Partial Completeness of Abstract Fixpoint Checking
(Invited paper)

Patrick Cousot

D partement d'informatique, cole normale sup rieure
45 rue d'Ulm, 75230 Paris cedex 05, France

Patrick.Cousot@ens.fr http://www.ens.fr/~cousot/

Abstract. Abstract interpretation is used in program static analysis and model checking to cope with infinite state spaces and/or with computer resource limitations. One common problem is to check abstract fixpoints for specifications. The abstraction is *partially complete* when the checking algorithm is exact in that, if the algorithm ever terminates, its answer is always affirmative for correct specifications. We characterize partially complete abstractions for various abstract fixpoint checking algorithms, including new ones, and show that the computation of complete abstract domains is essentially equivalent to invariance proofs that is to concrete fixpoint checking.

1 Introduction

In computer assisted program formal verification, program static analysis and model-checking, one must design algorithms to check fixpoints $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ [1,2]. For theoretical undecidability reasons or because of practical computer resource limitations, one must often resort to abstract interpretation [6, 10, 12] and check instead $\gamma\left(\mathit{lfp}^{\sqsubseteq} \boldsymbol{\lambda} X \cdot \alpha(I \vee F(\gamma(X)))\right) \leq S$. *Soundness* requires that a positive abstract answer implies a positive concrete answer. So no error is possible when reasoning in the abstract. *Completeness* requires that a positive concrete answer can always be found in the abstract. Since termination is a separate problem in the abstract (which can be solved by other means such as a coarser abstraction and/or widenings/narrowings), we consider *partial completeness* [3] requiring that in case of termination of the abstract fixpoint checking algorithms, no positive answer can be missed. The problem that we study in this paper is "*to constructively characterize the abstractions* $\langle\alpha, \gamma\rangle$ *for which abstract*

[1] The $\leq$-least fixpoint $\mathit{lfp}^{\leq} \varphi$ is the $\leq$-least fixpoint of φ, if it exists, which is the case e.g. by Knaster-Tarski fixpoint theorem [34].

[2] We use Church's $\boldsymbol{\lambda}$-notation such that if $\varphi \triangleq \boldsymbol{\lambda} x \cdot e$ if the value of $\varphi(y)$ is that of e where the value of x is y.

[3] The phrasing recalls that of *partial correctness* after [21].

fixpoint algorithms are partially complete". This highlights the problems related to the generalization of model-checking to infinite (or very large) state systems and the approximation ideas which are recurrent in static program analysis by abstract interpretation.

2 Concrete Fixpoint Checking

2.1 The Concrete Fixpoint Checking Problem

Program static analysis, as formalized by abstract interpretation [6, 10], consists in automatically determining program properties by a fixpoint computation and then in checking that the computed program properties imply a specification given by the programming language semantics. Universal model-checking [3, 32] consists in checking that a model of a system satisfies a specification given by some temporal logic formula.

From a mathematical point of view, the principle is, in both cases, that we are given a complete lattice $\langle L, \leq, 0, 1, \vee, \wedge\rangle$ of properties and a transformer $F \in L \overset{\text{mon}}{\longmapsto} L$ which is a $\leq$-monotonic mapping from L to L. One must check that $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ where $\langle I, S\rangle \in L^2$ is the given specification [4].

Example 1. It is quite frequent in abstract interpretation [6, 12], to specify the program semantics by a transition system $\langle \Sigma, \tau, I\rangle$ where Σ is a set of states, $\tau \subseteq \Sigma \times \Sigma$ is the transition relation and $I \subseteq \Sigma$ is the set of initial states. The *collecting semantics* is the set $\mathit{post}[\tau^\star](I) = \mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee \mathit{post}[\tau](X)$ of states which are reachable from the initial states in I (where $\mathit{post}[\tau](X) \triangleq \{s' \mid \exists s \in X : \langle s, s'\rangle \in \tau\}$ is the right-image of $X \subseteq \Sigma$ by relation τ and $\tau^\star$ is the reflexive transitive closure of τ). Let $S \subseteq \Sigma$ be a safety specification (typically the specification of absence of run-time errors). The safety specification S is satisfied if and only if $\mathit{post}[\tau^\star](I) \subseteq S$ that is $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ where $F = \mathit{post}[\tau]$ and $\langle L, \leq, 0, 1, \vee, \wedge\rangle$ is $\langle \wp(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap\rangle$. □

2.2 The Concrete Fixpoint Checking Algorithm

The hypotheses for the Knaster-Kleene-Tarski fixpoint theorem [11, 34] are:

Hypotheses 1 *1. $\langle L, \leq, 0, 1, \vee, \wedge\rangle$ is a complete lattice;*
2. $F \in L \overset{mon}{\longmapsto} L$ is $\leq$-monotonic.

This theorem leads to the following iterative Alg. 1 to check that $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$. This Alg. 1 is classical in abstract interpretation [10] and apparently more recent in model-checking [16, 26] [5]:

[4] The $\leq$-least fixpoint $\mathit{lfp}^{\leq} \varphi$ of φ exists by Knaster-Tarski fixpoint theorem [34]. The same way, $\mathit{gfp}^{\leq} \varphi$ is the $\leq$-greatest fixpoint of φ, if it exists.

[5] In the programming language, the logical disjunction is denoted &, the conjunction is | and the negation is ¬.

Algorithm 1

$$
\begin{array}{l}
X := I;\ \ Go := (X \leq S);\\
\textbf{while}\ Go\ \textbf{do}\\
\quad X' := I \vee F(X);\\
\quad Go := (X \neq X')\ \&\ (X' \leq S);\\
\quad X := X';\\
\textbf{od};\\
\textbf{return}\ (X \leq S);
\end{array}
$$

In the general context of program analysis, this algorithm does not terminate for the collecting semantics defining the program properties but it can be used whenever e.g. L satisfies the ascending chain condition which is the common case in finite-state model-checking [3, 32] (Σ is finite).

Theorem 2. *Under Hyp. 1, Alg. 1 is partially correct* [6]*: when terminating, it returns* $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$.

Proof. We have $I \leq I \vee F(I)$ so, as shown in [11], the transfinite sequence $X^0 \triangleq I$, $X^{\delta+1} \triangleq I \vee F(X^\delta)$ for all successor ordinals $\delta \in \mathbb{O}$ and $X^\lambda \triangleq \bigvee_{\delta<\lambda} X^\delta$ for all limit ordinals $\lambda \in \omega.\mathbb{O}$ is increasing and ultimately stationary, its limit being the least fixpoint of $\boldsymbol{\lambda} X \cdot I \vee F(X)$ greater than I, that is the least fixpoint of $\boldsymbol{\lambda} X \cdot I \vee F(X)$. By recurrence, X^n, $n \in \mathbb{O}$ is the value of the program variable X at the end of the n-th iteration in the loop, if any, with $X^0 = I$ being the initial value of X upon entry of the loop.

If the algorithm does terminate then three cases must be considered.

1. The first case is when the loop is never entered so $I \not\leq S$. Observe that $I \leq \mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X)$ so $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ implies by transitivity that $I \leq S$. By contraposition, $I \not\leq S$ implies $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \not\leq S$ so that Alg. 1 correctly returns *false* since upon termination $I = X \not\leq S$.

   Otherwise the loop is iterated at least once. Upon termination after $n \geq 1$ iterates, if ever, we have $(X^n = I \vee F(X^n)) \mid (X^n \not\leq S)$, so two cases remain to be considered.
2. The second case is when $X^n = I \vee F(X^n)$. Since X^n is a fixpoint of $\boldsymbol{\lambda} X \cdot I \vee F(X)$ and for all iterates $X^n \leq \mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X)$ [11], we have $X^n = \mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X)$. Alg. 1 returns $X^n = X \leq S$ whence $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ as required.
3. The third and last case is when $X^n = X \not\leq S$. For all iterates we have $X^n \leq \mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X)$, so $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ implies by transitivity that $X^n \leq S$. By contraposition, $X^n \not\leq S$ implies $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \not\leq S$ so that Alg. 1 correctly returns *false* that is $X^n = X \not\leq S$ as required. $\square$

[6] Recall that partial correctness is correctness whenever the algorithm terminates.

2.3 Adjoined Invariance Proof Methods

Concrete Adjoinedness In the following, we assume that:

Hypothesis 2 *F has an adjoint $\widetilde{F}$ such that $\langle L, \leq\rangle \underset{F}{\overset{\widetilde{F}}{\leftrightarrows}} \langle L, \leq\rangle$ is a Galois connection* [7].

Observe that in a Galois connection, both maps are monotonic so that Hyp. 2 subsumes Hyp. 1.2.

Example 3. We have $\langle \wp(\Sigma), \subseteq\rangle \underset{post[\tau]}{\overset{\widetilde{pre}[\tau]}{\leftrightarrows}} \langle \wp(\Sigma), \subseteq\rangle$ where $pre[\tau] \triangleq post[\tau^{-1}]$, τ^{-1} is the inverse of τ and $\widetilde{pre}[\tau](X) = \neg pre[\tau](\neg X)$ [8]. To prove this, observe that:

$$\begin{aligned}
& \widetilde{pre}[\tau](Y) \\
\triangleq\ & \neg post[\tau^{-1}](\neg Y) \\
=\ & \neg\{s \mid \exists s' : s' \in \neg Y \wedge \langle s', s\rangle \in \tau^{-1}\} && \wr\text{def. } post[\tau^{-1}]\wr \\
=\ & \{s \mid \forall s' : (\langle s, s'\rangle \in \tau) \Longrightarrow (s' \in Y)\ \ \wr\text{def. set complement } \neg\text{, inverse } \tau^{-1} \text{ of a relation } \tau \text{ and logical implication } \Longrightarrow.\wr
\end{aligned}$$

It follows that:

$$\begin{aligned}
& post[\tau](X) \subseteq Y \\
\Longleftrightarrow\ & \quad \wr\text{def. } post[\tau] \text{ and set inclusion } \subseteq\wr \\
& \forall s' \in \Sigma : (\exists s \in \Sigma : s \in X \wedge \langle s, s'\rangle \in \tau) \Longrightarrow (s' \in Y) \\
\Longleftrightarrow\ & \quad \wr\text{def. logical implication } \Longrightarrow\wr \\
& \forall s \in \Sigma : \forall s' \in \Sigma : (s \in X) \Longrightarrow ((\langle s, s'\rangle \in \tau) \Longrightarrow (s' \in Y)) \\
\Longleftrightarrow\ & \quad \wr\text{def. set inclusion } \subseteq\wr \\
& X \subseteq \{s \mid \forall s' : (\langle s, s'\rangle \in \tau) \Longrightarrow (s' \in Y)\} \\
\Longleftrightarrow\ & \quad \wr\text{def. } \widetilde{pre}[\tau]\wr \\
& X \subseteq \widetilde{pre}[\tau](Y)\ .
\end{aligned}$$

□

Invariance Proof Methods The Floyd-Naur [21, 31] as well as Morris & Wegbreit [30] invariance proof methods can be generalized to fixpoint checking [13]. We have:

[7] A *Galois connection*, written $\langle L, \leq\rangle \underset{f}{\overset{g}{\leftrightarrows}} \langle M, \sqsubseteq\rangle$, is such that $\langle L, \leq\rangle$ and $\langle M, \sqsubseteq\rangle$ are posets and the maps $f \in L \mapsto M$ and $g \in M \mapsto L$ satisfy $\forall x \in L : \forall y \in M : f(x) \sqsubseteq y$ if and only if $x \leq g(y)$. This is the semi-dual of a *Galois correspondence* $\langle L, \leq\rangle \underset{f}{\overset{g}{\leftrightarrows}} \langle M, \sqsupseteq\rangle$ as originally defined by E. Galois.

[8] $\neg X \triangleq \Sigma \setminus X$ is the set complement.

Theorem 4. *Under Hyps. 1.1 & 2,*

$$\begin{aligned}
& \mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S \\
\Longleftrightarrow\; & \exists A \in L : I \leq A \;\&\; F(A) \leq A \;\&\; A \leq S \\
\Longleftrightarrow\; & \exists A \in L : I \leq A \;\&\; A \leq \widetilde{F}(A) \;\&\; A \leq S \\
\Longleftrightarrow\; & I \leq \mathit{gfp}^{\leq}\, \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)\ .
\end{aligned} \tag{1}$$

Proof. By the Galois connection, $F \in L \stackrel{\text{mon}}{\longmapsto} L$ and $\widetilde{F} \in L \stackrel{\text{mon}}{\longmapsto} L$ are $\leq$-monotonic so that by the Knaster-Kleene-Tarski fixpoint theorem [11, 34], the extreme fixpoints do exist. Moreover:

$\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$

$\Longleftrightarrow$ $\wr$For $\Longrightarrow$, $A = \mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X)$ satisfies $A = I \vee F(A)$ so $(I \vee F(A)) \leq A$ by reflexivity and $A \leq S$. For $\Longleftarrow$, $I \vee F(A) \leq A$ so the Knaster-Tarski fixpoint theorem [34] stating that $\mathit{lfp}^{\leq}\, \varphi = \bigwedge\{X \mid \varphi(X) \leq X\}$, implies that $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) = \bigwedge\{X \mid (I \vee F(X)) \leq X\} \leq A \leq S$.$\wr$

$\exists A : (I \vee F(A)) \leq A \;\&\; A \leq S$

$\Longleftrightarrow$ $\wr$def. least upper bound $\vee$$\wr$

$\exists A : I \leq A \;\&\; F(A) \leq A \;\&\; A \leq S$

$\Longleftrightarrow$ $\wr$Galois connection $\langle L, \leq\rangle \underset{F}{\overset{\widetilde{F}}{\leftrightarrows}} \langle L, \leq\rangle$ so by definition $F(A) \leq A$ if and only if $A \leq \widetilde{F}(A)$$\wr$

$\exists A : I \leq A \;\&\; A \leq \widetilde{F}(A) \;\&\; A \leq S$

$\Longleftrightarrow$ $\wr$def. greatest lower bound $\wedge$$\wr$

$\exists A : I \leq A \;\&\; A \leq (S \wedge \widetilde{F}(A))$

$\Longleftrightarrow$ $\wr$For $\Longleftarrow$, $A = \mathit{gfp}^{\leq}\, \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)$ satisfies $A = S \wedge \widetilde{F}(A)$ so $A \leq (S \wedge \widetilde{F}(A))$ by reflexivity and $I \leq A$. For $\Longrightarrow$, $A \leq S \wedge \widetilde{F}(A)$ so the dual of Knaster-Tarski fixpoint theorem [34] stating that $\mathit{gfp}^{\leq}\, \varphi = \bigvee\{X \mid X \leq \varphi(X)\}$, implies that $I \leq A \leq \bigvee\{X \mid X \leq (S \wedge \widetilde{F}(X))\} = \mathit{gfp}^{\leq}\, \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)$.$\wr$

$I \leq \mathit{gfp}^{\leq}\, \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)$ ◻

Corollary 5. *Under Hyps. 1.1 & 2, if* $\langle L, \leq\rangle \underset{F}{\overset{\widetilde{F}}{\leftrightarrows}} \langle L, \leq\rangle$ *then*

$$\langle L, \leq\rangle \underset{\boldsymbol{\lambda} I \cdot \mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X)}{\overset{\boldsymbol{\lambda} S \cdot \mathit{gfp}^{\leq} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)}{\leftrightarrows}} \langle L, \leq\rangle.$$

Proof. This simply restates that $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ if and only if $I \leq \mathit{gfp}^{\leq}\, \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)$. ◻

Concrete Invariants We call $A \in L$ an *invariant* for $\langle F, I, S\rangle$ if and only if it satisfies the verification conditions (1) stated in Th. 4.

Theorem 6. *Under Hyps. 1.1 & 2, the set $\mathcal{I}$ of invariants for $\langle F, I, S\rangle$ is a complete lattice $\langle \mathcal{I},\ \leq,\ \mathit{lfp}^{\leq}\boldsymbol{\lambda} X \cdot I \vee F(X)\,,\ \mathit{gfp}^{\leq}\boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)\,,\ \vee,\ \wedge\rangle$.*

Proof. As shown above, $\mathit{lfp}^{\leq}\boldsymbol{\lambda} X \cdot I \vee F(X)$ is an invariant, and any invariant A is such that $\mathit{lfp}^{\leq}\boldsymbol{\lambda} X \cdot I \vee F(X) \leq A$ proving that $\mathit{lfp}^{\leq}\boldsymbol{\lambda} X \cdot I \vee F(X)$ is the $\leq$-least invariant.

If $A_i, i \in \Delta$ is a family of invariants then $\forall i \in \Delta : I \leq A_i$ so obviously, $I \leq \bigvee_{i\in\Delta} A_i$. Similarly, $\forall i \in \Delta : A_i \leq S$ so $\bigvee_{i\in\Delta} A_i \leq S$ by definition of lubs [9]. Finally $\forall i \in \Delta : F(A_i) \leq A_i$ so $\bigvee_{i\in\Delta} F(A_i) \leq \bigvee_{i\in\Delta} A_i$. But $\langle L, \leq\rangle \underset{F}{\overset{\widetilde{F}}{\leftrightarrows}} \langle L, \leq\rangle$ so F is a complete joint morphism and consequently $F(\bigvee_{i\in\Delta} A_i) = \bigvee_{i\in\Delta} F(A_i) \leq \bigvee_{i\in\Delta} A_i$. We conclude that $\bigvee_{i\in\Delta} A_i \in \mathcal{I}$ is an invariant so $\vee$ is obviously the lub in $\mathcal{I}$.

That $\boldsymbol{\lambda} S \cdot \mathit{gfp}^{\leq}\boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)$ is the greatest invariant and $\wedge$ is the glb [10] follows by the order-theoretic duality principle where the dual of I is S and that of F is $\widetilde{F}$. $\square$

2.4 The Dual Concrete Fixpoint Checking Algorithm

It follows, as observed in [17], that Alg. 2 below which is based upon the iterative computation of $\mathit{gfp}^{\leq}\boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)$ is equivalent to the previous Alg. 1 for checking that $\mathit{lfp}^{\leq}\boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$. For the special case of reachability analysis (where $\widetilde{F} = \widetilde{\mathit{pre}}[\tau]$) this Alg. 2 corresponds to the backward state space traversal which is traditional in model-checking [26]. It is nonetheless traditional in program analysis (see a.o. [7]):

Algorithm 2

$Y := S;\ \ \mathit{Go} := (I \leq Y);$
while Go **do**
  $Y' := S \wedge \widetilde{F}(Y);$
  $\mathit{Go} := (Y \neq Y')\ \&\ (I \leq Y');$
  $Y := Y';$
od;
return $(I \leq Y);$

Theorem 7. *Under Hyps. 1.1 & 2, Alg. 2 is partially correct: when terminating, it returns $\mathit{lfp}^{\leq}\boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$.*

---

[9] *lub* is short for *least upper bound*.
[10] *glb* is short for *greatest lower bound*.

Proof. By order-theoretic duality extended so that the dual of I is S and that of F is $\widetilde{F}$ so $\langle L, \geq\rangle \xrightleftharpoons[\widetilde{F}]{F} \langle L, \geq\rangle$, we know from the proof of Alg. 1 that Alg. 2 returns *true* if and only if $I \leq \mathit{gfp}^{\leq}\, \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)$ or equivalently, by Th. 4, if and only if $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ as desired. □

2.5 Adjoined Concrete Fixpoint Checking

Adjoined Concrete Fixpoint Checking and its Dual Define $G \stackrel{\Delta}{=} \boldsymbol{\lambda} X \cdot I \vee F(X)$ and $\widetilde{G} \stackrel{\Delta}{=} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)$. We have:

Theorem 8. *Under Hyps. 1.1 & 2, $\mathit{lfp}^{\leq} G \leq S$ if and only if $\mathit{lfp}^{\leq} G \leq \mathit{gfp}^{\leq} \widetilde{G}$.*

Proof. We have $\mathit{lfp}^{\leq} G = G(\mathit{lfp}^{\leq} G) = I \vee F(\mathit{lfp}^{\leq} G)$ so $F(\mathit{lfp}^{\leq} G) \leq \mathit{lfp}^{\leq} G$ by def. of lubs. It follows that $\mathit{lfp}^{\leq} G \leq \widetilde{F}(\mathit{lfp}^{\leq} G)$ by the Galois connection $\langle L, \leq\rangle \xrightleftharpoons[F]{\widetilde{F}} \langle L, \leq\rangle$. So if $\mathit{lfp}^{\leq} G \leq S$ then $\mathit{lfp}^{\leq} G \leq S \wedge \widetilde{F}(\mathit{lfp}^{\leq} G)$ hence $\mathit{lfp}^{\leq} G \leq \widetilde{G}(\mathit{lfp}^{\leq} G)$ proving $\mathit{lfp}^{\leq} G \leq \mathit{gfp}^{\leq} \widetilde{G}$ since $\mathit{gfp}^{\leq} \widetilde{G} = \bigvee\{x \mid x \leq \widetilde{G}(x)\}$ by the dual of Tarski's fixpoint theorem [34]. Reciprocally, if $\mathit{lfp}^{\leq} G \leq \mathit{gfp}^{\leq} \widetilde{G}$ then $\mathit{lfp}^{\leq} G \leq S \wedge \widetilde{F}(\mathit{gfp}^{\leq} \widetilde{G})$ by the fixpoint property and def. of $\widetilde{G}$ so $\mathit{lfp}^{\leq} G \leq S$ by def. of glbs. We conclude that $\mathit{lfp}^{\leq} G \leq S$ if and only if $\mathit{lfp}^{\leq} G \leq \mathit{gfp}^{\leq} \widetilde{G}$. □

By duality, we have

Theorem 9. *Under Hyps. 1.1 & 2, $I \leq \mathit{gfp}^{\leq} \widetilde{G}$ if and only if $\mathit{lfp}^{\leq} G \leq \mathit{gfp}^{\leq} \widetilde{G}$.*

Proof. By the order theoretic duality principle where I is the dual of S and $\widetilde{F}$ that of F, hence $\widetilde{G}$ that of G. □

The Adjoined Concrete Fixpoint Checking Algorithm This observation leads to the combination of the above two Algs. 1 and 2 in the new one:

Algorithm 3

$X := I$; $Y := S$; $\mathit{Go} := (X \leq Y)$;
while Go **do**
  $X' := I \vee F(X)$; $Y' := S \wedge \widetilde{F}(X)$;
  $\mathit{Go} := (X \neq X')\ \&\ (Y \neq Y')\ \&\ (X' \leq Y')$;
  $X := X'$; $Y := Y'$;
od;
return $(X \leq Y)$;

Optimizations including parallel versions can be easily derived from the above basic version of Alg. 3. They will not be considered here, although they are essential to be more time-efficient than the previous Algs. 1 and 2. The advantage of this parallel version of Alg. 3 is that errors $\mathit{lfp}^{\le} \boldsymbol{\lambda} X \cdot I \vee F(X) \not\le S$ may be discovered faster than with either Alg. 1 or Alg. 2 since the (parallel) computation of the fixpoints stops as soon as a fixpoint is reached or an error is found.

The partial correctness proof of the algorithm is not completely trivial and is given below. Total correctness, hence termination, requires additional hypotheses such as L satisfies the ascending and descending chain conditions e.g. following from the finite-state hypothesis.

Theorem 10. *Under Hyps. 1.1 & 2, Alg. 3 is partially correct: when terminating, it returns* $\mathit{lfp}^{\le} \boldsymbol{\lambda} X \cdot I \vee F(X) \le S$.

Proof. By the Galois connection $\langle L, \le\rangle \underset{F}{\overset{\widetilde{F}}{\leftrightarrows}} \langle L, \le\rangle$, F is 0-strict and a complete $\vee$-morphism so G is a complete $\vee$-morphism. Let $X^0 = I$ be the initial value of X in the loop and X^n its value at the end of the n-th iteration. Let $\varphi^0(x) \triangleq x$ and $\varphi^{n+1}(x) \triangleq \varphi(\varphi^n(x))$. We have $X^0 = I = F^0(I) = I \vee 0 = I \vee F(0) = G(0)$. Assume by induction hypothesis that $X^n = \bigvee_{k=0}^{n} F^k(I) = G^{n+1}(0)$. We have $X^{n+1} = I \vee F(X^n) = G(X^n) = G(G^{n+1}(0)) = G^{n+2}(0)$. Moreover $X^{n+1} = I \vee F(X^n) = I \vee F(\bigvee_{k=0}^{n} F^k(I)) = I \vee \bigvee_{k=0}^{n} F(F^k(I)) = F^0(I) \vee \bigvee_{k=0}^{n} F^{k+1}(I) = \bigvee_{k=0}^{n+1} F^k(I)$. It follows by recurrence and the Kleene-Tarski theorem [11, 34] that $\forall n \in \mathbb{O} : I \le X^n \le \bigvee_{k \ge 0} G^n(0) \le \mathit{lfp}^{\le} G$.

The order-theoretic dual of the above proof, extended so that the dual of I is S, that of F is $\widetilde{F}$ so that of G is $\widetilde{G}$, shows that $\forall n \in \mathbb{O} : \mathit{gfp}^{\le} \widetilde{G} \le \bigwedge_{k \ge 0} \widetilde{G}^n(1) \le Y^n \le S$.

Combining the above results by transitivity, we observe that if $\mathit{lfp}^{\le} G \le S$ then $\mathit{lfp}^{\le} G \le \mathit{gfp}^{\le} \widetilde{G}$ so $\forall n \in \mathbb{O} : X^n \le \mathit{lfp}^{\le} G \le \mathit{gfp}^{\le} \widetilde{G} \le Y^n$. By contraposition $\exists n \in \mathbb{O} : X^n \not\le Y^n$ implies $\mathit{lfp}^{\le} G \not\le S$.

If the algorithm terminates then four cases must be considered.

1. The first case is when the loop is never entered so $I \not\le S$. We have $\mathit{lfp}^{\le} \boldsymbol{\lambda} X \cdot I \vee F(X) \le S$ which implies $I \le \mathit{gfp}^{\le} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X) = S \wedge \widetilde{F}(\mathit{gfp}^{\le} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X))$ by the fixpoint property so that $I \le S$ by definition of least upper bounds. So $I \not\le S$ implies $\mathit{lfp}^{\le} \boldsymbol{\lambda} X \cdot I \vee F(X) \not\le S$. Therefore the algorithm returns *false*, as required;

   Otherwise the loop is entered at least once and termination implies $(X = X') \vee (Y = Y') \vee (X \not\le Y)$.

2. In the second case, termination is with $X \not\le Y$ so $\exists n \in \mathbb{O} : X^n \not\le Y^n$ which implies $\mathit{lfp}^{\le} G \not\le S$ so $\mathit{lfp}^{\le} \boldsymbol{\lambda} X \cdot I \vee F(X) \not\le S$ and the algorithm returns *false*, as required;
3. The third case is $(X = X')$ & $(X \le Y)$. We have $X = G(X)$ and $\exists n \in \mathbb{O} : X = X^n \le \mathit{lfp}^{\le} G$ so $X = \mathit{lfp}^{\le} G$ by def. of the least fixpoint. Moreover, $Y = Y^n \le S$ whence $X \le Y$ implies $\mathit{lfp}^{\le} G \le S$ that is $\mathit{lfp}^{\le} \boldsymbol{\lambda} X \cdot I \vee F(X) \le S$ and the algorithm returns *true*, as required.

4. The fourth and final case is $(Y = Y') \,\&\, (X \leq Y)$. The order-theoretic dual of the above proof, extended so that the dual of I is S, that of F is $\widetilde{F}$ so that of G is $\widetilde{G}$, shows that $I \leq \mathit{gfp}^{\leq}\, \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)$ or equivalently, by Th. 4, $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ and the algorithm returns *true*, as required. □

3 Abstract Fixpoint Checking

3.1 Abstract Interpretation

In the context of program analysis, abstraction [10] is needed for expressiveness (the elements of L are not computer-representable) and undecidability (the fixpoints are not effectively computable). In the context of model-checking [4], abstraction is needed for concrete complexity reasons, because of machine memory-size (so called state-explosion problem) and/or computation time limitations. The difference is that in program analysis, the concrete semantics is not computable whereas it is in the case of model-checking [11]. Abstract interpretation [10, 12] can be used in both cases, but an important difference is that the abstraction/concretization can be considered to be computable for model-checking which is hardly conceivable for program analysis. In the latter case the abstraction/concretization process must be handled by hand (may be with some computer assistance) whereas in the first case, it can be (at least partially) automatized (see e.g. [5, 20, 24]).

3.2 The Abstract Fixpoint Checking Algorithm

We now consider an abstract complete lattice $\langle M, \sqsubseteq, \perp, \top, \sqcap, \sqcup \rangle$ which is an abstraction of $\langle L, \leq, 0, 1, \vee, \wedge \rangle$ by the abstraction/concretization pair $\langle \alpha, \gamma \rangle$. For simplicity we assume that any concrete property $p \in L$ has a best approximation $\alpha(p) \in M$ which is tantamount to assuming that $\langle L, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq \rangle$ is a Galois connection [12, 14]:

Hypotheses 3 *1. The abstract domain* $\langle M, \sqsubseteq, \perp, \top, \sqcap, \sqcup \rangle$ *is a complete lattice;*
2. $\langle L, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq \rangle$.

Example 11. As observed in [18], the abstraction which is almost exclusively used in abstract model-checking has the form $\alpha_h(X) \triangleq \{h(x) \mid x \in X\}$ and $\gamma_h(Y) \triangleq \{x \mid h(x) \in Y\}$ where $h \in \Sigma \mapsto \overline{\Sigma}$. Considering the function h as a relation, we have $\alpha_h = \mathit{post}[h]$ and $\gamma_h = \widetilde{\mathit{pre}}[h]$ so that $\langle \wp(\Sigma), \subseteq \rangle \underset{\alpha_h}{\overset{\gamma_h}{\leftrightarrows}} \langle \wp(\overline{\Sigma}), \subseteq \rangle$, as shown in Ex. 3. An example for $\Sigma = \mathbb{Z}$ consists in choosing $h(z)$ to be the sign of z [12]. □

[11] Obviously, if the state space is infinite then the situation may be the same in model-checking as it is in program analysis. However the boolean abstractions used in model-checking with BDD encoding, are too weak when considering complex data structures, higher-order recursion, etc. which are a common difficulty in program analysis.

The abstract form Alg. 4 below of the fixpoint checking Alg. 1 is classical in abstract interpretation [7, 10, 12] [12]:

Algorithm 4

$X := \alpha(I)$; $Go := (\gamma(X) \leq S)$;
while Go **do**
  $X' := \alpha(I \vee F(\gamma(X)))$;
  $Go := (X \neq X')$ & $(\gamma(X') \leq S)$;
  $X := X'$;
od;
return if $(\gamma(X) \leq S)$ ***then*** *true* ***else*** *I don't know;*

Theorem 12. *Under Hyps. 1.1 & 3, Alg. 4 is partially correct: if it terminates and returns "true" then* $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$.

Proof. We have $\langle L, \leq\rangle \xleftrightarrow[\alpha]{\gamma} \langle M, \sqsubseteq\rangle$ so α is a complete join morphism hence $\alpha(I \vee F(\gamma(X))) = \alpha(I) \sqcup \alpha(F(\gamma(X)))$. It follows that $\alpha(I) \leq \alpha(I) \sqcup \alpha \circ F \circ \gamma(I)$ [13] so, as shown in [11], the transfinite sequence $X^0 \triangleq \alpha(I)$, $X^{\delta+1} \triangleq \alpha(I \vee F(\gamma(X^\delta))$ for all successor ordinals $\delta \in \mathbb{O}$ and $X^\lambda \triangleq \bigsqcup_{\delta<\lambda} X^\delta$ for all limit ordinals $\lambda \in \omega.\mathbb{O}$ is increasing and ultimately stationary, its limit being the least fixpoint of $\boldsymbol{\lambda} X \cdot \alpha(I \vee F(\gamma(X)))$ greater than $\alpha(I)$, that is the least fixpoint of $\boldsymbol{\lambda} X \cdot \alpha(I \vee F(\gamma(X))$. By recurrence, X^n is the value of the program variable X at the end of the n-th iteration in the loop, if any, with $X^0 = \alpha(I)$ being the initial value of X upon entry of the loop.

If the algorithm does terminate then three cases must be considered.

1. The first case is when the loop is never entered so $\gamma(I) \not\leq S$. Then Alg. 4 returns *I don't know* which is certainly correct.
   Otherwise the loop is iterated at least once. Upon termination after $n \geq 1$ iterates, if ever, we have $(X^n = \alpha(I \vee F(\gamma(X^n))) \mid (\gamma(X^n) \leq S)$, so two cases remain to be considered.
2. The second case is when $X^n = \alpha(I \vee F(\gamma(X^n)))$. Since X^n is a fixpoint of $\boldsymbol{\lambda} X \cdot \alpha(I \vee F(\gamma(X)))$ and for all iterates $X^n \sqsubseteq \mathit{lfp}^{\sqsubseteq}\, \alpha(I \vee F(\gamma(X)))$ [11], we have $X^n = \mathit{lfp}^{\sqsubseteq}\, \boldsymbol{\lambda} X \cdot \alpha(I \vee F(\gamma(X)))$. Alg. 4 checks $\gamma(X^n) = \gamma(X) \leq S$. When returning *true*, we have $\gamma(\mathit{lfp}^{\sqsubseteq}\, \boldsymbol{\lambda} X \cdot \alpha(I \vee F(\gamma(X)))) \leq S$. By a classical fixpoint approximation result of abstract interpretation [12, Th. 7.1.0.4], $\alpha(\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X)) \sqsubseteq \mathit{lfp}^{\sqsubseteq}\, \boldsymbol{\lambda} X \cdot \alpha(I \vee F \circ \gamma(X))$ so by $\langle L, \leq\rangle \xleftrightarrow[\alpha]{\gamma} \langle M, \sqsubseteq\rangle$, we have $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq \gamma(\mathit{lfp}^{\sqsubseteq}\, \boldsymbol{\lambda} X \cdot \alpha(I \vee F \circ \gamma(X)))$ whence by transitivity $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ as required.

---

[12] Since in program analysis neither γ nor $\leq$ is computable the termination condition $\gamma(X) \leq S$ is replaced by the abstract form $X \sqsubseteq \alpha(S)$. When assuming that S is an abstract specification in that $S = \gamma(\alpha(S))$, this abstract condition is stronger (whence correct) since $X \sqsubseteq \alpha(S)$ implies by monotony that $\gamma(X) \sqsubseteq \gamma(\alpha(S)) = S$.

[13] $\circ$ is functional composition $f \circ g(x) \triangleq f(g(x))$.

3. The third and last case is when $\gamma(X^n) \leq S$. Then $\gamma(X^n) = \gamma(X) \not\leq S$ so that Alg. 4 returns *I don't know* which is certainly correct. $\square$

3.3 Partial Completeness

We have seen that any abstraction $\langle \alpha, \gamma \rangle$ is *sound* in that Alg. 4 returns *true* only if $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$.

This abstraction is said to be *partially complete* if, whenever Alg. 4 terminates and $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ then the returned result is *true*[14].

Because soundness is mandatory, partial completeness corresponds to the case when Alg. 4 returns *true* upon termination exactly when $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$, that is Alg. 4 is equivalent to Alg. 1, up to termination [15].

3.4 Partially Complete Abstractions for Algorithm 4

Characterization of Partially Complete Abstractions for Algorithm 4 It was informally observed in [15] (and similarly in [27]) that partial completeness in abstract interpretation requires an invariance proof. More formally the abstract domain must contain the exact representation $A = \alpha(A')$ of an invariant $A' = \gamma(A)$ for $\langle F, I, S \rangle$:

Theorem 13. *Under Hyps. 1.1 & 3, the abstraction $\langle \alpha, \gamma \rangle$ is partially complete for Alg. 4 if and only if $\alpha(L)$ contains an abstract value A such that $\gamma(A)$ is an invariant for $\langle F, I, S \rangle$.*

Proof. Assume that $\langle \alpha, \gamma \rangle$ is a partially complete abstraction for Alg. 4. If $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ then Alg. 4 must return *true* so upon exit $\gamma(X) \leq S$. By definition of the loop termination condition $\neg Go$, the loop must have been entered at least once. So upon termination, after $n \geq 1$ iterations, the final value X^n of X satisfies $X^n = \alpha(I \vee F(\gamma(X))) = \alpha(I) \sqcup \alpha \circ F \circ \gamma(X^n)$ so $\alpha(I) \sqsubseteq X^n$ and $\alpha \circ F \circ \gamma(X^n) \sqsubseteq X^n$ by definition of lubs, whence by $\langle L, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq \rangle$, $I \leq \gamma(X^n)$ and $F \circ \gamma(X^n) \leq \gamma(X^n)$. We conclude that $\gamma(X^n)$ is an invariant for $\langle F, I, S \rangle$, so $A = X^n$.

Reciprocally let $A \in \alpha(L)$ be such $\gamma(A)$ is an invariant for $\langle F, I, S \rangle$. We have $I \leq \gamma(A)$ so by $\langle L, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq \rangle$ $\alpha(I) \sqsubseteq A$ whence $X^0 \sqsubseteq A$. By recurrence, assume that $X^n \sqsubseteq A$ and that one more iterate is needed in the loop. We have $I \leq \gamma(A)$ and $F(\gamma(A)) \leq \gamma(A)$ so by $\langle L, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq \rangle$,

[14] Observe that this notion of *partial completeness* is different from the notions of *fixpoint completeness* ($\alpha(\mathit{lfp}^{\leq} G) = \mathit{lfp}^{\sqsubseteq} \alpha \circ G \circ \gamma$) and the stronger one of *local completeness* ($\alpha \circ G = \alpha \circ G \circ \gamma \circ \alpha$) introduced in [12] and further studied in [22, 23].

[15] Observe that for locally complete abstractions, termination of the concrete Alg. 1 implies that of the abstract Alg. 4 since, as shown in [9, Th. 3], convergence of the abstract iterates to a fixpoint is faster than that of the concrete iterates for locally complete abstractions.

$\alpha(I) \sqsubseteq A$ and $\alpha(F(\gamma(A))) \sqsubseteq A$ whence by monotony $X^{n+1} = \alpha(I \vee F(\gamma(X^n))) \sqsubseteq \alpha(I \vee F(\gamma(A))) = \alpha(I) \sqcup \alpha(F(\gamma(A))) \sqsubseteq A$. Observe that upon termination, if any, X has value X^n such that $X^n \sqsubseteq A$ so by monotony $\gamma(X^n) \leq \gamma(A)$. Since $\gamma(A)$ is an invariant $\gamma(A) \leq S$ so by transitivity $X = \gamma(X^n) \leq S$ whence the algorithm returns *true*, if it terminates, as required. ☐

The Most Abstract Partially Complete Abstraction for Algorithm 4 Among the partially complete abstractions, we are interested in the simplest ones, with a minimal number of abstract values, in particular those corresponding to the weakest or strongest concrete properties. Formally:

Definition 14. *The* most abstract partially complete abstraction $\langle\overline{\alpha}, \overline{\gamma}\rangle$*, if it exists, is defined such that:*
1. *The* abstract domain $\overline{M} = \overline{\alpha}(L)$ *has the smallest possible cardinality;*
2. *If another abstraction $\langle\alpha', \gamma'\rangle$ is a partially complete abstraction with the same cardinality, then there exists a bijection β such that $\forall x \in \overline{M} : \gamma'(\beta(x)) \leq \overline{\gamma}(x)$* [16]*.*

Theorem 15. *Under Hyps. 1.1 & 3, the most abstract partially complete abstraction for Alg. 4 is such that:*
- *if $S = 1$ then $\overline{M} = \{\top\}$ where $\overline{\alpha} \triangleq \boldsymbol{\lambda} X \cdot \top$ and $\overline{\gamma} \triangleq \boldsymbol{\lambda} Y \cdot 1$;*
- *if $S \neq 1$ then $\overline{M} = \{\perp, \top\}$ where $\perp \sqsubseteq \perp \sqsubset \top \sqsubseteq \top$ with $\langle\overline{\alpha}, \overline{\gamma}\rangle$ such that:*

$$\begin{aligned}
\overline{\alpha}(X) &\triangleq \perp && \textit{if } X \leq \textit{gfp}^{\leq} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)\\
\overline{\alpha}(X) &\triangleq \top && \textit{otherwise}\\
\overline{\gamma}(\perp) &\triangleq \textit{gfp}^{\leq} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X) && \qquad (2)\\
\overline{\gamma}(\top) &\triangleq 1
\end{aligned}$$

Proof. In the first case $S = 1$, we have $I \leq 1$, $F(1) \leq 1$ and $1 \leq S$ so $\overline{\gamma}(\top) = 1$ is invariant for $\langle F, I, S\rangle$ whence, by Th. 13, the abstraction $\langle\overline{\alpha}, \overline{\gamma}\rangle$ is partially complete for Alg. 4. $\overline{M} = \{\top\}$ has the smallest possible cardinality since a complete lattice is not empty ($\sqcup\emptyset$ must exist). $\overline{M} = \{\top\}$ is obviously the most abstract since $\overline{\gamma}(\top) = 1$.

The second case is when $S \neq 1$. By definition (2), $\overline{\alpha}(L)$ contains $\perp$ such that $\overline{\gamma}(\perp) \triangleq \textit{gfp}^{\leq} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)$ which, by Th. 6, is an invariant so that, by Th. 13, the abstraction $\langle\overline{\alpha}, \overline{\gamma}\rangle$ is partially complete for Alg. 4.

To show that the cardinality of $\overline{M}$ is minimal, let us consider another $M' = \alpha'(L)$ such that $\langle\alpha', \gamma'\rangle$ is partially complete for Alg. 4. Observe that L is a complete lattice whence $M' = \overline{\alpha}(L)$ is also a complete lattice whence not empty. Let $\top'$ be its supremum. We have $\alpha'(1) \sqsubseteq \top'$ whence $1 \leq \gamma'(\top')$ so $1 = \gamma'(\top')$ by antisymmetry since 1 is the supremum. Since $S \neq 1$, we have $\gamma'(\top') \nleq S$ so that $\gamma'(\top')$ is not an invariant for $\langle F, I, S\rangle$. By Th. 13, it follows that M' must

[16] Otherwise stated, the abstract values in $\overline{\alpha}(L)$ are more approximate than the corresponding elements in $\alpha'(L)$.

contain another element $A \in M'$ such that $\gamma'(A)$ is an invariant for $\langle F, I, S\rangle$ so $A \neq \top'$ proving that the cardinality of $M' = \alpha'(L)$ must be at least 2. It follows that the cardinality of $\overline{M}$ is minimal.

To show that $\langle\overline{\alpha}, \overline{\gamma}\rangle$ is the most abstract, let us consider another $M' = \alpha'(L)$ of cardinality 2 (i.e. $M' = \{\perp', \top'\}$, $\perp' \sqsubseteq' \perp' \sqsubset' \top' \sqsubseteq' \top'$) such that $\langle\alpha', \gamma'\rangle$ is partially complete for Alg. 4. Since $\top'$ is the supremum of M' and $\langle L, \leq\rangle \underset{\alpha'}{\overset{\gamma'}{\leftrightarrows}} \langle M', \sqsubseteq'\rangle$, we have $\gamma'(\top') = 1 = \overline{\gamma}(\top)$ which are not invariant for $\langle F, I, S\rangle$. By partial completeness hypothesis and Th. 13, $\gamma'(\perp')$ must be an invariant for $\langle F, I, S\rangle$ so, by Th. 6, $\gamma'(\perp') \leq gfp^{\leq} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X) = \overline{\gamma}(\perp)$. The bijection $\beta(\perp) = \perp'$ and $\beta(\top) = \top'$ is such that $\forall x \in \overline{M} : \gamma'(\beta(x)) \leq \overline{\gamma}(x)$, proving that $\langle\overline{\alpha}, \overline{\gamma}\rangle$ is the most abstract partially complete abstraction for Alg. 4. $\square$

The Least Abstract Partially Complete Abstraction for Algorithm 4 The *least abstract partially complete abstraction* is defined dually to definition 14.

Theorem 16. *Under Hyps. 1.1 & 3, the least abstract partially complete abstraction for Alg. 4 is such that:*
- *if $I = 1$ then $\underline{M} = \{\top\}$ where $\underline{\alpha} \triangleq \boldsymbol{\lambda} X \cdot \top$ and $\underline{\gamma} \triangleq \boldsymbol{\lambda} Y \cdot 1$;*
- *if $I \neq 1$ then $\underline{M} = \{\perp, \top\}$ where $\perp \sqsubseteq \perp \sqsubset \top \sqsubseteq \top$ with $\langle\underline{\alpha}, \underline{\gamma}\rangle$ such that:*

$$
\begin{aligned}
&\underline{\alpha}(X) \triangleq \perp && \textit{if } X \leq \mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X)\\
&\underline{\alpha}(X) \triangleq \top && \textit{otherwise}\\
&\underline{\gamma}(\perp) \triangleq \mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) && \\
&\underline{\gamma}(\top) \triangleq 1 &&
\end{aligned} \tag{3}
$$

Proof. In the first case $I = 1$, we have $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) = 1$ so if $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ then $1 \leq S$ so $S = 1$. We have $I = 1 \leq 1$, $F(1) \leq 1$ and $1 \leq 1 = S$ so $\underline{\gamma}(\top) = 1$ is invariant for $\langle F, I, S\rangle$ whence, by Th. 13, the abstraction $\langle\underline{\alpha}, \underline{\gamma}\rangle$ is partially complete for Alg. 4. $\underline{M} = \{\top\}$ has the smallest possible cardinality since a complete lattice is never empty. let $M' = \{\top'\}$ be another partially complete abstraction $\langle\alpha', \gamma'\rangle$ with the same cardinality. We have $\gamma'(\top')$ which is an invariant for $\langle F, I, S\rangle$ so $\gamma'(\top') \geq I = 1$ proving that $\gamma'(\top') = 1$. We have $\forall x \in \underline{M} : \gamma'(\beta(x)) \leq \underline{\gamma}(x)$ by definition $\beta(\top) = \top'$.

The second case is when $I \neq 1$. By definition (3), $\underline{\alpha}(L)$ contains $\perp$ such that $\underline{\gamma}(\perp) \triangleq \mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X)$ which, by Th. 6, is an invariant so that, by Th. 13, the abstraction $\langle\underline{\alpha}, \underline{\gamma}\rangle$ is partially complete for Alg. 4.

To show that the cardinality of $\underline{M}$ is minimal, let us consider another $M' = \alpha'(L)$ such that $\langle\alpha', \gamma'\rangle$ is partially complete for Alg. 4. Observe that L is a complete lattice whence $M' = \underline{\alpha}(L)$ is also a complete lattice whence not empty. Let $\top'$ be its supremum. We have $\alpha'(1) \sqsubseteq \top'$ whence $1 \leq \gamma'(\top')$ so $1 = \gamma'(\top')$ by antisymmetry since 1 is the supremum. Since $I \neq 1$, we have $I \not\leq \gamma'(\top')$ so that $\gamma'(\top')$ is not an invariant for $\langle F, I, S\rangle$. By Th. 13, it follows that M' must contain another element $A \in M'$ such that $\gamma'(A)$ is an invariant for $\langle F, I, S\rangle$ so $A \neq \top'$ proving that the cardinality of $M' = \alpha'(L)$ must be at least 2. It follows that the cardinality of $\underline{M}$ is minimal.

To show that $\langle\underline{\alpha}, \underline{\gamma}\rangle$ is the least abstract, let us consider another $M' = \alpha'(L)$ of cardinality 2 (i.e. $M' = \{\perp', \top'\}$, $\perp' \sqsubseteq' \perp' \sqsubset' \top' \sqsubseteq' \top'$) such that $\langle\alpha', \gamma'\rangle$ is partially complete for Alg. 4. Since $\top'$ is the supremum of M' and $\langle L, \leq\rangle \underset{\alpha'}{\overset{\gamma'}{\leftrightarrows}} \langle M', \sqsubseteq'\rangle$, we have $\gamma'(\top') = 1 = \underline{\gamma}(\top)$ which are not invariant for $\langle F, I, S\rangle$. By partial completeness hypothesis and Th. 13, $\gamma'(\perp')$ must be an invariant for $\langle F, I, S\rangle$ so, by Th. 6, $\underline{\gamma}(\perp) \leq \mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq \gamma'(\perp')$. The bijection $\beta(\perp) = \perp'$ and $\beta(\top) = \top'$ is such that $\forall x \in \underline{M} : \gamma'(\beta(x)) \geq \underline{\gamma}(x)$, proving that $\langle\underline{\alpha}, \underline{\gamma}\rangle$ is the least abstract partially complete abstraction for Alg. 4. ☐

The Complete Lattice of Minimal Partially Complete Abstractions for Algorithm 4 By Th. 6, the set $\mathcal{I}$ of invariants is a complete lattice $\langle \mathcal{I}, \leq, \mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X), \mathit{gfp}^{\leq} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X), \vee, \wedge\rangle$. Its abstract image leads to the partially complete abstractions of minimal cardinality for Alg. 4:

Theorem 17. *Under Hyps. 1.1 & 3, the set $\mathcal{A}$ of partially complete abstractions of minimal cardinality for Alg. 4 is the set of all $\langle M, \sqsubseteq, \alpha, \gamma\rangle$ such that $M = \{\perp, \top\}$ with $\perp \sqsubseteq \perp \sqsubseteq \top \sqsubseteq \top$, Hyp. 3.2 holds, $\gamma(\perp) \in \mathcal{I}$ and $\perp = \top$ if and only if $\gamma(\top) \in \mathcal{I}$.*

The relation $\langle\{\perp, \top\}, \sqsubseteq, \alpha\rangle\gamma \preceq \langle\{\perp', \top'\}, \sqsubseteq', \alpha'\rangle\gamma'$ is a pre-ordering on $\mathcal{A}$. Let $\langle\{\perp, \top\}, \alpha, \gamma\rangle \cong \langle\{\perp', \top'\}, \alpha', \gamma'\rangle$ if and only if $\gamma(\perp) = \gamma'(\perp')$ be the corresponding equivalence.

The quotient $\mathcal{A}_{/\cong}$ is a complete lattice [17] for $\preceq$ with infimum class representative $\langle\underline{M}, \underline{\alpha}, \underline{\gamma}\rangle$ and supremum $\langle\overline{M}, \overline{\alpha}, \overline{\gamma}\rangle$.

Proof. We have $\gamma(\top) \in \mathcal{I}$ or $\gamma(\perp) \in \mathcal{I}$ so by Th. 13, $\langle M, \sqsubseteq, \alpha, \gamma\rangle$ is a partially complete abstraction for Alg. 4.

By Hyp. 3.2, $\alpha(1) \sqsubseteq \top$ so $1 \leq \gamma(\top)$ whence $\gamma(\top) = 1$. If $\gamma(\top) \in \mathcal{I}$ then $\perp = \top$ so the cardinality is minimal since M is a complete lattice whence not empty. Otherwise $\gamma(\top) \notin \mathcal{I}$ and $\gamma(\perp) \in \mathcal{I}$ so $\perp \neq \top$. Again the cardinality of M is minimal since by Th. 13, M must contain an element A such that $\gamma(A) \in \mathcal{I}$ and, in this second case, A cannot be $\top$. So we conclude that $\mathcal{A}$ is the set of partially complete abstractions of minimal cardinality for Alg. 4.

By definition $\preceq$ is a pre-order on $\mathcal{A}$ since $\leq$ is a partial order on L. Consequently the restriction of $\preceq$ to the representatives of the equivalent classes of the quotient $\mathcal{A}_{/\cong}$ is a poset.

Let $\langle M_i, \sqsubseteq_i, \alpha_i, \gamma_i\rangle$, $i \in \Delta$ be given elements of $\mathcal{A}$. By Th. 6, $\bigvee_{i\in\Delta} \gamma_i(\perp) \in \mathcal{I}$ is an invariant. So there is some $\langle M, \sqsubseteq, \alpha, \gamma\rangle \in \mathcal{A}$ (may be with $\perp = \top$) such that $\gamma(\perp) = \bigvee_{i\in\Delta} \gamma_i(\perp)$. Trivially, the class of $\langle M, \sqsubseteq, \alpha, \gamma\rangle \in \mathcal{A}$ is the lub of the set $\{\langle M_i, \sqsubseteq_i, \alpha_i, \gamma_i\rangle \mid i \in \Delta\}$ for $\preceq$.

[17] Observe however that it is not a sublattice of the lattice of abstract interpretations of [10, 12] with reduced product as glb.

The fact that $\langle \underline{M}, \underline{\alpha}, \underline{\gamma} \rangle$ and $\langle \overline{M}, \overline{\alpha}, \overline{\gamma} \rangle$ are representative of the extreme classes of $\mathcal{A}_{/\cong}$ is also a direct consequence of Th. 6. $\square$

3.5 Abstract Adjoinedness

In the following, we assume that we have a dual abstraction:

Hypothesis 4 $\langle L, \geq \rangle \underset{\widetilde{\alpha}}{\overset{\widetilde{\gamma}}{\leftrightarrows}} \langle M, \sqsupseteq \rangle$.

Example 18. A classical example [18] when $\langle L, \leq, 0, 1, \vee, \wedge, \neg \rangle$ and $\langle M, \sqsubseteq, \bot, \top, \sqcap, \sqcup, \backsim \rangle$ are complete boolean lattices and $\langle L, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq \rangle$ is to define $\widetilde{\alpha} = \backsim \circ\, \alpha \circ \neg$ and $\widetilde{\gamma} = \neg \circ \gamma \circ \backsim$ so that $\langle L, \geq \rangle \underset{\widetilde{\alpha}}{\overset{\widetilde{\gamma}}{\leftrightarrows}} \langle M, \sqsupseteq \rangle$ or equivalently $\langle M, \sqsubseteq \rangle \underset{\widetilde{\gamma}}{\overset{\widetilde{\alpha}}{\leftrightarrows}} \langle L, \leq \rangle$. Indeed:

$$
\begin{array}{lll}
 & \widetilde{\alpha}(X) \sqsupseteq Y & \\
\Longleftrightarrow & \backsim \circ\, \alpha \circ \neg(X) \sqsupseteq Y & \wr \text{def. } \widetilde{\alpha} \int \\
\Longleftrightarrow & \alpha \circ \neg(X) \sqsubseteq \backsim Y & \wr \text{contraposition in } M \int \\
\Longleftrightarrow & \neg(X) \leq \gamma(\backsim Y) & \wr \langle L, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq \rangle \int \\
\Longleftrightarrow & X \geq \neg \circ \gamma \circ \backsim(Y) & \wr \text{contraposition in } L \int \\
\Longleftrightarrow & X \geq \widetilde{\gamma}(Y) & \wr \text{def. } \widetilde{\gamma} \int
\end{array}
$$

For a typical example, we have $\langle \wp(\Sigma), \subseteq \rangle \underset{post[\tau]}{\overset{\widetilde{pre}[\tau]}{\leftrightarrows}} \langle \wp(\Sigma), \subseteq \rangle$ and $\widetilde{pre}[\tau](X) = \neg pre[\tau](\neg X)$ (see Ex. 3) so that by defining $\widetilde{post}[\tau](X) = \neg post[\tau](\neg X)$ we have $\langle \wp(\Sigma), \supseteq \rangle \underset{\widetilde{post}[\tau]}{\overset{pre[\tau]}{\leftrightarrows}} \langle \wp(\Sigma), \supseteq \rangle$ or equivalently $\langle \wp(\Sigma), \subseteq \rangle \underset{pre[\tau]}{\overset{\widetilde{post}[\tau]}{\leftrightarrows}} \langle \wp(\Sigma), \subseteq \rangle$. $\square$

We have:

Theorem 19. *Under Hyps. 2, 3.2 & 4,* $\langle M, \sqsubseteq \rangle \underset{\alpha \circ F \circ \widetilde{\gamma}}{\overset{\widetilde{\alpha} \circ \widetilde{F} \circ \gamma}{\leftrightarrows}} \langle M, \sqsubseteq \rangle$.

Proof.

$$
\begin{array}{lll}
 & \alpha \circ F \circ \widetilde{\gamma}(X) \Longleftrightarrow Y & \\
\Longleftrightarrow & \quad \wr \text{Galois connection } \langle L, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \Longleftrightarrow \rangle \int & \\
 & F \circ \widetilde{\gamma}(X) \leq \gamma(Y) & \\
\Longleftrightarrow & \quad \wr \text{Galois connection } \langle L, \leq \rangle \underset{F}{\overset{\widetilde{F}}{\leftrightarrows}} \langle L, \leq \rangle \int & \\
 & \widetilde{\gamma}(X) \leq \widetilde{F} \circ \gamma(Y) & \\
\Longleftrightarrow & \widetilde{F} \circ \gamma(Y) \geq \widetilde{\gamma}(X) & \wr \text{inverse} \geq \text{of} \leq \int \\
\Longleftrightarrow & \quad \wr \text{Galois connection } \langle L, \geq \rangle \underset{\widetilde{\alpha}}{\overset{\widetilde{\gamma}}{\leftrightarrows}} \langle M, \sqsupseteq \rangle \int & \\
 & \widetilde{\alpha} \circ \widetilde{F} \circ \gamma(Y) \sqsupseteq X & \\
\Longleftrightarrow & X \sqsubseteq \widetilde{\alpha} \circ \widetilde{F} \circ \gamma(Y) & \wr \text{inverse} \sqsubseteq \text{of} \sqsupseteq \int
\end{array}
$$

$\square$

3.6 The Dual Abstract Fixpoint Checking Algorithm

The dual of Alg. 4 is the following:

Algorithm 5

$Y := \widetilde{\alpha}(S);\;\; Go := (I \leq \widetilde{\gamma}(Y));$
while Go **do**
  $Y' := \widetilde{\alpha}(S \wedge \widetilde{F}(\widetilde{\gamma}(Y)));$
  $Go := (Y \neq Y') \,\&\, (I \leq \widetilde{\gamma}(Y'));$
  $Y := Y';$
od;
return if $(I \leq \widetilde{\gamma}(Y))$ ***then*** *true* ***else*** *I don't know;*

Theorem 20. *Under Hyps. 1.1, 2, 3.1 & 4, Alg. 5 is partially correct: if it terminates and returns "true" then* $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$.

Proof. The proof is the order-theoretic dual of the proof of Alg. 4 where the dual of I is S, that of F is $\widetilde{F}$ and that of $\langle \alpha, \gamma \rangle$ satisfying Hyp. 3.2 is $\langle \widetilde{\alpha}, \widetilde{\gamma} \rangle$ satisfying Hyp. 4. Its conclusion is that upon termination while returning *true*, $I \leq \mathit{gfp}^{\leq} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X)$ that is, by Th. 4, $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$. □

3.7 Characterization of Partially Complete Abstractions for Algorithm 5

By Th. 4, the notion of *partial completeness* of Sec. 3.3 is self-dual and A is an invariant for $\langle F, I, S \rangle$ if and only if A is a dual invariant for $\langle \widetilde{F}, S, I \rangle$. Therefore we have:

Theorem 21. *Under Hyps. 1.1, 2, 3.1, & 4, the abstraction* $\langle \widetilde{\alpha}, \widetilde{\gamma} \rangle$ *is partially complete for Alg. 5 if and only if* $\widetilde{\alpha}(L)$ *contains an abstract value* A *such that* $\widetilde{\gamma}(A)$ *is an invariant for* $\langle F, I, S \rangle$.

Proof. The proof of Th. 21 is the order-theoretic dual of the proof of Th. 13 where the dual of I is S, that of F is $\widetilde{F}$ and that of $\langle \alpha, \gamma \rangle$ satisfying Hyp. 3.2 is $\langle \widetilde{\alpha}, \widetilde{\gamma} \rangle$ satisfying Hyp. 4. □

3.8 The Complete Lattice of Minimal Partially Complete Abstractions for Algorithm 5

Theorem 22. *Under Hyps. 1.1, 4 & 3.1, the dual of Th. 17 holds for Alg. 5.*

Proof. The proof of Th. 21 is the order-theoretic dual of the proof of Th. 17 where the dual of I is S, that of F is $\widetilde{F}$ and that of $\langle \alpha, \gamma \rangle$ satisfying Hyp. 3.2 is $\langle \widetilde{\alpha}, \widetilde{\gamma} \rangle$ satisfying Hyp. 4. □

3.9 The Particular Case of Complement Abstraction

Alg. 5 is better known in the important particular case when the following hypotheses 5 below, which scope is local to this Sec. 3.9, hold:

Hypotheses 5 *1.* $\langle L, \leq, 0, 1, \vee, \wedge, \neg\rangle$ *is a complete boolean lattice;*
2. $\langle M, \sqsubseteq, \bot, \top, \sqcup, \sqcap, \backsim\rangle$ *is a complete boolean lattice;*
3. $\langle L, \leq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq\rangle$*;*
4. $\langle L, \leq\rangle \underset{F}{\overset{\widetilde{F}}{\leftrightarrows}} \langle L, \leq\rangle$*;*
5. $\widetilde{F} \triangleq \neg \circ F \circ \neg$, $\widetilde{\alpha} \triangleq \backsim \circ\, \alpha \circ \neg$ *and* $\widetilde{\gamma} \triangleq \neg \circ \gamma \circ \backsim$.

in which case Hyp. 4 holds so that Alg. 5 becomes [19]:

Algorithm 6

$Z := \alpha(\neg S)$; $Go := (I \wedge \gamma(Z) = 0)$;
while Go **do**
  $Z' := \alpha(\neg S \vee F(\gamma(Z)))$;
  $Go := (Z \neq Z')\ \&\ (I \wedge \gamma(Z') = 0)$;
  $Z := Z'$;
od;
return if $(I \wedge \gamma(Z) = 0)$ ***then*** *true* ***else*** *I don't know;*

Corollary 23. *Under Hyp. 5, Alg. 6 is partially correct: if it terminates and returns "true" then* $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$.

Proof. First observe that $\langle L, \geq\rangle \underset{\widetilde{\alpha}}{\overset{\widetilde{\gamma}}{\leftrightarrows}} \langle M, \sqsupseteq\rangle$ since:

$$
\begin{aligned}
& \widetilde{\alpha}(X) \sqsupseteq Y \\
\Longleftrightarrow\ & \backsim(\alpha(\neg(X))) \sqsupseteq Y && \wr\text{def. 5.5 of } \widetilde{\alpha}\int \\
\Longleftrightarrow\ & \alpha(\neg(X))) \sqsubseteq \backsim(Y) && \wr\text{def. complement } \backsim \text{ in Hyp. 5.2}\int \\
\Longleftrightarrow\ & \neg(X)) \leq \gamma(\backsim(Y)) && \wr\text{Galois connection of Hyp. 5.3}\int \\
\Longleftrightarrow\ & \neg \circ \gamma \circ \backsim(Y) \leq X && \wr\text{def. complement } \neg \text{ in Hyp. 5.1}\int \\
\Longleftrightarrow\ & X \geq \widetilde{\gamma}(Y) && \wr\text{def. 5.5 of } \widetilde{\alpha}\int
\end{aligned}
$$

Then we observe that the value of Z in Alg. 6 is that of $\backsim Y$ in Alg. 5. $\square$

3.10 The Adjoined Abstract Fixpoint Checking Algorithm

If follows that Alg. 3 can be used in the abstract to check that $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$ (assuming, as is the case for model-checking, that abstraction/ concretization is computable):

Algorithm 7

$X := \alpha(I);\ Y := \widetilde{\alpha}(S);\ \mathit{Go} := (\gamma(X) \leq S)\ \&\ (I \leq \widetilde{\gamma}(Y));$
while Go **do**
  $X' := \alpha(I \vee F \circ \gamma(X));\ Y' := \widetilde{\alpha}(S \wedge \widetilde{F} \circ \widetilde{\gamma}(Y));$
  $\mathit{Go} := (X \neq X')\ \&\ (Y \neq Y')\ \&\ (\gamma(X') \leq S)\ \&\ (I \leq \widetilde{\gamma}(Y'));$
  $X := X';\ Y := Y';$
od;
return if $(\gamma(X) \leq S) \mid (I \leq \widetilde{\gamma}(Y))$ ***then*** *true* ***else*** *I don't know;*

Theorem 24. *Under Hyps. 1.1, 2, 3.1 & 4, Alg. 7 is partially correct: if it terminates and returns "true" then* $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$.

Proof. The respective values of X and Y after $n \geq 0$ iterations, if ever, are X^n and Y^n as respectively defined in the proofs of Th. 10 and Th. 20. If the algorithm does terminate, then

1. either the loop is never entered so the values of X and Y are respectively $X^0 = \alpha(I)$ and $Y^0 = \widetilde{\alpha}(S)$ such that $\gamma(X) \not\leq S \mid I \not\leq \widetilde{\gamma}(Y)$, in which case Alg. 7 correctly returns *I don't know*;

   or the loop is entered at least once so that upon exit after $n \geq 1$ iterations, we have $X^n = \alpha(I \vee F \circ \gamma(X^n)) \mid Y^n = \widetilde{\alpha}(S \wedge \widetilde{F} \circ \widetilde{\gamma}(Y^n)) \mid \gamma(X^n) \not\leq S \mid I \not\leq \widetilde{\gamma}(Y^n)$.
2. If $\gamma(X^n) \not\leq S \mid I \not\leq \widetilde{\gamma}(Y^n)$ then Alg. 7 correctly returns *I don't know*;

   otherwise, we have $\gamma(X^n) \leq S\ \&\ I \leq \widetilde{\gamma}(Y^n)$ and two cases remain to be considered.
3. If $X^n = \alpha(I \vee F \circ \gamma(X^n))\ \&\ \gamma(X^n) \leq S$, then we conclude as in the proof of Th. 10;
4. if $Y^n = \widetilde{\alpha}(S \wedge \widetilde{F} \circ \widetilde{\gamma}(Y^n))\ \&\ I \not\leq \widetilde{\gamma}(Y^n)$, then we conclude as in the proof of Th. 20. ☐

3.11 The Adjoined Abstract Fixpoint Abstract Checking Algorithm

In program static analysis, one cannot compute γ, $\widetilde{\gamma}$ and $\leq$ and sometimes neither I nor S may even be machine representable. So Alg. 7, which can be useful in model-checking, is of limited interest in program static analysis. In that latter case, the termination condition $(\gamma(X') \leq S)\ \&\ (I \leq \widetilde{\gamma}(Y'))$ must be checked in the abstract, as proposed in Alg. 8 below. This is less precise but is nevertheless correct with the following:

Hypotheses 6 *1.* $\forall X \in L : \gamma \circ \widetilde{\alpha}(X) \leq X$*;*
2. $\forall X \in L : X \leq \widetilde{\gamma} \circ \alpha(X)$*.*

Example 25. Continuing Ex. 11 with $\alpha \triangleq \mathit{post}[h]$, $\gamma \triangleq \widetilde{\mathit{pre}}[h]$, $\widetilde{\alpha} \triangleq \widetilde{\mathit{post}}[h]$ and $\widetilde{\gamma} \triangleq \mathit{pre}[h]$, we have:

$$
\begin{array}{ll}
& \widetilde{\gamma} \circ \alpha(X) \\
= & \wr\text{def. } \widetilde{\gamma} = \mathit{pre}[h] = \boldsymbol{\lambda} X \cdot \{x \mid h(x) \in X\} \text{ and } \alpha = \mathit{post}[h] = \boldsymbol{\lambda} X \cdot \{h(y) \mid y \in X\}\wr \\
& \{x \mid \exists y \in X : h(x) = h(y)\} \\
\supseteq & X \,. \qquad \wr\text{choosing } y = x\wr
\end{array}
$$

In particular for all $X \in L$:

$$
\begin{array}{ll}
& \neg X \subseteq \widetilde{\gamma} \circ \alpha(\neg X) \\
\Longrightarrow & \neg\widetilde{\gamma} \circ \alpha(\neg X) \subseteq X \qquad \wr\text{by contraposition in } L\wr \\
\Longrightarrow & \wr\ \widetilde{\gamma} = \mathit{pre}[h] = \neg \circ \widetilde{\mathit{pre}}[h] \circ \neg = \neg \circ \gamma \circ \neg\wr \\
& \neg \circ \neg \circ \gamma \circ \neg \circ \alpha \circ \neg(X) \subseteq X \\
\Longrightarrow & \wr\ \neg \circ \alpha \circ \neg = \neg \circ \mathit{post}[h] \circ \neg = \widetilde{\mathit{post}}[h] = \widetilde{\alpha} \text{ and } \neg \circ \neg(Y) = Y\wr \\
& \gamma \circ \widetilde{\alpha}(X) \subseteq X \,.
\end{array}
$$

$\square$

Algorithm 8

$X := \alpha(I);\ Y := \widetilde{\alpha}(S);\ \mathit{Go} := (X \sqsubseteq Y);$
while Go **do**
  $X' := \alpha(I) \sqcup \alpha \circ F \circ \gamma(X);\ Y' := \widetilde{\alpha}(S) \sqcap \widetilde{\alpha} \circ \widetilde{F} \circ \widetilde{\gamma}(Y);$
  $\mathit{Go} := (X \neq X')\ \&\ (Y \neq Y')\ \&\ (X' \sqsubseteq Y');$
  $X := X';\ Y := Y';$
od;
return if $X \sqsubseteq Y$ ***then*** *true* ***else*** *I don't know;*

Theorem 26. *Under Hyps. 1.1, 2, 3.1, 4 & 6, Alg. 8 is partially correct: if it terminates and returns "true" then* $\mathit{lfp}^{\leq}\, \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$.

Proof. If the loop ever terminates after $n \geq 0$ iterations then upon exit we have $\alpha(I) \sqsubseteq X^n = X$ and $Y = Y^n \sqsubseteq \widetilde{\alpha}(S)$. So if $X \sqsubseteq Y$ then by Hyp. 6 and monotony, $\gamma(X) \leq \gamma(Y) \leq \gamma \circ \widetilde{\alpha}(S) \leq S$ and $I \leq \widetilde{\gamma} \circ \alpha(I) \leq \widetilde{\gamma}(X) \leq \widetilde{\gamma}(Y)$. So $X \sqsubseteq Y$ implies $\gamma(X) \leq S \,\&\, I \leq \widetilde{\gamma}(Y)$ and the argument used in the proof of Th. 24 concludes the partial correctness proof of Alg. 8. $\square$

Theorem 27. *Under Hyps. 1.1, 2, 3.1, 4 & 6, the abstraction* $\langle \alpha, \gamma\rangle$ *and* $\langle \widetilde{\alpha}, \widetilde{\gamma}\rangle$ *is partially complete for Alg. 4 if and only if either* $\alpha(L)$ *contains an abstract value* A *such that* $\gamma(A)$ *is an invariant for* $\langle F, I, S\rangle$ *or dually* $\widetilde{\alpha}(L)$ *contains an abstract value* $\widetilde{A}$ *such that* $\widetilde{\gamma}(\widetilde{A})$ *is an invariant for* $\langle F, I, S\rangle$.

Proof. The proof is similar to that of Th. 13 or its dual.

Then Def. 14 and Th. 15 are easily generalized in order to characterize the most abstract partially complete abstractions $\langle \alpha, \gamma\rangle$ and $\langle \widetilde{\alpha}, \widetilde{\gamma}\rangle$ for Alg. 8.

Finally, we can apply Alg. 3 to $\langle M, \sqsubseteq\rangle \underset{\alpha\circ F\circ\widetilde{\gamma}}{\overset{\widetilde{\alpha}\circ\widetilde{F}\circ\gamma}{\leftrightarrows}} \langle M, \sqsubseteq\rangle$. We get Alg. 9 below using basic operations performing exclusively on the abstract domain. The correctness of Alg. 9 follows from:

Theorem 28. *Under Hyps. 1.1, 1.2 or 2, 3 and 6, we have* $lfp^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq \gamma(lfp^{\sqsubseteq} \boldsymbol{\lambda} X \cdot \alpha(I) \sqcup \alpha \circ F \circ \widetilde{\gamma}(X))$.

Proof. Let X^{δ}, $\delta \in \mathbb{O}$ and $\widetilde{X}^{\delta}$, $\delta \in \mathbb{O}$ the respective transfinite sequences of iterates for $\boldsymbol{\lambda} X \cdot I \vee F(X)$ and $\boldsymbol{\lambda} X \cdot \alpha(I) \sqcup \alpha \circ F \circ \widetilde{\gamma}(X)$ which by monotony and definition on complete lattices are well-defined, increasing, ultimately stationary and respectively converging to $lfp^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X)$ and $lfp^{\sqsubseteq} \boldsymbol{\lambda} X \cdot \alpha(I) \sqcup \alpha \circ F \circ \widetilde{\gamma}(X)$ as shown in [11]. Let us show by transfinite induction that $\forall \delta \in \mathbb{O} : \widetilde{X}^{\delta} \sqsupseteq \alpha(X^{\delta})$. For the basis, $\widetilde{X}^{0} \triangleq \bot = \alpha(0) = \alpha(X^{0})$. For successor ordinals:

$$
\begin{array}{ll}
 & \widetilde{X}^{\delta+1} \\
= & \alpha(I) \sqcup \alpha \circ F \circ \widetilde{\gamma}(\widetilde{X}^{\delta}) \hfill \wr\text{by def. of the iterates.}\int \\
\sqsupseteq & \alpha(I) \sqcup \alpha \circ F \circ \widetilde{\gamma} \circ \alpha(X^{\delta}) \hfill \wr\text{by ind. hyp. and monotony}\int \\
\sqsupseteq & \alpha(I) \sqcup \alpha \circ F(X^{\delta}) \hfill \wr\text{by Hyp. 6.2 and monotony}\int \\
= & \alpha(I \vee F(X^{\delta})) \hfill \wr\text{by Hyp. 3.2 so that } \alpha \text{ preserves lubs}\int \\
= & X^{\delta+1} \hfill \wr\text{by def. of the iterates.}\int
\end{array}
$$

For limit ordinals $\lambda \in \omega.\mathbb{O}$:

$$
\begin{array}{ll}
 & \widetilde{X}^{\lambda} \\
= & \bigsqcup_{\beta<\lambda} \widetilde{X}^{\beta} \hfill \wr\text{by def. of the iterates.}\int \\
\sqsupseteq & \bigsqcup_{\beta<\lambda} \alpha(X^{\beta}) \hfill \wr\text{ind. hyp. and def. lubs}\int \\
\sqsupseteq & \alpha(\bigvee_{\beta<\lambda} X^{\beta}) \hfill \wr\text{by Hyp. 3.2 so that } \alpha \text{ preserves lubs}\int \\
= & \alpha(X^{\lambda}) \hfill \wr\text{by def. } X^{\lambda}.\int
\end{array}
$$

There exists $\epsilon \in \mathbb{O}$ such that $\alpha(lfp^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X)) = \alpha(X^{\epsilon}) \sqsubseteq \widetilde{X}^{\epsilon} = lfp^{\sqsubseteq} \boldsymbol{\lambda} X \cdot \alpha(I) \sqcup \alpha \circ F \circ \widetilde{\gamma}(X)$ so that by the Galois connection Hyp. 3.2, we conclude that $lfp^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq \gamma(lfp^{\sqsubseteq} \boldsymbol{\lambda} X \cdot \alpha(I) \sqcup \alpha \circ F \circ \widetilde{\gamma}(X))$. □

By duality, we get:

Theorem 29. *Under Hyps. 1.1, 2, 3.1, 4 and 6, we have* $gfp^{\leq} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X) \geq \widetilde{\gamma}(gfp^{\sqsubseteq} \boldsymbol{\lambda} X \cdot \widetilde{\alpha}(S) \sqcap \widetilde{\alpha} \circ \widetilde{F} \circ \gamma(X))$.

Proof. The proof is order-theoretic dual of that of Th. 28 where I is S, F is $\widetilde{F}$, the rôles of $\langle \alpha, \gamma \rangle$ and $\langle \widetilde{\alpha}, \widetilde{\gamma} \rangle$ are exchanged so that Hyp. 6 is self-dual. □

We obtain Alg. 9 below which operates only on the abstract domain:

Algorithm 9

$X := \alpha(I);\ Y := \widetilde{\alpha}(S);\ Go := (X \sqsubseteq Y);$
while Go **do**
  $X' := \alpha(I) \sqcup \alpha \circ F \circ \widetilde{\gamma}(X);\ Y' := \widetilde{\alpha}(S) \sqcap \widetilde{\alpha} \circ \widetilde{F} \circ \gamma(Y);$
  $Go := (X \neq X')\ \&\ (Y \neq Y')\ \&\ (X' \sqsubseteq Y');$
  $X := X';\ Y := Y';$
od;
return if $X \sqsubseteq Y$ ***then*** *true* ***else*** *I don't know;*

Theorem 30. *Under Hyps. 1.1, 2, 3, 4 & 6, Alg. 9 is partially correct: if it terminates and returns "true" then* $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S$.

Proof. 1. If the loop is never entered then Alg. 9 terminates with $X \not\sqsubseteq Y$ so the returned result *I don't know* is correct;

Otherwise the loop is entered at least once so that if it is ever exited, we have $X = \alpha(I) \sqcup \alpha \circ F \circ \widetilde{\gamma}(X) \mid Y = \widetilde{\alpha}(S) \sqcap \widetilde{\alpha} \circ \widetilde{F} \circ \gamma(Y) \mid X \not\sqsubseteq Y$.

2. If $X \not\sqsubseteq Y$, the returned result *I don't know* is correct;

   Otherwise $X \sqsubseteq Y$ and two cases remain to be considered;
3. if $X = \alpha(I) \sqcup \alpha \circ F \circ \widetilde{\gamma}(X)$ and $X \sqsubseteq Y$ then by Th. 28 and monotony, we have $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \leq \gamma(\mathit{lfp}^{\sqsubseteq} \boldsymbol{\lambda} X \cdot \alpha(I) \sqcup \alpha \circ F \circ \widetilde{\gamma}(X)) \leq \gamma(X) \leq \gamma(Y)$ $\leq \gamma(\widetilde{\alpha}(S) \leq S$ (by Hyp. 6.1), as required;
4. if $Y = \widetilde{\alpha}(S) \sqcap \widetilde{\alpha} \circ \widetilde{F} \circ \gamma(Y)$ then by Th. 29 and monotony, we have dually $\mathit{gfp}^{\leq} \boldsymbol{\lambda} X \cdot S \wedge \widetilde{F}(X) \geq \widetilde{\gamma}(\mathit{gfp}^{\sqsubseteq} \boldsymbol{\lambda} X \cdot \widetilde{\alpha}(S) \sqcap \widetilde{\alpha} \circ \widetilde{F} \circ \gamma(X)) \geq \widetilde{\gamma}(Y) \geq \widetilde{\gamma}(X) \geq$ $\widetilde{\gamma}(\alpha(I)) \geq I$ (by Hyp. 6.2), as required. □

3.12 On Termination

Observe that, due to classical undecidable results for program analysis, if the abstract fixpoint checking algorithms of Sec. 3 are required to always terminate (e.g. by choosing a coarse enough abstraction or by enforcing convergence by widening/narrowing [12]) then there must be some programs for which the algorithm terminates and returns *I don't know.* This can be either because the program is incorrect (i.e. $\mathit{lfp}^{\leq} \boldsymbol{\lambda} X \cdot I \vee F(X) \not\leq S$) or because the abstraction is too imprecise to prove its correctness.

4 Conclusion

The traditional universal model checking Alg. 2 [3, 32] is the dual of the traditional algorithm for program analysis [7, 10, 12] (with no abstraction i.e. $\langle \alpha, \gamma \rangle$ is the identity). Both algorithms are logically equivalent (Th. 4) although, because of computer resources limitations, one may fail while the other succeeds. We have introduced a new Alg. 3 combining these Algs. 1 and 2 which parallel version is logically equivalent (Th. 10) but more time efficient than both algorithms.

When considering infinite-sate systems, model-checking must resort to abstraction, which is always the case in program static analysis. Abstract interpretation [7, 10, 12] yields the abstract Alg. 4 and its dual Alg. 5 (with its particular case Alg. 6 used in universal abstract model checking [19]) which are both sound (Th. 12, 20 and Col. 23) and logically equivalent. Again their (parallel) combination in algorithm 7 is possible, sound (Th. 24) and more efficient. Finally Algs. 4, 5, 6 and 7 compute abstract fixpoints but use a concrete specification checking (e.g. $\gamma(X) \leq S$ for Alg. 4) so are hardly usable for program static analysis. In this last case one must resort to Algs. 8 or 9, or their parallel versions, which operate only in the abstract.

In model-checking one is deeply interested in partially complete abstractions which, despite the loss of information inherent to approximate abstract interpretations, always yield an affirmative answer when the specification is correct and the checking algorithm does terminate. Would soundness be required only, but not completeness (i.e. including termination, not considered here), abstract universal model-checking would be nothing more than classical transition system analysis by abstract interpretation [12] (and existential model checking its mere dual).

We have characterized these partially complete abstractions and shown for both Algs. 4 and 5 that any partially complete abstract domain must contain the exact abstraction of an invariant, as computed by e.g. by Algs. 1 and 2 respectively (Th. 13 and 21 respectively).

In practice, this means that no full automation of the abstraction process is possible for infinite-state transition systems (but for particular cases of limited interest such as specific classes of program specifications), since finding or computing the proper abstraction always boils down to making a full correctness proof. This appears to be a fundamental restriction to this popular approach [1, 2, 5, 20, 24, 25, 28, 33], and shows that some human assistance is ultimately necessary as long recognized in the use of abstract interpretation to design program static analyzers manually or with interactive computer assistance [29].

References

[1] P.A. Abdulla, A. Annichini, S. Bensalem, A. Bouajjani, P. Habermehl, and L. Lakhnech. Verification of infinite-state systems by combining abstraction and reachability analysis. In N. Halbwachs and D. Peled, editors, *Proceedings of the Eleventh International Conference on Computer Aided Verification, CAV '99*, Trento, Italy, Lecture Notes in Computer Science 1633, pages 146–159. Springer-Verlag, Berlin, Germany, 6–10 July 1999.

[2] S. Bensalem, Y. Lakhnech, and S. Owre. Computing abstractions of infinite state systems compositionally and automatically. In A.J. Hu and M.Y. Vardi, editors, *Proceedings of the Tenth International Conference on Computer Aided Verification, CAV '98*, Vancouver, British Columbia, Canada, Lecture Notes in Computer Science 1427, pages 319–331. Springer-Verlag, Berlin, Germany, 28 June – 2 July 1998.

[3] E.M. Clarke and E.A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *IBM Workshop on Logics of Programs*, Lecture Notes in Computer Science 131. Springer-Verlag, Berlin, Germany, May 1981.

[4] E.M. Clarke, O. Grumberg, and D.E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, september 1994.

[5] M. Col n and T.E. Uribe. Generating finite-state abstractions of reactive systems using decision procedures. In A.J. Hu and M.Y. Vardi, editors, *Proceedings of the Tenth International Conference on Computer Aided Verification, CAV '98*, Vancouver, British Columbia, Canada, Lecture Notes in Computer Science 1427, pages 293–304. Springer-Verlag, Berlin, Germany, 28 June – 2 July 1998.

[6] P. Cousot. *M thodes it ratives de construction et d'approximation de points fixes d'op rateurs monotones sur un treillis, analyse s mantique de programmes.* Th se d' tat s sciences math matiques, Universit scientifique et m dicale de Grenoble, Grenoble, 21 mars 1978.

[7] P. Cousot. Semantic foundations of program analysis. In S.S. Muchnick and N.D. Jones, editors, *Program Flow Analysis: Theory and Applications*, chapter 10, pages 303–342. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, United States, 1981.

[8] P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Electronic Notes in Theoretical Computer Science*, 6, 1997. URL: `http://www.elsevier.nl/locate/entcs/volume6.html`, 25 pages.

[9] P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theoretical Computer Science*, To appear (Preliminary version in [8]).

[10] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, New York, United States.

[11] P. Cousot and R. Cousot. Constructive versions of Tarski's fixed point theorems. *Pacific Journal of Mathematics*, 82(1):43–57, 1979.

[12] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 269–282, San Antonio, Texas, 1979. ACM Press, New York, New York, United States.

[13] P. Cousot and R. Cousot. Induction principles for proving invariance properties of programs. In D. N el, editor, *Tools & Notions for Program Construction*, pages 43–119. Cambridge University Press, Cambridge, United Kindom, 1982.

[14] P. Cousot and R. Cousot. Abstract interpretation and application to logic programs. *Journal of Logic Programming*, 13(2–3):103–179, 1992. (The editor of Journal of Logic Programming has mistakenly published the unreadable galley proof. For a correct version of this paper, see `http://www.di.ens.fr/~cousot`.).

[15] P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, invited paper. In M. Bruynooghe and M. Wirsing, editors, *Proceedings of the International Workshop Programming Language Implementation and Logic Programming, PLILP '92*, Leuven, Belgium, 13–17 August 1992, Lecture Notes in Computer Science 631, pages 269–295. Springer-Verlag, Berlin, Germany, 1992.

[16] P. Cousot and R. Cousot. Parallel combination of abstract interpretation and model-based automatic analysis of software. In R. Cleaveland and D. Jackson, editors, *Proceedings of the First ACM SIGPLAN Workshop on Automatic Analysis of Software, AAS '97*, pages 91–98, Paris, France, January 1997. ACM Press, New York, New York, United States.

[17] P. Cousot and R. Cousot. Refining model checking by abstract interpretation. *Automated Software Engineering*, 6:69–95, 1999.

[18] P. Cousot and R. Cousot. Temporal abstract interpretation. In *Conference Record of the Twentyseventh Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 12–25, Boston, Massachusetts, January 2000. ACM Press, New York, New York, United States.

[19] D. Dams, O. Grumberg, and R. Gerth. Abstract interpretation of reactive systems. *ACM Transactions on Programming Languages and Systems*, 19(2):253–291, 1997.

[20] S. Das, D.L. Dill, and S. Park. Experience with predicate abstraction. In N. Halbwachs and D. Peled, editors, *Proceedings of the Eleventh International Conference on Computer Aided Verification, CAV '99*, Trento, Italy, Lecture Notes in Computer Science 1633, pages 160–171. Springer-Verlag, Berlin, Germany, 6–10 July 1999.

[21] R.W. Floyd. Assigning meaning to programs. In J.T. Schwartz, editor, *Proceedings of the Symposium in Applied Mathematics*, volume 19, pages 19–32. American Mathematical Society, Providence, Rhode Island, United States, 1967.

[22] R. Giacobazzi, F. Ranzato, and F. Scozzari. Complete abstract interpretations made constructive. In L. Brim, J. Gruska, and J. Zlatuska, editors, *Proceedings of the Twentythird International Symposium on Mathematical Foundations of Computer Science, MFCS'98*, volume 1450 of *Lecture Notes in Computer Science*, pages 366–377. Springer-Verlag, Berlin, Germany, 1998.

[23] R. Giacobazzi, F. Ranzato, and F. Scozzari. Making abstract intrepretations complete. *Journal of the Association for Computing Machinary*, 2000. To appear.

[24] S. Graf and C. Loiseaux. A tool for symbolic program verification and abstraction. In C. Courcoubetis, editor, *Proceedings of the Fifth International Conference on Computer Aided Verification, CAV '93*, Elounda, Greece, Lecture Notes in Computer Science 697, pages 71–84. Springer-Verlag, Berlin, Germany, 28 June –1 July 1993.

[25] S. Graf and H. Sa di. Construction of abstract state graphs with PVS. In O. Grumberg, editor, *Proceedings of the Ninth International Conference on Computer Aided Verification, CAV '97*, Haifa, Israel, Lecture Notes in Computer Science 1254, pages 72–83. Springer-Verlag, Berlin, Germany, 22–25 July 1997.

[26] T.A. Henzinger, O. Kupferman, and S. Qadeer. From *Pre*-historic to *Post*-modern symbolic model checking. In A.J. Hu and M.Y. Vardi, editors, *Proceedings of the Tenth International Conference on Computer Aided Verification, CAV '98*, Vancouver, British Columbia, Canada, Lecture Notes in Computer Science 1427, pages 195–206. Springer-Verlag, Berlin, Germany, June /July 1998.

[27] Y. Kesten and A. Pnueli. Modularization and abstraction: The keys to formal verification. In L. Brim, J. Gruska, and J. Zlatuska, editors, *Twentythird International Symposium on Mathematical Foundations of Computer Science 1998*, Lecture Notes in Computer Science 1450, pages 54–71. Springer-Verlag, Berlin, Germany, 1998.

[28] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6(1), 1995.

[29] D. Monniaux. R alisation m canis e d'interpr teurs abstraits. Rapport de stage, DEA "S mantique, Preuve et Programmation", July 1998.

[30] J.H. Morris and B. Wegbreit. Sungoal induction. *Communications of the Association for Computing Machinary*, 20(4):209–222, April 1977.

[31] P. Naur. Proofs of algorithms by general snapshots. *BIT*, 6:310–316, 1966.

[32] J.-P. Queille and J. Sifakis. Verification of concurrent systems in CESAR. In *Proceedings of the International Symposium on Programming*, Lecture Notes in Computer Science 137, pages 337–351. Springer-Verlag, Berlin, Germany, 1982.

[33] H. Sa di and N. Shankar. Abstract and model check while you prove. In N. Halbwachs and D. Peled, editors, *Proceedings of the Eleventh International Conference on Computer Aided Verification, CAV '99*, Trento, Italy, Lecture Notes in Computer Science 1633, pages 443–454. Springer-Verlag, Berlin, Germany, 6–10 July 1999.

[34] A. Tarski. A lattice theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–310, 1955.