ \alpha = \alpha \circ F$ (commutative property), then the transfer theorem says that $\alpha(gfpF) = gfpF_A$. In our case, the concrete domain is the infinite semantics τ^ω, the abstract domain is the safety semantics τ^{safe}, the abstraction is clearly the prefix abstraction φ_ω, while the concrete and the abstract operators are respectively F^ω and F^{safe}. Hence, in order to apply this transfer to greatest fixpoints the abstraction function has to be co-continuous but we know by Prop. 5.1 that φ_ω is not co-continuous. Fortunately, this is not a problem because Cousot noticed [10] that co-continuity is not needed in general, since the proof of the transfer theorem uses only the fact that the abstraction preserves the greatest lower bound of the (possibly transfinite) iterates of the concrete operator starting from $\top$. Therefore, the first thing to prove is that φ_ω preserves the greatest lower bound of all the iterates of F^ω. Fortunately, as the following results shows, F^{safe} is co-additive, hence we have only to check whether φ_ω preserves the greatest lower bound of the iterates, limited by ω, of the concrete operator starting from Σ^+. The following lemmas provide some useful properties of the concatenation operation. We recall that the concatenation used in this paper is not a simple juxtaposition of traces, but a concatenation possible only when the two traces share, respectively, the last and the first symbol, e.g., $ab \frown bc^\omega = abc^\omega$ while $ab \frown c^\omega = \varnothing$ (see Sect. 2.3).

LEMMA 3.4 *Let* $\{X_i\}_{i\in I} \subseteq \wp(\Sigma^\infty)$. *Then* $\tau^{\dot{2}} \frown (\bigcap_i X_i) = \bigcap_i (\tau^{\dot{2}} \frown X_i)$.

Proof

$$\begin{array}{rl}\delta \in \bigcap_i \tau^{\dot{2}} \frown X_i \Leftrightarrow & \forall i \,.\, \delta = \delta_0\delta_1\delta_2 \ldots \delta_n \ldots \in \tau^{\dot{2}} \frown X_i \\ \Leftrightarrow & \delta_0\delta_1 \in \tau^{\dot{2}},\ \forall i \,.\, \delta_1 \ldots \delta_n \ldots \in X_i \\ \Leftrightarrow & \delta_0\delta_1 \in \tau^{\dot{2}},\ \delta_1 \ldots \delta_n \ldots \in \bigcap_i X_i \\ \Leftrightarrow & \delta = \delta_0\delta_1\delta_2 \ldots \delta_n \ldots \in \tau^{\dot{2}} \frown \bigcap_i X_i\end{array}$$

■

In sake of readability, in the following of the paper we will use the notation $\gamma_\omega\varphi_\omega(X)$ instead of $\gamma_\omega(\varphi_\omega(X))$.

LEMMA 3.5 *Let $X \in \wp(\Sigma^\omega)$ and $Y \in \mathcal{S}$, then*

$$(i)\ \varphi_\omega(\tau^{\dot{2}} \frown \varphi_\omega(X)) = \varphi_\omega(\tau^{\dot{2}} \frown X) \qquad (ii)\ \tau^{\dot{2}} \frown \gamma_\omega(Y) = \gamma_\omega\varphi_\omega(\tau^{\dot{2}} \frown Y)$$

Proof (i) By definition $\tau^{\dot{2}} \frown \varphi_\omega(X) = \tau^{\dot{2}} \frown \left\{\ x \in \Sigma^+ \,\middle|\, \exists \sigma \in X \,.\, x \preccurlyeq \sigma\ \right\}$, then

$$\begin{array}{l} x' \in \varphi_\omega(\tau^{\dot{2}} \frown \left\{\ x \in \Sigma^+ \,\middle|\, \exists \sigma \in X \,.\, x \preccurlyeq \sigma\ \right\}) \\ \quad \Leftrightarrow\ x' \preccurlyeq x_0x_1x_2\cdots x_n \text{ with } x_0x_1 \in \tau^{\dot{2}},\ x_1x_2\cdots x_n \in \varphi_\omega(X) \\ \quad \Leftrightarrow\ x' \preccurlyeq x_0x_1\cdots x_n,\ x_0x_1 \in \tau^{\dot{2}},\ \exists \sigma \in X \,.\, x_1x_2\cdots x_n \preccurlyeq \sigma \\ \quad \Leftrightarrow\ x' \preccurlyeq x_0x_1x_2\cdots x_n \preccurlyeq \tau^{\dot{2}} \frown \sigma \in \tau^{\dot{2}} \frown X \\ \quad \Leftrightarrow\ x' \in \left\{\ x \in \Sigma^+ \,\middle|\, \exists \sigma \in \tau^{\dot{2}} \frown X \,.\, x \preccurlyeq \sigma\ \right\} = \varphi_\omega(\tau^{\dot{2}} \frown X)\end{array}$$

(ii) By definition $\tau^{\dot{2}} \frown \gamma_\omega(Y) = \tau^{\dot{2}} \frown \left\{\ \sigma \in \Sigma^\omega \,\middle|\, \varphi_\omega(\sigma) \subseteq Y\ \right\}$, then

$$\begin{array}{l} x' \in \tau^{\dot{2}} \frown \left\{\ \sigma \in \Sigma^\omega \,\middle|\, \varphi_\omega(\sigma) \subseteq Y\ \right\} \\ \quad \Leftrightarrow\ x' = x_0x_1x_2\cdots x_n\cdots \text{ with } x_0x_1 \in \tau^{\dot{2}},\ x_1x_2\cdots x_n\cdots \in \gamma_\omega(Y) \\ \quad \Leftrightarrow\ x' = x_0x_1\cdots x_n\cdots,\ x_0x_1 \in \tau^{\dot{2}},\ \varphi_\omega(x_1x_2\cdots x_n\cdots) \subseteq Y \\ \quad \Leftrightarrow\ \varphi_\omega(x') = \varphi_\omega(x_0x_1x_2\cdots x_n\cdots) \subseteq \varphi_\omega(\tau^{\dot{2}} \frown Y) \\ \quad \Leftrightarrow\ x' \in \gamma_\omega\varphi_\omega(\tau^{\dot{2}} \frown Y)\end{array}$$

where in the last implications we have to consider $\varphi_\omega(\tau^{\dot{2}} \frown Y)$ instead of $\tau^{\dot{2}} \frown Y$ in order to have also the prefixes of x' whose length is 1. ■

At this point, let us show that F^{safe} is co-additive, meaning also that we can reach its fixpoint in at most ω iterations.

PROPOSITION 3.6 *F^{safe} is co-additive.*

Proof

$$\begin{array}{rll} F^{\text{safe}}(\prod_i^{\text{safe}} X_i) = & F^{\text{safe}}(\varphi_\omega(\bigcap_i \gamma_\omega(X_i))) & \\ = & \varphi_\omega(\tau^{\dot{2}} \frown \varphi_\omega(\bigcap_i \gamma_\omega(X_i))) & [\ \text{by Lemma 3.5(i)}\ ] \\ = & \varphi_\omega(\tau^{\dot{2}} \frown \bigcap_i \gamma_\omega(X_i)) & [\ \text{by Lemma 3.4}\ ] \\ = & \varphi_\omega(\bigcap_i (\tau^{\dot{2}} \frown \gamma_\omega(X_i))) & [\ \text{by Lemma 3.5(ii)}\ ] \\ = & \varphi_\omega(\bigcap_i \gamma_\omega\varphi_\omega(\tau^{\dot{2}} \frown X_i)) = \prod_i^{\text{safe}} F^{\text{safe}}(X_i) & \end{array}$$

■

Finally, we can prove that φ_ω preserves the iterations of F^ω, and the previous result justifies the fact that we do not consider transfinite iterations.

PROPOSITION 3.7 $\varphi_\omega(\bigcap_{n\in\mathbb{N}}(F^\omega)^n(\Sigma^\omega)) = \prod^{safe}_{n\in\mathbb{N}} \varphi_\omega((F^\omega)^n(\Sigma^\omega))$

Proof Note that $(F^\omega)^n(\Sigma^\omega) = \tau^{n\dot{+}1} \frown \Sigma^\omega$ [10]. Therefore we have to prove that $\varphi_\omega(\bigcap_{n\in\mathbb{N}}(\tau^{n\dot{+}1} \frown \Sigma^\omega) = \prod^{\mathrm{safe}}_{n\in\mathbb{N}} \varphi_\omega(\tau^{n\dot{+}1} \frown \Sigma^\omega)$. By definition of $\sqcap^{\mathrm{safe}}$ (see Proposition 3.2) we have $\prod^{\mathrm{safe}}_{n\in\mathbb{N}} \varphi_\omega(\tau^{n\dot{+}1} \frown \Sigma^\omega) = \varphi_\omega(\bigcap_{n\in\mathbb{N}} \gamma_\omega\varphi_\omega(\tau^{n\dot{+}1} \frown \Sigma^\omega))$. Let $n \in \mathbb{N}$, let us show $\tau^{n\dot{+}1} \frown \Sigma^\omega = \gamma_\omega\varphi_\omega(\tau^{n\dot{+}1} \frown \Sigma^\omega)$. Clearly the inclusion $\subseteq$ comes from the extensivity of *Safe*. Let us prove the other inclusion. Consider $\delta \in \gamma_\omega\varphi_\omega(\tau^{n\dot{+}1} \frown \Sigma^\omega)$, then by definition of γ_ω this implies that $\varphi_\omega(\delta) \subseteq \varphi_\omega(\tau^{n\dot{+}1} \frown \Sigma^\omega)$. Suppose $\delta \notin \tau^{n\dot{+}1} \frown \Sigma^\omega$, then $\exists i \leq n+1\,.\,(\delta_i, \delta_{i+1}) \notin \tau$, therefore we have $\delta_0 \ldots \delta_{i+1} \in \varphi_\omega(\delta)$ but $\delta_0 \ldots \delta_{i+1} \notin \varphi_\omega(\tau^{n\dot{+}1} \frown \Sigma^\omega)$, which is absurd for the inclusion above. Hence $\delta \in \tau^{n\dot{+}1} \frown \Sigma^\omega$. The equality just proved implies trivially the thesis. ■

At this point, in order to apply the Kleene transfer theorem we have simply to show the commutative property, namely $F^{\mathrm{safe}} \circ \varphi_\omega = \varphi_\omega \circ F^\omega$, which corresponds to saying that the abstraction φ_ω is complete with respect to the operation F^ω, i.e., $\varphi_\omega \circ F^\omega \circ \varphi_\omega = \varphi_\omega \circ F^\omega$ [13], being $F^{\mathrm{safe}} \stackrel{\mathrm{def}}{=} \varphi_\omega \circ F^\omega$. This is precisely what we proved in the first point of Lemma 3.5.
The next theorem collects together all the properties we proved for F^ω and φ_ω giving a fixpoint characterization of safety semantics as the greatest fixpoint of F^{safe}, obtained by Kleene fixpoint transfer.

THEOREM 3.8 $\tau^{safe} = gfp^{\subseteq}_{\Sigma^+} F^{safe}$.

Proof We can prove the theorem by using the dual of Kleene's fixpoint transfer theorem. Moreover by Lemma 3.5(i) we can simply verify that F^{safe} is complete with respect to abstraction φ_ω and to the function F^ω. By Prop. 3.6 we have that at least in ω iterations we find the fixpoint. Finally by the Prop. 3.7 we know that the abstraction function commutes with finite iterations of F^ω so we can apply the dual of Kleene's fixpoint transfer theorem. Therefore we have that $\tau^{\mathrm{safe}} = \varphi_\omega(gfp^{\subseteq}_{\Sigma^\omega} F^\omega) = gfp^{\subseteq}_{\Sigma^+} F^{\mathrm{safe}}$. ■

Example 3.9 Let us consider a very simple example where we can show how the operator F^{safe} builds the safety semantics. Consider the transition system $\tau = \{\langle a, a\rangle, \langle a, c\rangle, \langle c, c\rangle\}$. Then we have the following iterations:

$$
\begin{aligned}
F^{\mathrm{safe}}(\Sigma^+) &= \varphi_\omega(\tau^{\dot{2}} \frown \Sigma^+) = \varphi_\omega(\{\ a\sigma, c\sigma \,|\, \sigma \in \Sigma^+\ \}) \\
&= \{a, c\} \cup \{\ a\sigma, c\sigma \,|\, \sigma \in \Sigma^+\ \} \stackrel{\mathrm{def}}{=} F_1 \\
F^{\mathrm{safe}}(F_1) &= \varphi_\omega(\tau^{\dot{2}} \frown F_1) = \varphi_\omega(\{aa, ac, cc\} \cup \{\ aa\sigma, ac\sigma, cc\sigma \,|\, \sigma \in \Sigma^+\ \}) \\
&= \{a, c, aa, ac, cc\} \cup \{\ aa\sigma, ac\sigma, cc\sigma \,|\, \sigma \in \Sigma^+\ \} \stackrel{\mathrm{def}}{=} F_2 \\
F^{\mathrm{safe}}(F_2) &= \varphi_\omega(\tau^{\dot{2}} \frown F_2) \\
&= \varphi_\omega(\{aa, ac, cc, aaa, aac, acc, ccc\} \cup \{\ aaa\sigma, aac\sigma, acc\sigma, ccc\sigma \,|\, \sigma \in \Sigma^+\ \}) \\
&= \{a, c, aa, ac, cc, aaa, aac, acc, ccc\} \cup \{\ aaa\sigma, aac\sigma, acc\sigma, ccc\sigma \,|\, \sigma \in \Sigma^+\ \} \\
&\ldots \\
F^{\mathrm{safe}}(F_n) &= \varphi_\omega(\tau^{\dot{2}} \frown F_n) \\
&= \varphi_\omega(\{\ a^i, c^i, a^j c^k \,|\, 2 \leq i \leq n+1,\ 1 \leq j, k \leq n\ \} \cup \\
&\qquad \{\ a^{n+1}\sigma, c^{n+1}\sigma, a^j c^k \sigma \,|\, \sigma \in \Sigma^+,\ 1 \leq j, k \leq n\ \}) \\
&= \{\ a^i, c^i, a^j c^k \,|\, 1 \leq i \leq n+1,\ 1 \leq j, k \leq n\ \} \cup \\
&\qquad \{\ a^{n+1}\sigma, c^{n+1}\sigma, a^j c^k \sigma \,|\, \sigma \in \Sigma^+,\ 1 \leq j, k \leq n\ \}
\end{aligned}
$$

It is quite straightforward to check that the greatest fixpoint is $\tau^{\mathrm{safe}} = \{\ a^j c^k \,|\, j, k \in \mathbb{N}\ \}$.

Finally, let us note that, in this case, we can also show how F^{safe} generates τ^{safe}. In

particular, note that the n^{th} iteration of F^{safe} is $X^n = \varphi_\omega(\tau^{n\dot{+}1} \frown \Sigma^+)$ (by induction and by Lemma 3.5). Next result is a property of φ_ω useful for characterizing the fixpoint of F^{safe} without using the transfer theorem.

PROPOSITION 3.10 *Consider* $\delta \in \Sigma^\omega$, *then we have* $\forall n \in \mathbb{N} \,.\, \varphi_\omega(\delta) \subseteq \varphi_\omega(\tau^{\dot{n}} \frown \Sigma^+) \;\Leftrightarrow\; \forall n \in \mathbb{N} \,.\, \delta \in \tau^{\dot{n}} \frown \Sigma^\omega$

Proof Suppose that $\forall n \in \mathbb{N} \,.\, \varphi_\omega(\delta) \subseteq \varphi_\omega(\tau^{\dot{n}} \frown \Sigma^+)$, and that $\exists n \in \mathbb{N} \,.\, \delta \notin \tau^{\dot{n}} \frown \Sigma^\omega$ then there must exists $\delta_{i-1}, \delta_i \in \Sigma$ with $i \leq n$ such that $(\delta_{i-1}, \delta_i) \notin \tau$. This implies that $\varphi_\omega(\delta) \not\subseteq \varphi_\omega(\tau^{\dot{i}} \frown \Sigma^+)$, which is absurd. Suppose now that $\forall n \in \mathbb{N} \,.\, \delta \in \tau^{\dot{n}} \frown \Sigma^\omega$, and that $\exists n \in \mathbb{N} \,.\, \varphi_\omega(\delta) \not\subseteq \varphi_\omega(\tau^{\dot{n}} \frown \Sigma^+)$, then $\exists x \in \varphi_\omega(\delta) \,.\, x \notin \varphi_\omega(\tau^{\dot{n}} \frown \Sigma^+)$ and there are at least two states $x_{i-1}, x_i \in \Sigma$ with $i \leq n$ such that $(x_{i-1}, x_i) \notin \tau$. This means that $\delta \notin \tau^{\dot{i}} \frown \Sigma^\omega$, which is absurd. ■

Hence, we can provide the following direct proof of Th. 3.8.

$$
\begin{array}{rl}
gfp^{\subseteq}_{\Sigma^+} F^{\text{safe}} & = \bigsqcap^{\text{safe}}_{n\in\mathbb{N}} X^n = \bigsqcap^{\text{safe}}_{n\in\mathbb{N}} \varphi_\omega(\tau^{n\dot{+}1} \frown \Sigma^+) = \bigsqcap^{\text{safe}}_{n>0} \varphi_\omega(\tau^{\dot{n}} \frown \Sigma^+) \\
& = \varphi_\omega(\bigcap_{n>0} \gamma_\omega \varphi_\omega(\tau^{\dot{n}} \frown \Sigma^+)) \\
& = \varphi_\omega\left(\bigcap_{n>0} \left\{ \sigma \in \Sigma^\omega \,\middle|\, \varphi_\omega(\sigma) \subseteq \varphi_\omega(\tau^{\dot{n}} \frown \Sigma^+) \right\}\right) \\
& = \left\{ x \in \Sigma^+ \,\middle|\, \exists \delta \in \bigcap_{n>0} \left\{ \sigma \in \Sigma^\omega \,\middle|\, \varphi_\omega(\sigma) \subseteq \varphi_\omega(\tau^{\dot{n}} \frown \Sigma^+) \right\} \,.\, x \preccurlyeq \delta \right\} \\
& = \left\{ x \in \Sigma^+ \,\middle|\, \exists \delta \in \Sigma^\omega \,.\, \forall n > 0 \,.\, \varphi_\omega(\delta) \subseteq \varphi_\omega(\tau^{\dot{n}} \frown \Sigma^+) \;\wedge\; x \preccurlyeq \delta \right\} \\
& \qquad [\text{ by Prop. 3.10 }] \\
& = \left\{ x \in \Sigma^+ \,\middle|\, \exists \delta \in \Sigma^\omega \,.\, \forall n > 0 \,.\, \delta \in \tau^{\dot{n}} \frown \Sigma^\omega \;\wedge\; x \preccurlyeq \delta \right\} \\
& = \left\{ x \in \Sigma^+ \,\middle|\, \exists \delta \in \bigcap_{n>0} \tau^{\dot{n}} \frown \Sigma^\omega \,.\, x \preccurlyeq \delta \right\} \\
& = \varphi_\omega(\bigcap_{n>0} \tau^{\dot{n}} \frown \Sigma^\omega) = \varphi_\omega(\bigcap_{n\in\mathbb{N}} \tau^{n\dot{+}1} \frown \Sigma^\omega) = \varphi_\omega(\tau^\omega) = \tau^{\text{safe}}
\end{array}
$$

At this point, we can underline that the fixpoint construction explicitly described above can be interpreted as monitoring, for two main observations. First, the n^{th} iteration of F^{safe}, i.e., $\varphi_\omega(\tau^{n\dot{+}1} \frown \Sigma^+)$, corresponds to the set of all the prefixes of all the computations that at least for n steps are computations of the considered program. From the abstract interpretation point of view it is like to abstract traces only to the first $n+1$ states, or in other words, to observe only the first n steps. Second, we can note that if we check the program for n steps, and therefore for all the prefixes of these steps of computation, in order to check the program for $n+1$ steps it is sufficient to move one step forward in the computation, since we know that $\varphi_\omega(\tau^{n+1}) = \tau^{n+1} \cup \varphi_\omega(\tau^n)$. But these two things together, intuitively, provide a theoretical description of how an execution monitor works.

4. Other safety properties as abstractions in the hierarchy

In this section, we consider three different kinds of safety semantics known in the literature, and we show that all of them can be modelled as abstractions of safety in the hierarchy of semantics. This characterization is important also because allows us to prove that some kind of safety properties, in particular those admitting cancellation of states, lose the well-known constructive nature.
We mainly focus on two notions of safety: safety *without stuttering* [2] (also called *stuttering safety*) and *strong safety* [38]. Intuitively a property is safety without stuttering if it is safety and if it is insensitive with respect to the repetition of states. In other words, a property is without stuttering if, given a sequence of states σ that satisfies the property, then any other sequence σ' that differs from σ only for the repetition of a set of states of σ, satisfies the property. An example of property without stuttering is the following: Consider the sequence σ of states

representing the evolution of a clock with a variable h for hours and m for minutes. Then consider another sequence σ' again representing a clock with a variable h for hours, a variable m for minutes and a variable s for seconds. Then a property without stuttering cannot distinguish the two sequences even if σ' evolves in 59 consecutive different states while σ does not change [29] (namely repeats for 59 times the same state). On the other hand a property Π is a strong safety property, if it is a safety property without stuttering and is insensitive to deletion of states, i.e., from any sequence in Π if we delete an arbitrary number of states, then the resulting sequence is also in Π. In the following we will identify a trace property as the set of traces satisfying the property.

DEFINITION 4.1 *Let Π be a property on (potentially infinite) traces. Then Π is safety without stuttering if it is safety and if*

$$\sigma \in \Pi \,.\, \sigma = \sigma_0\sigma_1 \ldots \sigma_n \ldots \;\; \textit{then} \; \forall i \geq 0 \,.\, \sigma_0 \ldots \sigma_i\sigma_i \ldots \in \Pi$$

Π is strong safety if it is safety without stuttering and if

$$(*) \qquad \sigma \in \Pi \,.\, \sigma = \sigma_0\sigma_1 \ldots \sigma_n \ldots \;\; \textit{then} \; \forall i > 0 \,.\, \sigma_0 \ldots \sigma_{i-1}\sigma_{i+1} \ldots \in \Pi$$

In Definition 4.1 we call *cancellation safety* a safety property that satisfies only $(*)$. Note that in the cancellation property it is assumed that the initial state is always observed [37].
The importance of properties without stuttering is in both requirement and system specification. In system specification, a property with stuttering exposes too much details of the internal structure, while in requirement specifications these properties preclude, in model checking, efficient verification [33]. The meaning of the definition of strong safety properties is that if we do not observe the system during certain instances then the observed behaviour should still be permissible, and similarly if we observe the same state many times before a state change occurs, then the resulting behaviour should still be permissible. The strong safety properties are important since invariant properties are a subset of them [37].

At this point we can define the abstractions characterizing the restricted safety properties as abstractions of $\mathcal{S}$. Let us define these abstractions in the most general form, namely consider the abstraction given in Table 3, where, for each $\alpha \in \{\alpha^{\mathrm{str}}, \alpha^{\mathrm{stu}}, \alpha^{\mathrm{can}}\}$ we have $\alpha : \wp(\Sigma^\infty) \to \alpha(\wp(\Sigma^\infty))$, namely α is generically defined in the set of all the possible traces, even if the corresponding semantics in the hierarchy are obtained by applying α to the set $\mathcal{S}$. In the following we will call all these new abstract safety semantics *restricted safety properties*. Note that the set of properties characterized by α^{str} contains the properties characterized by both α^{stu} and α^{can}. This is coherent with Definition 4.1 where we can note that both safety without stuttering and cancellation safety properties are particular strong safety properties.

Consider the definitions in Table 3. From these definitions it turns out that the three abstractions differ only for the hypotheses on the number of possible repetitions k_i. In the following, for each α in Table 3, we write $k_i \in D_\alpha$ in order to denote that k_i respect the hypothesis imposed by the abstraction α. In particular we have that $D_{\alpha^{\mathrm{stu}}} = \mathbb{N} \setminus \{0\}$, $k_i \in D_{\alpha^{\mathrm{can}}}$ means that $\forall i > 0 \,.\, k_i \in \{0, 1\}$, while $k_0 = 1$, and $k_i \in D_{\alpha^{\mathrm{str}}}$ means that $\forall i > 0 \,.\, k_i \in \mathbb{N}$ while $k_0 \in \mathbb{N} \setminus \{0\}$. The following lemma says that the restricted properties commute with the safety abstraction, and this property is important afterwards for proving that the α abstractions are

Safety property	Abstraction and concretization
Stuttering safety:	$\alpha^{\mathrm{stu}}(X) \stackrel{\mathrm{def}}{=} \left\{ x \in \Sigma^\infty \left\vert \begin{array}{l} \exists y \in X \,.\, x = y_0^{k_0} y_1^{k_1} \ldots y_n^{k_n} \ldots, \\ \forall i \,.\, k_i \in \mathbb{N} \smallsetminus \{0\} \end{array} \right. \right\}$
Cancellation safety:	$\alpha^{\mathrm{can}}(X) \stackrel{\mathrm{def}}{=} \left\{ x \in \Sigma^\infty \left\vert \begin{array}{l} \exists y \in X \,.\, x = y_0 y_1^{k_1} \ldots y_n^{k_n} \ldots, \\ \forall i > 0 \,.\, k_i \in \{0,1\} \end{array} \right. \right\}$
Strong safety:	$\alpha^{\mathrm{str}}(X) \stackrel{\mathrm{def}}{=} \left\{ x \in \Sigma^\infty \left\vert \begin{array}{l} \exists y \in X \,.\, x = y_0^{k_0} y_1^{k_1} \ldots y_n^{k_n} \ldots, \\ \forall i > 0 \,.\, k_i \in \mathbb{N},\ k_0 \in \mathbb{N} \smallsetminus \{0\} \end{array} \right. \right\}$

Table 3. Restricted safety properties

closure operators on the safety abstraction domain. In the following, we consider again the extension of φ_ω to any set of (finite or infinite) traces.

LEMMA 4.2 *Let $\alpha \in \{\alpha^{str}, \alpha^{stu}, \alpha^{can}\}$, and $\sigma \in \Sigma^\omega$. Then $\varphi_\omega(\alpha(\sigma)) = \alpha(\varphi_\omega(\sigma))$*[1].

Proof Let $x \in \varphi_\omega(\alpha(\sigma))$ then there exists $\sigma' \in \alpha(\sigma)$ such that $x \preccurlyeq \sigma'$. Since $\sigma' = \sigma_0^{k_0}\sigma_1^{k_1}\ldots$ then there exists i such that $x = \sigma_0^{k_0}\sigma_1^{k_1}\ldots\sigma_i^{k_i}$. Since $\sigma_0\sigma_1\ldots\sigma_i \preccurlyeq \sigma$ and $k_0, k_1, \ldots k_i \in \mathbb{N}$ we have that $x \in \alpha\varphi_\omega(\sigma)$.
Consider now $x \in \alpha\varphi_\omega(\sigma)$. Then $x = x_0^{k_0}x_1^{k_1}\ldots x_n^{k_n}$ with $x_0x_1\ldots x_n \preccurlyeq \sigma$. Let $\beta \in \Sigma^\omega$ such that $x_0x_1\ldots x_n\beta = \sigma$, then $x_0^{k_0}x_1^{k_1}\ldots x_n^{k_n}\beta \in \alpha(\sigma)$. This clearly implies that $x \in \varphi_\omega(\alpha(\sigma))$. ■

PROPOSITION 4.3 *Let $\alpha \in \{\alpha^{str}, \alpha^{stu}, \alpha^{can}\}$. α is an upper closure operator, i.e., $\alpha(\mathcal{S})$ is a Moore family of $\mathcal{S}$.*

Proof In order to show that $\alpha(\mathcal{S})$ is a Moore family of $\mathcal{S}$ we have to prove that, given a family $\{X_i\}_{i\in I} \subseteq \alpha(\mathcal{S})$, we have $\bigsqcap_i^{\mathrm{safe}} X_i \in \alpha(\mathcal{S})$. Recall that $\bigsqcap_i^{\mathrm{safe}} X_i = \varphi_\omega(\bigcap_i \gamma_\omega(X_i))$. Consider the following relations.

$$
\begin{array}{ll}
x = x_0\ldots x_h \in \varphi_\omega(\bigcap_i \gamma_\omega(X_i)) & \Rightarrow\ \exists\sigma \in \bigcap_i \gamma_\omega(X_i) \,.\, x \preccurlyeq \sigma \\
& \Rightarrow\ \exists\sigma \,.\, x \preccurlyeq \sigma,\ \forall i \,.\, \sigma \in \gamma_\omega(X_i)\ \Rightarrow\ \exists\sigma \,.\, x \preccurlyeq \sigma,\ \forall i \,.\, \varphi_\omega(\sigma) \subseteq X_i \\
& \Rightarrow\ \exists\sigma \,.\, x \preccurlyeq \sigma,\ \forall i \,.\, \alpha(\varphi_\omega(\sigma)) \subseteq X_i,\ [\ \text{being } X_i \in \alpha(\mathcal{S})\ ] \\
& \Rightarrow\ \exists\sigma \,.\, x \preccurlyeq \sigma,\ \forall i \,.\, \varphi_\omega(\alpha(\sigma)) \subseteq X_i,\ [\ \text{by Lemma 4.2}\ ] \\
& \Rightarrow\ \exists\sigma \,.\, x \preccurlyeq \sigma,\ \forall i \,.\, \alpha(\sigma) \subseteq \gamma_\omega(X_i) \\
& \Rightarrow\ \exists\sigma \,.\, x \preccurlyeq \sigma,\ \alpha(\sigma) \subseteq \bigcap_i \gamma_\omega(X_i) \\
& \Rightarrow\ \exists\sigma \,.\, x \preccurlyeq \sigma,\ \varphi_\omega(\alpha(\sigma)) \subseteq \varphi_\omega(\bigcap_i \gamma_\omega(X_i)) \\
& \Rightarrow\ \exists\sigma \,.\, x \preccurlyeq \sigma,\ \alpha(\varphi_\omega(\sigma)) \subseteq \varphi_\omega(\bigcap_i \gamma_\omega(X_i)),\ [\ \text{by Lemma 4.2}\ ] \\
& \Rightarrow\ \alpha(x) \subseteq \varphi_\omega(\bigcap_i \gamma_\omega(X_i)),\ [\ \text{by monotonicity of } \alpha, \text{ being } x \in \varphi_\omega(\sigma)\ ]
\end{array}
$$

We proved in this way that $\bigsqcap_i^{\mathrm{safe}} X_i \in \alpha(\mathcal{S})$, namely that $\bigsqcap_i^{\mathrm{safe}} X_i$ is an α safety property. ■

The proposition above implies that for any $X, Y \in \alpha(\mathcal{S})$, where α is a restricted safety, we have that $\alpha(X \sqcap^{\mathrm{safe}} Y) = X \sqcap^{\mathrm{safe}} Y$ being α a closure.

Now that safety without stuttering, as well as all the other restricted safety properties, are included in the Cousot's hierarchy of semantics as abstractions of the safety semantics, we can derive them as fixpoints of a semantic operator designed by fixpoint transfer from the fixpoint safety semantics given in the previous section. Consider the dual Kleene fixpoint transfer introduced before. We want to obtain

[1] α applied to infinite traces is the natural extension of the corresponding function defined in Table 3.

any of the abstract safety properties introduced so far as the greatest fixpoint of the semantic operator

$$F^{\alpha} = \alpha F^{\mathrm{safe}}$$

In this case, the concrete domain is the safety semantics τ^{safe}, the abstract semantics are the different τ^{α}, the abstractions are the respective α, and the operators are F^{safe} (concrete) and F^{α} (abstract). As we noticed in Sect. 3, in order to apply the transfer theorem the abstraction function has to be co-continuous. Unfortunately we can show that all the abstractions introduced so far are not co-continuous.

PROPOSITION 4.4 *Let $\alpha \in \{\alpha^{str}, \alpha^{stu}, \alpha^{can}\}$. Then α is not co-continuous.*

Proof We show first how the cancellation property makes the co-continuity to fail. Let $\alpha \in \{\alpha^{\mathrm{can}}, \alpha^{\mathrm{str}}\}$ Consider $\forall n \in \mathbb{N}\,.\, X_n \stackrel{\mathrm{def}}{=} \left\{\, x \in \Sigma^+ \,\middle|\, |x| \geq n+1 \;\Rightarrow\; x_n = a \,\right\}$, clearly $\forall n \in \mathbb{N}\,.\, X_n \in \mathcal{S}$ since $\forall n\,.\, X_n = \varphi_\omega(Y_n)$ where $Y_n = \left\{\, \sigma \in \Sigma^\omega \,\middle|\, \sigma_n = a \,\right\}$. $\alpha(X_n) = \left\{\, x_0^{k_0} \ldots x_m^{k_m} \,\middle|\, k_i \in D_\alpha, (m \geq n+1 \;\Rightarrow\; x_n = a) \,\right\}$. Note that $\alpha(X_n) = \Sigma^+$, indeed let $y \in \Sigma^+$, then if $|y| \leq n$ or $y_n = a$ it is in $\alpha(X_n)$. Let us consider $|y| > n$ and $y_n \neq a$, then we can write $y = (y_0)^1 \ldots (y_{n-1})^1 a^0 (y_n)^1 \ldots (y_m)^1$ where clearly $y_0 \ldots y_{n-1} a y_n \ldots y_m \in X_n$, therefore $y \in \alpha(X_n)$. But this implies immediately that $\forall n \in \mathbb{N}\,.\, \gamma_\omega\alpha(X_n) = \Sigma^\omega$ and therefore $\bigsqcap^{\mathrm{safe}}_{n\in\mathbb{N}} \alpha(X_n) = \varphi_\omega(\bigcap_{n\in\mathbb{N}} \gamma_\omega\alpha(X_n)) = \Sigma^\omega$. On the other hand we have that $\gamma_\omega(X_n) = \left\{\, \sigma \in \Sigma^\omega \,\middle|\, \sigma_n = a \,\right\}$, therefore $\bigcap_{n\in\mathbb{N}} \gamma_\omega(X_n) = \left\{\, \sigma \in \Sigma^\omega \,\middle|\, \forall n \in \mathbb{N}\,.\, \sigma_n = a \,\right\} = \{a^\omega\}$. This means that $\varphi_\omega(\bigcap_{n\in\mathbb{N}} \gamma_\omega(X_n)) = \{a, aa, aaa, \ldots\}$, i.e., $\alpha(\bigsqcap^{\mathrm{safe}}_{n\in\mathbb{N}} X_n) = \alpha(\varphi_\omega(\bigcap_{n\in\mathbb{N}} \gamma_\omega(X_n))) = \{a, aa, aaa, \ldots\}$, clearly different from Σ^ω.

Consider now α^{stu} and the sets $X_n \stackrel{\mathrm{def}}{=} \left\{\, a^i \,\middle|\, i \leq n \,\right\} \cup \left\{\, a^n x \,\middle|\, x \in \Sigma^+,\; b \in x \,\right\}$ which compose a decreasing chain. Then we have that the concretization is $\gamma_\omega(X_n) = \left\{\, \sigma \in \Sigma^\omega \,\middle|\, a^n \preccurlyeq \sigma,\; b \in \sigma \,\right\}$. Clearly, as in Prop. 5.1 we can note that $\bigcap_n \gamma_\omega(X_n) = \varnothing$. On the other hand for each n we have that $\left\{\, a^i \,\middle|\, i \in \mathbb{N} \,\right\} \subseteq \alpha^{\mathrm{stu}}(X_n)$, therefore $a^\omega \in \bigcap_n \gamma_\omega\alpha^{\mathrm{stu}}(X_n)$. From these facts we have $\bigsqcap_n \alpha(X_n) = \varphi_\omega(\bigcap_n \gamma_\omega\alpha(X_n)) \neq \varnothing$ while $\alpha(\bigsqcap_n X_i) = \alpha\varphi_\omega(\bigcap_n \gamma_\omega X_n) = \varnothing$ ■

Therefore all the abstractions introduced are not co-continuous. Anyway, as noticed before, co-continuity is a too strong condition. Indeed, it would be sufficient to prove that the abstraction functions introduced above preserve the greatest lower bound of the iterations of F^{safe}. Unfortunately, this holds for α^{stu}, but it does not hold for the other restricted safety properties as it is shown in the following example.

Example 4.5 It is worth noting that the dual Kleene transfer fixpoint theorem, also in its weakened form, is not applicable to strong and cancellation safety properties to generate a fixpoint semantics of them. The following example shows that the two restricted abstractions mentioned above do not commute with the iterations of F^{safe}. Let $\alpha \in \{\alpha^{\mathrm{str}}, \alpha^{\mathrm{can}}\}$. Consider the transition system with $\Sigma = \{a, b, c\}$ and $\tau = \{\langle a, b\rangle, \langle b, b\rangle, \langle b, c\rangle\}$, with c is a terminal state. Note that for each n we have that $ab^{n-2}c^\frown\Sigma^\omega \subseteq \tau^{\dot{n}}{}^\frown\Sigma^\omega$. This implies that $\forall n \in \mathbb{N}\,.\, ab^{n-2}ca^\omega \in \tau^{\dot{n}}{}^\frown\Sigma^\omega$. Consider $\forall i \in \mathbb{N}.aca^i$, then $\forall n \in \mathbb{N}, \forall i \in \mathbb{N}.aca^i \in \alpha\varphi_\omega(ab^{n-2}ca^\omega) \subseteq \alpha\varphi_\omega(\tau^{\dot{n}}{}^\frown\Sigma^\omega)$ since $aca^i = ab^0 \ldots b^0 ca^i$. Being $\varphi_\omega(aca^\omega) = \{a\} \cup \left\{\, aca^i \,\middle|\, i \in \mathbb{N} \,\right\}$, we have that $\forall n \in \mathbb{N}.\varphi_\omega(aca^\omega) \subseteq \alpha\varphi_\omega(\tau^{\dot{n}}{}^\frown\Sigma^\omega)$, namely $\forall n \in \mathbb{N}.aca^\omega \in \gamma_\omega\alpha\varphi_\omega(\tau^{\dot{n}}{}^\frown\Sigma^\omega)$. Hence we have the following implications

$$\begin{aligned} aca^\omega \in \textstyle\bigcap_n \gamma_\omega\alpha\varphi_\omega(\tau^{\dot{n}}{}^\frown\Sigma^\omega) &\Rightarrow \forall i \in \mathbb{N}\,.\, aca^i \in \varphi_\omega \textstyle\bigcap_n \gamma_\omega\alpha\varphi_\omega(\tau^{\dot{n}}{}^\frown\Sigma^\omega) \\ &\Rightarrow \forall i \in \mathbb{N}\,.\, aca^i \in \textstyle\bigsqcap_n \alpha\varphi_\omega(\tau^{\dot{n}}{}^\frown\Sigma^\omega) \end{aligned}$$

On the other hand, we have that $\gamma_\omega\varphi_\omega(\tau^{\dot n}\frown\Sigma^\omega) = \tau^{\dot n}\frown\Sigma^\omega$ (see the proof of Prop. 3.7). Then it is worth noting that $\bigcap_n \tau^{\dot n}\frown\Sigma^\omega = \{ab^\omega, b^\omega\}$. Therefore

$$\begin{aligned}\alpha\textstyle\bigcap_n \varphi_\omega(\tau^{\dot n}\frown\Sigma^\omega) &= \alpha\varphi_\omega\textstyle\bigcap_n \gamma_\omega\varphi_\omega(\tau^{\dot n}\frown\Sigma^\omega) = \alpha\varphi_\omega\textstyle\bigcap_n \tau^{\dot n}\frown\Sigma^\omega\\ &= \alpha(\{\ ab^i \mid i\in\mathbb{N}\ \}\cup\{\ b^i \mid i\in\mathbb{N}\ \})\end{aligned}$$

Now, if $\alpha = \alpha^{\mathrm{str}}$, then $\alpha^{\mathrm{str}}(\{\ ab^i \mid i\in\mathbb{N}\ \}\cup\{\ b^i \mid i\in\mathbb{N}\ \}) = \alpha^{\mathrm{str}}(\{\ ab^i \mid i\in\mathbb{N}\ \})\cup \alpha^{\mathrm{str}}(\{\ b^i \mid i\in\mathbb{N}\ \}) = \{\ a^jb^i \mid i,j\in\mathbb{N}, j>0\ \}\cup\{\ b^i \mid i\in\mathbb{N}\ \} = \{\ a^jb^i \mid i,j\in\mathbb{N}\ \}$. While, if $\alpha = \alpha^{\mathrm{can}}$ then $\alpha^{\mathrm{can}}(\{\ ab^i \mid i\in\mathbb{N}\ \}\cup\{\ b^i \mid i\in\mathbb{N}\ \}) = \alpha^{\mathrm{can}}(\{\ ab^i \mid i\in\mathbb{N}\ \})\cup \alpha^{\mathrm{can}}(\{\ b^i \mid i\in\mathbb{N}\ \}) = \{\ ab^i \mid i\in\mathbb{N}\ \}\cup\{\ b^i \mid i\in\mathbb{N}\ \}$. In both cases, we have that $\forall i\,.\, aca^i \notin \alpha\textstyle\bigcap_n \varphi_\omega(\tau^{\dot n}\frown\Sigma^\omega)$.

Hence, in the following we can investigate only on the fixpoint construction of safety without stuttering.
Next results show precisely that α^{stu} preserves the greatest lower bounds of the iterations of $F^{\alpha^{\mathrm{stu}}}$. As before, we have first to avoid transfinite iterations, proving simply that $F^{\alpha^{\mathrm{stu}}}$ reaches the fixpoint before ω iterations, in order to show the preservation of glb only for ω limited greatest lower bounds.

LEMMA 4.6 *Let $\alpha = \alpha^{stu}$, then we have that*

$$\forall n\in\mathbb{N}\,.\,\forall X\in\alpha(\mathcal{S})\,.\,(F^\alpha)^n(X) = \alpha((F^{safe})^n(X))$$

Proof Let $\alpha = \alpha^{\mathrm{stu}}$. We prove the thesis by induction on the number of applications of F^α. By definition we have $(F^\alpha(X))^0 = X$ and $\alpha((F^{\mathrm{safe}}(X))^0) = \alpha(X) = X$, being $X\in\alpha(\mathcal{S})$. Recall that $(F^\alpha)^n(X) \stackrel{\mathrm{def}}{=} (\alpha F^{\mathrm{safe}})^n(X)$. Let $(\alpha F^{\mathrm{safe}})^n(X) = \alpha((F^{\mathrm{safe}})^n(X))$ be the inductive hypothesis. We prove that this holds also for $n+1$. Consider

$$\begin{aligned}(F^\alpha)^{n+1}(X) &= (\alpha F^{\mathrm{safe}})^{n+1}(X) = (\alpha F^{\mathrm{safe}})((\alpha F^{\mathrm{safe}})^n(X))\\ &= (\alpha F^{\mathrm{safe}})(\alpha((F^{\mathrm{safe}})^n(X))) && [\text{ by inductive hypothesis }]\\ &= \alpha(F^{\mathrm{safe}}(\alpha((F^{\mathrm{safe}})^n(X)))) && [\text{ by composition }]\\ &= \alpha(F^{\mathrm{safe}}((F^{\mathrm{safe}})^n(X))) && [\text{ being } (F^{\mathrm{safe}})^n(X)\in\alpha(\mathcal{S}) \text{ }]\\ &= \alpha((F^{\mathrm{safe}})^{n+1}(X))\end{aligned}$$

■

PROPOSITION 4.7 *Let $\alpha = \alpha^{stu}$. Then*

$$F^\alpha\left(\prod_{n\in\mathbb{N}}^{safe}(F^\alpha)^n(\Sigma^+)\right) = \prod_{n\in\mathbb{N}}^{safe}(F^\alpha)^n(\Sigma^+).$$

Proof Recall that $F^\alpha \stackrel{\mathrm{def}}{=} \alpha\circ F^{\mathrm{safe}}$, therefore

$$\begin{aligned}\textstyle\prod^{\mathrm{safe}}_{n\in\mathbb{N}}(\alpha F^{\mathrm{safe}})^n(\Sigma^+) &= \textstyle\prod^{\mathrm{safe}}_{n\in\mathbb{N}}\alpha((F^{\mathrm{safe}})^n(\Sigma^+)) && [\text{ by Lemma 4.6 }]\\ &= \alpha\textstyle\prod^{\mathrm{safe}}_{n\in\mathbb{N}}(F^{\mathrm{safe}})^n(\Sigma^+) && [\text{ by Prop. 4.9 }]\\ &= \alpha F^{\mathrm{safe}}(\textstyle\prod^{\mathrm{safe}}_{n\in\mathbb{N}}(F^{\mathrm{safe}})^n(\Sigma^+)) && [\text{ by Prop. 3.6 }]\\ &= \alpha F^{\mathrm{safe}}\alpha(\textstyle\prod^{\mathrm{safe}}_{n\in\mathbb{N}}(F^{\mathrm{safe}})^n(\Sigma^+)) && [\text{ by Lemma 4.10 }]\\ &= \alpha F^{\mathrm{safe}}(\textstyle\prod^{\mathrm{safe}}_{n\in\mathbb{N}}\alpha(F^{\mathrm{safe}})^n(\Sigma^+)) && [\text{ by Prop. 4.9 }]\\ &= \alpha F^{\mathrm{safe}}(\textstyle\prod^{\mathrm{safe}}_{n\in\mathbb{N}}(\alpha F^{\mathrm{safe}})^n(\Sigma^+)) && [\text{ by Lemma 4.6 }]\end{aligned}$$

■

These results tell us that the fixpoint is reached at most in ω iterations, hence we have simply to show now that α^{stu} preserves ω-bounded iteration only.

LEMMA 4.8 *Let $\alpha = \alpha^{stu}$ and $\delta \in \Sigma^\omega$ then we have $\forall n \in \mathbb{N}$:*

$$\varphi_\omega(\delta) \subseteq \alpha\varphi_\omega(\tau^{\dot n} \frown \Sigma^+) \;\Leftrightarrow\; \delta \in \alpha(\tau^{\dot n} \frown \Sigma^\omega)$$

Proof ($\Rightarrow$) Consider $\delta \in \Sigma^\omega$, and $\varphi_\omega(\delta) \subseteq \alpha\varphi_\omega(\tau^{\dot n} \frown \Sigma^+)$. By definition of α this corresponds to saying that $\forall x \in \varphi_\omega(\delta)$ there exists $z \in \varphi_\omega(\tau^{\dot n} \frown \Sigma^+)$ such that $x = z_0^{k_0} z_1^{k_1} \dots z_m^{k_m}$, for some $m \in \mathbb{N}, k_0, k_1, \dots, k_m \in D_\alpha$. Now we prove that this fact implies that $\exists \sigma \in \Sigma^\omega$ such that $\varphi_\omega(\sigma) \subseteq \varphi_\omega(\tau^{\dot n} \frown \Sigma^+)$ and such that $\delta = \sigma_0^{k_0}\sigma_1^{k_1} \dots$ for $k_i \in D_\alpha$. Starting from $\varphi_\omega(\delta)$ we want to find a set of prefixes $\varphi_\omega(\sigma)$ for some $\sigma \in \Sigma^\omega$.
First of all we prove that if $x = ys$, with $x, y \in \Sigma^+$, $s \in \Sigma$ and such that $x = z_0^{k_0} \dots z_m^{k_m}$, $y = w_0^{h_0} \dots w_l^{h_l}$ with $z, w \in \varphi_\omega(\tau^{\dot n} \frown \Sigma^+)$, then we can find $w' \in \varphi_\omega(\tau^{\dot n} \frown \Sigma^+)$ such that $y = w_0^{\prime h'_0} \dots w_{l'}^{\prime h'_{l'}}$ and $w' \preccurlyeq z$. Indeed suppose, without losing generality, that k_m and h_l are different from 0, otherwise we would take the longest prefix of z and w with the last exponent different from 0 which is by construction in $\varphi_\omega(\tau^{\dot n} \frown \Sigma^+)$. The fact that $x = ys$ implies that $z_0^{k_0} \dots z_m^{k_m} = w_0^{h_0} \dots w_l^{h_l} s$. Therefore $z_0^{k_0} \dots z_m^{k_m - 1} = w_0^{h_0} \dots w_l^{h_l} = y$. Now if $z \in \varphi_\omega(\tau^{\dot n} \frown \Sigma^+)$ then also $w' \stackrel{\text{def}}{=} z_0 z_1 \dots z_{m-1} z_m^{\{0,1\}} \in \varphi_\omega(\tau^{\dot n} \frown \Sigma^+)$[1] since $z_0 z_1 \dots z_{m-1} z_m^{\{0,1\}} \preccurlyeq z$. In this way we found $w' \in \varphi_\omega(\tau^{\dot n} \frown \Sigma^+)$ such that $y \in \alpha(w')$ with $w' \preccurlyeq z$. It is worth noting that the set of these elements of $\varphi_\omega(\tau^{\dot n} \frown \Sigma^+)$ is an infinite set of prefixes, therefore it is the set of prefixes of a certain infinite trace σ, $\varphi_\omega(\sigma)$, and moreover the relation among prefixes of δ and σ implies that $\delta = \sigma_0^{k_0}\sigma_1^{k_1} \dots$. Therefore:

$$\begin{aligned}
\varphi_\omega(\delta) \subseteq \alpha\varphi_\omega(\tau^{\dot n} \frown \Sigma^+) &\Rightarrow \forall x \in \varphi_\omega(\delta)\,.\,\exists z \in \varphi_\omega(\tau^{\dot n} \frown \Sigma^+)\,.\,x = z_0^{k_0} \dots z_m^{k_m} \\
&\Rightarrow \exists \sigma \in \Sigma^\omega\,.\,\varphi_\omega(\sigma) \subseteq \varphi_\omega(\tau^{\dot n} \frown \Sigma^+),\ \delta = \sigma_0^{k_0}\sigma_1^{k_1} \dots \\
&\Rightarrow \exists \sigma \in \tau^{\dot n} \frown \Sigma^\omega\,.\,\delta = \sigma_0^{k_0}\sigma_1^{k_1} \dots \\
&\Rightarrow \delta \in \alpha(\tau^{\dot n} \frown \Sigma^\omega)
\end{aligned}$$

where it is trivial to verify that $\varphi_\omega(\sigma) \subseteq \varphi_\omega(\tau^{\dot n} \frown \Sigma^+)$ implies $\sigma \in \tau^{\dot n} \frown \Sigma^\omega$.
($\Leftarrow$) Consider $\delta \in \alpha(\tau^{\dot n} \frown \Sigma^\omega)$. Then the following implications hold:

$$\begin{aligned}
\delta \in \alpha(\tau^{\dot n} \frown \Sigma^\omega) &\Rightarrow \exists \sigma \in \tau^{\dot n} \frown \Sigma^\omega\,.\,\delta = \sigma_0^{k_0}\sigma_1^{k_1} \dots \\
&\Rightarrow \exists \sigma \in \Sigma^\omega\,.\,\varphi_\omega(\sigma) \subseteq \varphi_\omega(\tau^{\dot n} \frown \Sigma^+),\ \delta = \sigma_0^{k_0}\sigma_1^{k_1} \dots \\
&\Rightarrow \varphi_\omega(\delta) \subseteq \alpha(\varphi_\omega(\sigma)) \subseteq \alpha\varphi_\omega(\tau^{\dot n} \frown \Sigma^+)
\end{aligned}$$

where the last inclusions are due to the fact that $\delta = \sigma_0^{k_0}\sigma_1^{k_1} \dots$. ■

PROPOSITION 4.9 *Let $\alpha = \alpha^{stu}$. Then*

$$\prod_{n\in\mathbb{N}}^{safe} \alpha((F^{safe})^n(\Sigma^+)) = \alpha\left(\prod_{n\in\mathbb{N}}^{safe} (F^{safe})^n(\Sigma^+)\right)$$

Proof Note that it always holds that $\alpha(\bigcap_i X_i) \subseteq \bigcap_i^\alpha \alpha(X_i)$. This means that $\bigcap_{n\in\mathbb{N}}^{\text{safe}} \alpha((F^{\text{safe}})^n(\Sigma^+)) \supseteq \alpha\left(\bigcap_{n\in\mathbb{N}}^{\text{safe}} (F^{\text{safe}})^n(\Sigma^+)\right)$ holds trivially. Let us consider the other inclusion. We noted in Sect. 3 that, $(F^{\text{safe}})^n(\Sigma^+) = \varphi_\omega(\tau^{n\dot{+}1} \frown \Sigma^+)$, where φ_ω

[1] We wrote $z^{\{0,1\}}$ since we do not know if $k_m > 1$.

here is the extension to both finite and infinite traces defined in Sect 3. Therefore the following relations hold.

$$
\begin{array}{l}
x \in \prod^{\mathrm{safe}}_{n\in\mathbb{N}} \alpha((F^{\mathrm{safe}})^n(\Sigma^+)) \;\Rightarrow\; x \in \varphi_\omega(\bigcap_{n\in\mathbb{N}} \gamma_\omega\alpha((F^{\mathrm{safe}})^n(\Sigma^+))) \\
\quad\Rightarrow\; x \in \varphi_\omega(\bigcap_{n\in\mathbb{N}} \gamma_\omega\alpha\varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+)) \\
\quad\Rightarrow\; \exists\sigma \in \bigcap_{n\in\mathbb{N}} \gamma_\omega\alpha\varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+)\,.\, x \preccurlyeq \sigma \\
\quad\Rightarrow\; \exists\sigma\,.\,\forall n \in \mathbb{N}\,.\, \sigma \in \gamma_\omega\alpha\varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+),\; x \preccurlyeq \sigma \\
\quad\Rightarrow\; \exists\sigma\,.\,\forall n \in \mathbb{N}\,.\, \varphi_\omega(\sigma) \subseteq \alpha\varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+),\; x \preccurlyeq \sigma \\
\quad\Rightarrow\; \exists\sigma\,.\,\forall n \in \mathbb{N}\,.\, \sigma \in \alpha(\tau^{n\dot{+}1}\frown\Sigma^\omega),\; x \preccurlyeq \sigma \;[\;\text{ by Lemma 4.8 }\;] \\
\quad\Rightarrow\; \exists\sigma\,.\,\forall n \in \mathbb{N}\,.\, \exists\delta \in \tau^{n\dot{+}1}\frown\Sigma^\omega\,.\,\sigma = \delta_0^{k_0}\delta_1^{k_1}\ldots,\; x \preccurlyeq \sigma,\; k_i \in D_\alpha \\
\quad\Rightarrow\; \exists\sigma\,.\,\forall n \in \mathbb{N}\,.\, \exists\delta \in \Sigma^\omega\,.\,\varphi_\omega(\delta) \subseteq \varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+),\; \sigma = \delta_0^{k_0}\delta_1^{k_1}\ldots,\; x \preccurlyeq \sigma
\end{array}
$$

At this point we have to prove that the condition above, i.e., $\exists\sigma\,.\,\forall n \in \mathbb{N}\,.\,\exists\delta \in \Sigma^\omega\,.\,\varphi_\omega(\delta) \subseteq \varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+),\; \sigma = \delta_0^{k_0}\delta_1^{k_1}\ldots$ implies that we can build an infinite trace δ with the same properties and whose prefixes belong to $\varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+)$ for all n. First of all we can erase all the consecutive repetitions from σ, obtaining a minimal[1] (as number of states) trace σ' that generates σ by α: $\sigma = \sigma_0^{h_0}\sigma_1^{h_1}\ldots$ where $\forall i\,.\,h_i \neq 0$, and $\forall i\,.\,\sigma_i \neq \sigma_{i+1}$ by construction. If $|\sigma'| < \omega$, i.e., $\sigma' = \sigma_0\ldots\sigma_k$, then we consider $\sigma' \stackrel{\mathrm{def}}{=} \sigma_0\ldots\sigma_k\sigma_k\ldots$, namely we do not erase the repetitions of the last different state.
For each n consider the trace δ such that $\varphi_\omega(\delta) \subseteq \varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+)$ and $\sigma = \delta_0^{k_0}\delta_1^{k_1}\ldots$, which exists by hypothesis. This means that $\delta_0^{k_0}\delta_1^{k_1}\ldots = \sigma_0^{h_0}\sigma_1^{h_1}\ldots$. Since we are dealing with stuttering safety we have that $\forall i\,.\,k_i > 0$. This implies that δ and σ' contain the same states, only the number of their repetitions can change. Consider a prefix $x = \sigma_0\ldots\sigma_i$ of σ'. Let us prove by induction on the length of x that $\forall n\,.x \in \tau^{n\dot{+}1}\frown\Sigma^+$. If $|x| = 1$ then it must be $x = \sigma_0$. But any δ such that $\delta_0^{k_0}\delta_1^{k_1}\ldots = \sigma_0^{h_0}\sigma_1^{h_1}\ldots$ has $\delta_0 = \sigma_0$ therefore, since $\forall n\,.\,\exists\delta\,.\,\varphi_\omega(\delta) \subseteq \varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+)\,.\,\delta_0^{k_0}\ldots = \sigma_0^{h_0}\ldots$, then $\forall n\,.\,x = \delta_0 \in \varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+)$. Let $x = \sigma_0\ldots\sigma_i$, i.e., $|x| = i+1$, then $\sigma_0^{h_0}\ldots\sigma_i^{h_i} \preccurlyeq \sigma$, therefore, let $h = |\sigma_0^{h_0}\ldots\sigma_i^{h_i}|$, there must exist δ such that $\varphi_\omega(\delta) \subseteq \varphi_\omega(\tau^{\dot{h}}\frown\Sigma^+)$ such that $\sigma = \delta_0^{k_0}\delta_1^{k_1}\ldots$. Clearly this last hypothesis implies that $\exists j\,.\,i \leq j \leq h\,.\,\sigma_0^{h_0}\ldots\sigma_i^{h_i} = \delta_0^{k_0}\ldots\delta_j^{k'_j}$. Since $\delta_0\ldots\delta_j \in \varphi_\omega(\tau^{\dot{h}}\frown\Sigma^+)$, we have that $\delta_0\ldots\delta_j \in \tau^{\dot{h}}$. Namely $\forall l \leq j-1\,.\,(\delta_l,\delta_{l+1}) \in \tau$, which implies, due to the equality above, that $\forall l \leq i-1\,.\,(\sigma_l,\sigma_{l+1}) \in \tau$, namely $x \in \tau^{i\dot{+}1}$. In this way we proved that $\forall x \preccurlyeq \sigma'$ we have $\exists n\,.\,x \in \tau^{\dot{n}}$. Now let $|x| = m$, then $\forall n \leq m$ we have $x \in \tau^{\dot{n}}\frown\Sigma^+ \subseteq \varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+)$, while $\forall n > m$ we have that $x \preccurlyeq \sigma_0\ldots\sigma_n \in \tau^{n\dot{+}1}\frown\Sigma^+$ since $\sigma_0\ldots\sigma_n \in \tau^{n\dot{+}1}$, therefore $\forall n\,.\,\forall x \in \varphi_\omega(\sigma')\,.\,x \in \varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+)$. We proved in this way that $\forall n \in \mathbb{N}\,.\,\varphi_\omega(\sigma') \subseteq \varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+)$. Therefore we have the following implications:

$$
\begin{array}{l}
\exists\sigma\,.\,\forall n \in \mathbb{N}\,.\,\exists\delta \in \Sigma^\omega\,.\,\varphi_\omega(\delta) \subseteq \varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+),\; \sigma = \delta_0^{k_0}\delta_1^{k_1}\ldots,\; x \preccurlyeq \sigma \\
\quad\Rightarrow\; \exists\sigma,\delta\,.\,\forall n \in \mathbb{N}\,.\,\varphi_\omega(\delta) \subseteq \varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+)\,.\,\sigma = \delta_0^{k_0}\delta_1^{k_1}\ldots,\; x \preccurlyeq \sigma \\
\quad\Rightarrow\; \exists\sigma,\delta\,.\,\forall n \in \mathbb{N}\,.\,\delta \in \gamma_\omega\varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^+)\,.\,\sigma = \delta_0^{k_0}\delta_1^{k_1}\ldots,\; x \preccurlyeq \sigma \\
\quad\Rightarrow\; \exists\sigma,\delta\,.\,\delta \in \bigcap_{n\in\mathbb{N}} \gamma_\omega\varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^\omega),\; \sigma = \delta_0^{k_0}\delta_1^{k_1}\ldots,\; x \preccurlyeq \sigma \\
\quad\Rightarrow\; \exists\sigma,\delta\,.\,\varphi_\omega(\delta) \subseteq \varphi_\omega\bigcap_{n\in\mathbb{N}} \gamma_\omega\varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^\omega),\; \sigma = \delta_0^{k_0}\delta_1^{k_1}\ldots,\; x \preccurlyeq \sigma \\
\quad\Rightarrow\; \exists\sigma,\delta\,.\,\varphi_\omega(\sigma) \subseteq \alpha\varphi_\omega(\delta) \subseteq \alpha\varphi_\omega\bigcap_{n\in\mathbb{N}} \gamma_\omega\varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^\omega),\; x \preccurlyeq \sigma \\
\quad\Rightarrow\; x \in \varphi_\omega(\sigma) \subseteq \alpha\varphi_\omega\bigcap_{n\in\mathbb{N}} \gamma_\omega\varphi_\omega(\tau^{n\dot{+}1}\frown\Sigma^\omega) = \alpha(\prod^{\mathrm{safe}}_{n\in\mathbb{N}}(F^{\mathrm{safe}})^n(\Sigma^+))
\end{array}
$$

[1]Minimal here means that if we erase some other states then we cannot rebuild σ by using α.

■

Finally, next result proves that the abstract domain defined by α^{stu} is complete for the operator F^{safe} [11, 23], namely the commutative property holds. In the following the domain D_α for the values k_i is always $\mathbb{N} \smallsetminus \{0\}$.

LEMMA 4.10 *Let $\alpha = \alpha^{stu}$. Then $\alpha \circ F^{safe} = \alpha \circ F^{safe} \circ \alpha$.*

Proof By definition we have that $\alpha \circ F^{\mathrm{safe}} = \alpha \circ F^{\mathrm{safe}} \circ \alpha$ corresponds to the property $\forall X \in \mathcal{S}\,.\, \alpha\varphi_\omega(\tau^{\dot{2}} \frown X) = \alpha\varphi_\omega(\tau^{\dot{2}} \frown \alpha(X))$. Since $\alpha(X) \supseteq X$ and being all the involved functions monotone, we have the immediate inclusion $\alpha\varphi_\omega(\tau^{\dot{2}} \frown X) \subseteq \alpha\varphi_\omega(\tau^{\dot{2}} \frown \alpha(X))$.
Let us prove the other inclusion.

$$
\begin{array}{rl}
x \in \alpha\varphi_\omega(\tau^{\dot{2}} \frown \alpha(X)) \Rightarrow & \exists y = y_0 y_1 \dots y_m \in \varphi_\omega(\tau^{\dot{2}} \frown \alpha(X))\,.\, x = y_0^{k_0} y_1^{k_1} \dots y_m^{k_m} \\
& \text{for some } k_0, k_1, \dots, k_m \in \mathbb{N} \smallsetminus \{0\} \\
\Rightarrow & x = y_0^{k_0} y_1^{k_1} \dots y_m^{k_m}, \exists w \in \tau^{\dot{2}} \frown \alpha(X)\,.\, y \preccurlyeq w \\
\Rightarrow & x = y_0^{k_0} y_1^{k_1} \dots y_m^{k_m}, y \preccurlyeq w,\ w_0 \tau w_1,\ w' \stackrel{\mathrm{def}}{=} w_1 \dots w_m \in \alpha(X) \\
\Rightarrow & x = y_0^{k_0} y_1^{k_1} \dots y_m^{k_m},\ y \preccurlyeq w,\ \exists z_0 z_1 \dots z_l \in X\,. \\
& w' = z_0^{h_0} z_1^{h_1} \dots z_l^{h_l} \text{ for some } h_0, h_1, \dots, h_l \in \mathbb{N} \smallsetminus \{0\} \\
\Rightarrow & x = y_0^{k_0} y_1^{k_1} \dots y_m^{k_m},\ y \preccurlyeq w,\ w' = z_0^{h_0} z_1^{h_1} \dots z_l^{h_l}, \\
& y_0 z_0 \dots z_l \in \tau^{\dot{2}} \frown X \quad [\ \text{since } z_0 = w_1,\ y_0 = w_0 \text{ and } h_0 \neq 0\ ]
\end{array}
$$

At this point, note that $y_1 \dots y_m \preccurlyeq w' = z_0^{h_0} z_1^{h_1} \dots z_l^{h_l}$. This implies $y_1^{k_1} \dots y_m^{k_m} = z_0^{h'_0} z_1^{h'_1} \dots z_{l_1}^{h'_{l_1}}$, with $l_1 \leq l$. From the implications above we obtain the equality $x = y_0^{k_0} y_1^{k_1} \dots y_m^{k_m} = y_0^{k_0} z_0^{h'_0} z_1^{h'_1} \dots z_{l_1}^{h'_{l_1}}$ with $y_0 z_0 \dots z_{l_1} \in \varphi_\omega(\tau^{\dot{2}} \frown X)$ being prefix of $y_0 z_0 \dots z_l \in \tau^{\dot{2}} \frown X$. Namely $x \in \alpha(\varphi_\omega(\tau^{\dot{2}} \frown X))$. ■

Hence, we can transfer the fixpoint of the operator F^{safe} on the stuttering abstract domain in order to construct it systematically.

THEOREM 4.11 *Let $\alpha = \alpha^{stu}$, $X \in \alpha(\mathcal{S})$ and $F^\alpha(X) \stackrel{def}{=} \alpha\varphi_\omega(\tau^{\dot{2}} \frown X)$. Then*

$$\tau^\alpha = \alpha(gfp^{\subseteq}_{\Sigma^+} F^{safe}) = gfp^{\subseteq}_{\Sigma^+} F^\alpha.$$

Proof $F^\alpha \circ \alpha = \alpha \circ F^{\mathrm{safe}} \circ \alpha$ by definition of F^α, and $\alpha \circ F^{\mathrm{safe}} \circ \alpha = \alpha \circ F^{\mathrm{safe}}$ by Lemma 4.10. Then we have that $F^\alpha \circ \alpha = \alpha \circ F^{\mathrm{safe}}$. Then by using Prop. 4.9, we can apply the dual weakened Kleene transfer theorem and obtain the thesis. ■

In this section we showed that safety without stuttering, allowing to replicate states, preserves the constructive characterization proved for safety semantics. This characterization is important since it tells us that we can enforce also this restriction of safety monitoring the computation of programs. We also showed that the same does not hold whenever we consider cancellation, namely whenever we want to enforce properties where the deletion of states is admitted. In other words safety semantics allowing cancellation of states cannot be characterized in a constructive way.

5. Safety vs Liveness in abstract interpretation

In this section we want to exploit the abstract interpretation based characterization of safety with a different task. Our final aim is to prove that the complementary

nature of safety and liveness properties does not have a corresponding interpretation in the abstract interpretation framework. In fact, it is well-known, that in the standard approach to safety/liveness [3], liveness is in some way a "complementary" notion of safety in the sense that any interesting property is indeed the intersection of a safety property with a liveness one [3][Th.1]. What we would like to investigate is whether this "complementary" relation holds also in the abstract interpretation framework, namely we want to understand if the complementation of the safety domain, as abstraction, is a significant domain and whether it models liveness properties. Hence we have to follow the following steps: (i) we first have to characterize safety property by means of a closure operator; (ii) we have to prove that this closure precisely captures safety properties in the Alpern-Schneider approach to safety/liveness properties; (iii) we characterize the complement of this safety closure in the abstract-interpretation framework.

5.1 *The closure operator Safe*

Consider the pair of adjoint functions used in the previous sections for characterizing safety in the hierarchy of semantics. It is well-know that the composition of a pair of adjoint function forms a closure operator, in particular, the composition $Safe = \gamma_\omega \circ \varphi_\omega$ is an upper closure operator (see Sect. 2.2):

$$Safe(X) = \left\{ \sigma \in \Sigma^\omega \,\middle|\, \varphi_\omega(\sigma) \subseteq \varphi_\omega(X) \right\}$$

In the following of this section we use the Alpern and Schneider [3] characterization of safety and liveness properties in order to formally prove that this closure precisely captures safety properties and can be used for characterizing also liveness properties. Indeed, *Safe* captures exactly the intuitive characterization of safety properties since it completes a set X of infinite traces with all those traces whose partial executions are partial executions of traces in X, in this sense it is maximal with respect to a given set of partial executions, those of X. On the other hand, liveness properties are intuitively described as properties that admits every possible partial execution, in this case formally *Safe* would complete the property with all the missing infinite traces. Hence the idea is to show that safety properties are exactly those such that $Safe(X) = X$, while liveness properties are those such that $Safe(X) = \Sigma^\omega$.

5.2 *Safe for safety/liveness properties*

According to Alpern and Schneider [3], safety and liveness properties can be characterized by considering the standard *Cantor topology* on the set of infinite traces Σ^ω induced by the metric $d : \Sigma^\omega \times \Sigma^\omega \to \mathbb{R}$ defined as

$$d(\sigma, \delta) = \begin{cases} 0 & \text{if } \sigma = \delta \\ 2^{-n} & \text{if } n = min\{i \mid \sigma_i = \delta_i\} \end{cases}$$

In this case, safety properties have been proved to be the closed sets of the Cantor's topology, while the dense sets are the liveness properties on $\wp(\Sigma^\omega)$. Hence, if we prove that the closure *Safe* is a topological closure and that its closed elements are closed in the Cantor topology then we have done, since the topological structure guarantees that also the dense elements can be characterized by means of the topological closure, i.e., $Safe(X) = \Sigma^\omega$.

Safe *is a topological closure.* Note that, the following properties are intuitively quite trivial for a Cantor's topology. Nevertheless, we provide a detailed proof in order to show, in sake of readability, how the closure *Safe* works.

LEMMA 5.1

(1) The closure operator Safe is finitely additive;
(2) The closure operator Safe is not continuous;
(3) The closure operator Safe is not co-continuous.

Proof

(1) First of all we prove that if we take two sets X and Y in $\wp(\Sigma^\omega)$ then $\mathit{Safe}(X \cup Y) = \mathit{Safe}(X) \cup \mathit{Safe}(Y)$: By definition we have that

$$\begin{aligned}\mathit{Safe}(X \cup Y) &= \left\{ \sigma \in \Sigma^\omega \,\middle|\, \varphi_\omega(\sigma) \subseteq \varphi_\omega(X \cup Y) \right\} \\ &= \left\{ \sigma \in \Sigma^\omega \,\middle|\, \varphi_\omega(\sigma) \subseteq \varphi_\omega(X) \cup \varphi_\omega(Y) \right\}\end{aligned}$$

We prove now that if $\varphi_\omega(\sigma) \subseteq \varphi_\omega(X) \cup \varphi_\omega(Y)$ then $\varphi_\omega(\sigma) \subseteq \varphi_\omega(X)$ or $\varphi_\omega(\sigma) \subseteq \varphi_\omega(Y)$. Suppose that $\varphi_\omega(\sigma) \cap \varphi_\omega(X) \neq \varnothing$ and that $\varphi_\omega(\sigma) \not\subseteq \varphi_\omega(X)$, then we have $\varnothing \neq \varphi_\omega(\sigma) \setminus \varphi_\omega(X) \subseteq \varphi_\omega(Y)$. For the first inequality we can say that $\exists x' \in \Sigma^+ \,.\, x' \preccurlyeq \sigma$, $x' \notin \varphi_\omega(X)$ and $x' \in \varphi_\omega(Y)$, since the difference operation doesn't erase x'. Moreover $\forall x \in \Sigma^+ \,.\, x'x \preccurlyeq \sigma \;\Rightarrow\; x'x \in \varphi_\omega(Y)$ for the same reason, and being the sets closed under prefix.
Hence the infinite traces in Y which have x' as prefix surely have as prefix also each prefix of x', then $\varphi_\omega(\sigma) \subseteq \varphi_\omega(Y)$. Therefore

$$\begin{aligned}&\left\{ \sigma \,\middle|\, \varphi_\omega(\sigma) \subseteq \varphi_\omega(X) \cup \varphi_\omega(Y) \right\} \\ &\qquad = \left\{ \sigma \,\middle|\, \varphi_\omega(\sigma) \subseteq \varphi_\omega(X) \right\} \cup \left\{ \sigma \,\middle|\, \varphi_\omega(\sigma) \subseteq \varphi_\omega(Y) \right\} \\ &\qquad = \mathit{Safe}(X) \cup \mathit{Safe}(Y)\end{aligned}$$

(2) We prove that the closure is not continuous by showing an example where the continuity fails. Consider the increasing chain $\{X_n\}_{n\in\mathbb{N}} \subseteq \wp(\Sigma^\omega)$, where $\forall n \in \mathbb{N} \,.\, X_n \stackrel{\text{def}}{=} \{b^\omega\} \cup \{a^i b^\omega \mid i \leq n\}$. It is worth noting that $\bigcup_n X_n = \{b^\omega\} \cup \{a^n b^\omega \mid n \in \mathbb{N}\}$. Therefore we have that $\varphi_\omega(\bigcup_n X_n) = \{b^i \mid i \in \mathbb{N}\} \cup \{a^i \mid i \in \mathbb{N}\} \cup \{a^i b^j \mid i, j \in \mathbb{N}\}$. Finally we can find that $\gamma_\omega\varphi_\omega(\bigcup_n X_n) = \{a^\omega, b^\omega\} \cup \{a^n b^\omega \mid n \in \mathbb{N}\}$.
On the other hand we have that for each $n \in \mathbb{N}$, $\varphi_\omega(X_n) = \{b^i \mid i \in \mathbb{N}\} \cup \{a^i \mid i \leq n\} \cup \{a^i b^j \mid i \leq n,\ j \in \mathbb{N}\}$. Therefore $\gamma_\omega\varphi_\omega(X_n) = \{b^\omega\} \cup \{a^i b^\omega \mid i \leq n\}$. Clearly we have that $a^\omega \notin \bigcup_n \gamma_\omega\varphi_\omega(X_n)$.

(3) Finally we can show that *Safe* is not co-continuous since we can find an example where co-continuity fails. Consider the decreasing chain $\{X_n\}_{n\in\mathbb{N}} \subseteq \wp(\Sigma^\omega)$, defined as follows: $\forall n \in \mathbb{N} \,.\, X_n \stackrel{\text{def}}{=} \{\sigma \mid a^n \preccurlyeq \sigma,\ b \in \sigma\}$. The only infinite trace σ that for each n has a^n as prefix is $\sigma = a^\omega$, but σ does not contain b, therefore $\bigcap_n X_n = \varnothing$. On the other hand for each n we have $\varphi_\omega(X_n) \supseteq \{a^i \mid i \in \mathbb{N}\}$ since for all $i \leq n$ we have that $a^i \preccurlyeq a^n$, while for all $i > n$ we have that $a^n \preccurlyeq a^i \preccurlyeq a^i b^\omega \in X_n$. Therefore $\forall n \in \mathbb{N} \,.\, a^\omega \in \gamma_\omega\varphi_\omega(X_n)$, which implies that $a^\omega \in \bigcap_n \gamma_\omega\varphi_\omega(X_n)$. We proved in this way that $\mathit{Safe} = \gamma_\omega \circ \varphi_\omega$ is not co-continuous since $\gamma_\omega\varphi_\omega(\bigcap_n X_n) = \varnothing$.

■

Note that the lemma above implies also that the function φ_ω is not co-continuous. It is immediate to prove the following result.

PROPOSITION 5.2 *Safe is a topological closure*

Proof *Safe* is an upper closure operator by construction. Moreover by Lemma 5.1, it is finitely additive and $Safe(\varnothing) = \varnothing$. This makes *Safe* a topological (Kuratowski) closure. ■

Safe *characterization of safety and liveness properties.* Note that (Σ^ω, d) is a complete metric space, namely every Cauchy sequence in Σ^ω has a limit. Recall that a sequence $\{\sigma_n\}$ in a metric space (U, d) is Cauchy provided that:

$$\forall \epsilon > 0\ \exists k.\ \forall n, m \geq k.\ d(\sigma_n, \sigma_m) \leq \epsilon$$

and that its limit, when it exists, is denoted as $\lim_{n\to\infty} \sigma_n$ and it is the (unique) σ such that

$$\forall \epsilon > 0\ \exists k.\ \forall n \geq k.\ d(\sigma_n, \sigma) \leq \epsilon$$

Let $X \subseteq \Sigma^+$ be a set of finite traces. We denote by $X^{\uparrow n}$ the set of traces in X of length n. Then, in our case, a sequence $\{\sigma_n\}$ of infinite traces is Cauchy if for every $\epsilon > 0$ there exists $k = -\lfloor \log \epsilon \rfloor$ such that for every $n, m \geq k$ $\varphi_\omega(\sigma_n)^{\uparrow k} = \varphi_\omega(\sigma_m)^{\uparrow k}$. (Σ^ω, d) is therefore clearly complete because it contains all infinite traces. It is known [39] that a set $X \subseteq U$ is closed in the metric topology induced by the complete metric space (U, d) if and only if the limit of any Cauchy sequence of points in X is contained in X.

LEMMA 5.3 *$X = Safe(X)$ iff it is closed in the Cantor topology on Σ^ω.*

Proof In order to prove this result we have only to prove that for any $X \subseteq \Sigma^\omega$, $\sigma \in \gamma_\omega\varphi_\omega(X)$ iff there exists a Cauchy sequence $\{x^n\}_{n\in\mathbb{N}} \subseteq \gamma_\omega\varphi_\omega(X)$ of infinite traces such that $\lim_{n\to\infty} x^n = \sigma$.
($\Rightarrow$.) Let $\sigma \in \gamma_\omega\varphi_\omega(X)$. This holds iff $\varphi_\omega(\sigma) \subseteq \varphi_\omega(X)$. We consider the sequence of traces $\{x^n\}_{n\in\mathbb{N}}$ such that $x_n = y\eta$ with $y \in \varphi_\omega(\sigma)^{\uparrow n}$ and $y\eta \in X \subseteq \gamma_\omega\varphi_\omega(X)$. These objects exist because any finite prefix y of σ is a finite prefix of some infinite trace $y\eta$ in X. This sequence is clearly Cauchy by definition and $\lim_{n\to\infty} x^n = \sigma$.
($\Leftarrow$.) Let $\{x^n\}_{n\in\mathbb{N}}$ be a Cauchy sequence in $\gamma_\omega\varphi_\omega(X)$ such that $\lim_{n\to\infty} x^n = \sigma$. We prove that $\sigma \in \gamma_\omega\varphi_\omega(X)$. From what we observed above, and the definition of limits of Cauchy sequences, for any $m \geq 0$, there exists k such that $\varphi_\omega(\sigma)^{\uparrow m} = \varphi_\omega(x^k)^{\uparrow m}$. Therefore

$$\varphi_\omega(\sigma) = \bigcup_{m<\omega} \varphi_\omega(\sigma)^{\uparrow m} \subseteq \bigcup_{k<\omega} \bigcup_{m<\omega} \varphi_\omega(x^k)^{\uparrow m} \subseteq \varphi_\omega(X)$$

Then, we have that $\varphi_\omega(\sigma) \subseteq \varphi_\omega(X)$ which implies that $\sigma \in \gamma_\omega\varphi_\omega(X)$. ■

Hence, due to the Alpern and Schneider topological characterization of safety and liveness properties [3], we have the following characterization of these properties by means of the closure *Safe*.

THEOREM 5.4 *Given $X \in \wp(\Sigma^\omega)$, property on infinite traces*

- *X is a safety property iff $Safe(X) = X$;*
- *X is a liveness property iff $Safe(X) = \Sigma^\omega$.*

5.3 *Complementing Safe*

It is clear from the previous construction and from Gumm's characterization of safety and liveness [25], that safety properties are abstractions of infinite traces.

In this sense, the safety semantics can be considered as an abstract interpretation of the infinite trace semantics in Cousot's hierarchy (see Fig. 1). This abstraction allows also to provide a characterization of liveness properties in terms of *Safe* as we have seen before, i.e., X is liveness iff $\mathit{Safe}(X) = \Sigma^\omega$. However, we do not have a characterization of liveness properties by means of an abstraction whose closed elements are indeed liveness properties.
With this aim in mind we study the structure of meet-irreducible elements, i.e., those sets which cannot be obtained by intersection. Indeed, the importance in this investigation is twofold: (i) the complementation in abstract interpretation is based on these elements, as we will see later on; (ii) Closure operators on $\wp(S)$, with S being any complete lattice, are traditionally specified in terms of their meet-irreducible elements [7]. This is justified by the fact that closure operators are Moore families. In fact, for complete lattices generated by their meet-irreducible elements, like algebraic complete lattices, meet-irreducibles specify the least (often irredundant) set from which the whole lattice can be generated. Unfortunately, this is not the case for the closed sets of Cantor topology on Σ^ω, i.e., for the elements in *Safe*. Namely, in our context, for each element X such that $X = \mathit{Safe}(X)$ we can find two other different elements in *Safe* whose *glb* is equal to X. This implies that *Safe* does not have meet-irreducible elements. This fact, itself, is quite unusual in the abstract interpretation framework, but what makes *Safe* even more interesting, is the fact that we can anyway characterize a subset Δ of closed elements, which is order generating for *Safe*. In other words, each closed *Safe* element is meet generated by elements in Δ.

The algebraic structure of safety properties. In this section we want to characterize the algebraic structure of the domain *Safe*. For this reason, we have to investigate about the existence of its meet-irreducible elements (*Mirr*(*Safe*) for short), which are the elements closest to the top of the lattice. In order to understand the following results we have to underline some aspects about meet-irreducible elements. We recall that (see Sect 2.1) a meet-irreducible set X is different from the top, i.e., Σ^ω, and cannot be obtained as intersection of sets different from itself, i.e., if $X = X_1 \cap X_2$ then $X = X_1$ or $X = X_2$. On the other hand, note that, given a metric space X, any closed $C \subset X$ can be obtained as the intersection of the closed sets $C \cup \{x_1\}$ and $C \cup \{x_2\}$, with $x_1, x_2 \notin C$. In general, this means that each element in *Safe* can be obtained as intersection of other two sets in *Safe*. Even if this allows to say that *Safe* has not meet-irreducible elements, it is not sufficient for *constructively* characterizing whether *Safe* is, anyway, order generated. Hence, our aim is to understand which are the closed sets just below the top, and to characterize the structure of the elements that can generate the whole domain of closed elements of *Safe*.
Clearly, the following study is based on the fact that *Safe* is an abstraction of the infinite trace semantics in the Cousot's hierarchy of semantics [10]. Moreover, in the following, any element in *Safe* is called a *safety set*. At this point, before entering in the construction, it is worth noting that $\forall \delta \in \Sigma^\omega \,.\, \mathit{Safe}(\{\delta\}) = \{\delta\}$, since, being Σ^ω a metric space, all the singletons $\{\delta\}$ are closed in the Cantor topology.
Now, let us start defining the following sets. Given $x \in \Sigma^+$, we define

$$\Lambda_x \stackrel{\text{def}}{=} \left\{ \; \delta \in \Sigma^\omega \;\middle|\; x \not\preceq \delta \; \right\}$$

This is the set of all the traces δ such that x is not prefix of δ. In other words, the only traces that are not in Λ_x are all the possible infinite extensions of x. We use

these sets for defining the following subset of $\wp(\Sigma^\omega)$:

$$\Delta = \left\{ \Lambda_x \in \wp(\Sigma^\omega) \,\middle|\, x \in \Sigma^+ \right\}$$

Δ collects all the set maximal with respect to all the possible finite prefixes but one, namely Λ_x is maximal with respect to the set of partial executions $\Sigma^+ \setminus \{x\}$. For this reason, intuitively, they are the safety properties "closest" to the top. These elements, like meet-irreducible, contains all the information necessary for meet generating each safety property. Nevertheless, they cannot be meet-irreducible since they can also be generated by other different elements in Δ, as we show in the following results.

LEMMA 5.5 *$\forall X \in \Delta$. $Safe(X) = X$ and X is not meet-irreducible.*

Proof Let $X = \Lambda_x$. Then we have:

$$\begin{aligned} Safe(X) &= Safe(\Lambda_x) \\ &= Safe(\left\{ \delta \,\middle|\, x \not\preccurlyeq \delta \right\}) \\ &= \left\{ \delta \in \Sigma^\omega \,\middle|\, \varphi_\omega(\delta) \subseteq \varphi_\omega(\{\delta \mid x \not\preccurlyeq \delta\}) \right\} \\ &= \left\{ \delta \,\middle|\, x \not\preccurlyeq \delta \right\} \\ &= \Lambda_x = X \end{aligned}$$

Clearly, these elements cannot be meet-irreducible since they are closed of the Cantor topology, in the metric space Σ^ω. ■

COROLLARY 5.6 *$Mirr(Safe) = \varnothing$.*

Now we can prove that the abstract domain of *Safe* is order generated by Δ, namely we can show that each closed element can be obtained as intersection of elements in Δ. This means, that Δ is all we need for describing the closed elements in *Safe*.

PROPOSITION 5.7 *$\Delta \subseteq$ Safe is order generating for Safe.*

Proof We prove that each element X in *Safe* can be obtained as intersection of elements in Δ. Consider $X \in Safe$, then

$$\begin{aligned} \delta \in X &\Leftrightarrow \varphi_\omega(\delta) \subseteq \varphi_\omega(X) \\ &\Leftrightarrow \forall x \in \Sigma^+ . (x \in \varphi_\omega(\delta) \;\Rightarrow\; x \in \varphi_\omega(X)) \\ &\Leftrightarrow \forall x \in \Sigma^+ . (x \notin \varphi_\omega(X) \;\Rightarrow\; x \notin \varphi_\omega(\delta) \\ &\Leftrightarrow \forall x \in \Sigma^+ . (x \notin \varphi_\omega(X) \;\Rightarrow\; x \not\preccurlyeq \delta) \\ &\Leftrightarrow \forall x \in \Sigma^+ . (x \notin \varphi_\omega(X) \;\Rightarrow\; \delta \in \Lambda_x) \\ &\Leftrightarrow \delta \in \bigcap \left\{ \Lambda_x \,\middle|\, x \notin \varphi_\omega(X) \right\} \end{aligned}$$

■

The proposition above says that in order to obtain a safety set it is necessary to cancel from the top Σ^ω an infinite number of traces. This because we are unable to rebuild the missing traces simply by looking at the prefixes of the traces in the set, in other words the set $Mirr(Safe)$ is not *order generating* [24].

A first characterization of liveness properties as sets of infinite traces can be obtained by analyzing the structure of the elements in Δ. Indeed, we can use these elements for understanding the sets representing liveness properties. We noticed, in fact, that the elements in Δ are in *Safe* since they lack an infinite amount of traces. We can note that if, instead, we cut off from the top Σ^ω a finite number of traces then we obtain liveness properties, since all their prefixes are prefixes of other remaining traces of the set.

PROPOSITION 5.8 *Consider $X \in \wp(\Sigma^\omega)$ such that X has finite cardinality, i.e., $|X| \in \mathbb{N}$, then $\Sigma^\omega \smallsetminus X$ is liveness.*

Proof Consider $Y = \Sigma^\omega \smallsetminus X$. We have first to prove that $X \subseteq \mathit{Safe}(Y)$. Namely we have to prove that $\delta \in X$ implies $\delta \in \mathit{Safe}(Y)$. By definition of *Safe* this holds if $\forall \delta \in X$ we have $\varphi_\omega(\delta) \subseteq \varphi_\omega(Y)$. Consider $x \in \varphi_\omega(\delta)$, then we can always build an infinite trace α such that $\forall \delta \in X\,.\, x\alpha \neq \delta$, since the δ are finite in number. This implies that, for each $x \in \varphi_\omega(\delta)$ we have $x\alpha \in Y$, therefore $x \in \varphi_\omega(Y)$. Hence we proved that $X \subseteq \mathit{Safe}(Y)$, on the other hand clearly we have that $Y \subseteq \mathit{Safe}(Y)$, and therefore $\Sigma^\omega = Y \cup X \subseteq \mathit{Safe}(Y)$. This, finally, means that $\mathit{Safe}(Y) = \Sigma^\omega$, being $\mathit{Safe}(Y) \subseteq \Sigma^\omega$. ■

***Complementing* Safe.** In the following we consider the complement operation defined in [9, 19] as a systematic method to compare abstract domains. Abstract domain complementation introduced in [9] provides a systematic method for decomposing abstract domains. Complementation is the *inverse* operation of the reduced product (see [22]), namely an operation which, starting from any two domains $C \sqsubseteq D$, 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* for 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 $x \in L$, if it exists, is the unique element $x^* \in L$ such that $x \wedge x^* = \bot$ and $\forall y \in L.\ (x \wedge y = \bot) \Rightarrow (y \leq x^*)$ [7]. 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 for abstract interpretation. Indeed, it is well-known [18, 31] 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 [19, 21] provide sufficient conditions on C such that $\mathit{uco}(C)$ is pseudo-complemented. Moreover C is meet-generated by $S \subseteq C$ if $C = \mathcal{M}(S)$.

THEOREM 5.9 *Let C be a complete lattice.*

(1) If C is a meet-continuous then $\mathit{uco}(C)$ is pseudo-complemented [21].
(2) If C is meet-generated by $\mathit{Mirr}(C)$ then $\mathit{uco}(C)$ is pseudo-complemented and, for any $A \in \mathit{uco}(C)$, we have $A^* \stackrel{def}{=} C \ominus A = \mathcal{M}(\mathit{Mirr}(C) \smallsetminus A)$ [19].

By this theorem, we have that Σ^ω is pseudo-complemented, and trivially meet-generated by its meet-irreducible elements, hence we can think of characterizing the complement of *Safe* on the infinite trace semantic domain. Note that X is meet-irreducible in $\wp(\Sigma^\omega)$ if and only if $\exists \sigma \in \Sigma^\omega$ such that $X = \Sigma^\omega \smallsetminus \{\sigma\}$. It is worth noting that this fact, together with Prop. 5.8, implies that if X is a meet-irreducible element of $\wp(\Sigma^\omega)$ then $\mathit{Safe}(X) = \Sigma^\omega$, i.e., X is liveness.

COROLLARY 5.10 *Let $\mathit{Inf} = \gamma_\omega \circ \alpha_\omega$ (see Table 1) be the infinite semantics, then*

$$\mathit{Inf} \ominus \mathit{Safe} = \mathit{Inf}$$

The interpretation of this result is that, from an algebraic point of view, liveness is not the complement information of safety, since safety as closure has no complement in the set of infinite traces.

6. Conclusions

In this paper we have studied the lattice-theoretical structure of safety semantics in terms of the abstract interpretation of a maximal trace semantics of a transition system. This allows us to prove some properties of the safety semantics as properties of the corresponding abstraction on infinite traces. In particular we proved that the safety abstraction is complete in the sense of abstract interpretation with respect to the fixpoint semantics operator characterizing infinite computations. This construction provides a complete characterization of the safety semantics and of some of its abstractions such as stuttering and strong safety in the Cousot's hierarchy. The whole resulting picture, including Cousot's standard hierarchy and the new observable of safety properties, is depicted in Fig. 1. Further abstractions of safety can be derived by abstract interpretation of τ^{safe}. In particular it is possible to reinterpret the Alpern and Schneider [3] result by isolating the safety component of any property π in the lattice of abstract interpretations simply by considering $\pi \sqcup \mathit{Safe}$, which is the common abstraction between π and safety. Further research directions are towards the inclusion of security properties in Cousot's hierarchy of semantics. In particular in [35] the author proves that the only enforceable security policies are those representing safety properties. By enforceable we mean that there exists a mechanism that works by monitoring execution steps of a program and terminating the executions that are about to violate the security policy being enforced.

Acknowledgments

We thanks the anonymous referees for the very helpful comments on the previous versions of this paper. We want also to thank Patrick Cousot for the useful discussions made about the ideas developed in this paper.

References

[1] S. Abramsky and A. Jung, *Domain theory*, in *Handbook of Logic in Computer Science*, S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, eds., vol. 3, Oxford University Press, Inc., 1994, pp. 1–168.
[2] B. Alpern, A.J. Demers, and F.B. Schneider, *Safety without stuttering*, Information Processing Letters 23 (1986), pp. 177–180.
[3] B. Alpern and F.B. Schneider, *Defining liveness*, Information Processing Letters 21 (1985), pp. 181–185.
[4] ———, *Recognizing safety and liveness*, Distributed Comp. 2 (1987), pp. 117–126.
[5] K.R. Apt and G.D. Plotkin, *Countable nondeterminism and random assignment*, J. of the ACM 33 (1986), pp. 724–767.
[6] C. Baier and M. Kwiatkowska, *On topological hierarchies of temporal properties*, Fundamenta Informaticae 41 (2000), pp. 259–294.
[7] G. Birkhoff, *Lattice Theory*, AMS Colloquium Publication, 3rd edition, AMS (1967).
[8] E. Chang, Z. Manna, and A. Pnueli, *Characterization of temporal property classes*, in *Proc. of the Internat. Colloq. on Automata, Languages and Programming* (*ICALP '92*), *Lecture Notes in Computer Science*, vol. 623, Springer-Verlag, 1992, pp. 474–486.
[9] A. Cortesi, G. Filé, R. Giacobazzi, C. Palamidessi, and F. Ranzato, *Complementation in abstract interpretation*, ACM Trans. Program. Lang. Syst. 19 (1997), pp. 7–47.
[10] P. Cousot, *Constructive design of a hierarchy of semantics of a transition system by abstract interpretation*, Theor. Comput. Sci. 277 (2002), pp. 47–103.
[11] P. Cousot and R. Cousot, *Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints*, in *Proc. of Conf. Record of the 4th ACM Symp. on Principles of Programming Languages* (*POPL '77*), ACM Press, New York, 1977, pp. 238–252.
[12] ———, *Constructive versions of Tarski's fixed point theorems*, Pacific J. Math. 82 (1979), pp. 43–57.
[13] ———, *Systematic design of program analysis frameworks*, in *Proc. of Conf. Record of the 6th ACM Symp. on Principles of Programming Languages* (*POPL '79*), ACM Press, New York, 1979, pp. 269–282.
[14] ———, *Inductive definitions, semantics and abstract interpretation*, in *Proc. of Conf. Record of the 19th ACM Symp. on Principles of Programming Languages* (*POPL '92*), ACM Press, New York, 1992, pp. 83–94.

[15] J. de Bakker, *Mathematical theory of program correctness*, Prentice-Hall International (1980).
[16] E. Dijkstra, *Guarded commands, nondeterminism and formal derivation of programs*, Comm. of The ACM 18 (1975), pp. 453–457.
[17] E.W. Dijkstra, *A discipline of programming*, Series in automatic computation, Prentice-Hall (1976).
[18] P. Dwinger, *On the closure operators of a complete lattice*, Indagat. Math. 16 (1954), pp. 560–563.
[19] G. Filé and F. Ranzato, *Complementation of abstract domains made easy*, in *Proc. of the 1996 Joint Internat. Conf. and Symp. on Logic Programming* (*JICSLP '96*), The MIT Press, Cambridge, Mass., 1996, pp. 348–362.
[20] P.W. Fong, *Access Control By Tracking Shallow Execution History*, in *IEEE Symposium on Security and Privacy*, 2004, pp. 43 – 55.
[21] R. Giacobazzi, C. Palamidessi, and F. Ranzato, *Weak relative pseudo-complements of closure operators*, Algebra Universalis 36 (1996), pp. 405–412.
[22] R. Giacobazzi and F. Ranzato, *Refining and compressing abstract domains*, in *Proc. of the 24th Internat. Colloq. on Automata, Languages and Programming* (*ICALP '97*), *Lecture Notes in Computer Science*, vol. 1256, Springer-Verlag, Berlin, 1997, pp. 771–781.
[23] R. Giacobazzi, F. Ranzato, and F. Scozzari, *Making abstract interpretations complete*, J. of the ACM. 47 (2000), pp. 361–416.
[24] G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, and D.S. Scott, *A Compendium of Continuous Lattices*, Springer-Verlag (1980).
[25] H.P. Gumm, *Another glance at the Alpern-Schneider theorem*, Information Processing Letters 47 (1993), pp. 291–294.
[26] K.W. Hamlen, G. Morrisett, and F.B. Schneider, *Computability classes for enforcement mechanisms*, ACM Trans. on Programming Languages and Systems 28 (2006), pp. 175 – 205.
[27] C. Hoare, *An axiomatic basis for computer programming*, Comm. of The ACM 12 (1969), pp. 576–580.
[28] L. Lamport, *Proving correctness of multiprocess programs*, IEEE Trans. on Software Eng. 3 (1977), pp. 125–143.
[29] ———, *The temporal logic of actions*, ACM Trans. on Programming Languages and Systems 16 (1994), pp. 872–923.
[30] J. Ligatti, L. Bauer, and D. Walker, *Enforcing Non-safety Security Policies with Program Monitors*, in *10th European Symposium on Research in Computer Security (ESORICS)*, *Lecture Notes in Computer Science*, vol. 3679, Springer-Verlag, 2005, pp. 355 – 373.
[31] J. Morgado, *Note on complemented closure operators of complete lattices*, Portug. Math. 21 (1962), pp. 135–142.
[32] S. Owiki and L. Lamport, *Proving liveness properties of concurrent programs*, ACM Trans. Program. Lang. Syst. 4 (1982), pp. 455–495.
[33] D.O. Paun, *Closure under stuttering in temporal formulas* (1999).
[34] G. Plotkin, *A structural approach to operational semantics*, DAIMI-19 Aarhus University, Denmark (1981).
[35] F.B. Schneider, *Enforceable security policies*, Information and System Security 3 (2000), pp. 30–50.
[36] Z. Shmuely, *The structure of Galois connections*, Pacific J. Math. 54 (1974), pp. 209–225.
[37] A.P. Sistla, *Safety, liveness and fairness in temporal logic*, URL `citeseer.nj.nec.com/prasadsistla99safety.html`.
[38] ———, *On Characterization of Safety and Liveness Properties in Temporal Logic*, in *Proc. of the 4th ACM Symp. on Principles of Distributed Computing*, ACM Press, New York, 1985.
[39] M.B. Smyth, *Topology*, in *Handbook of logic in computer science (vol. 1): background: mathematical structures*, vol. 1, Oxford University Press, Inc., 1992, pp. 641–761.
[40] W. Thomas, *Safety and liveness properties in propositional temporal logic: Characterization and decidability*, Schriften Zur Informatik 116 (1986).