Automated Software Engineering, 6, 69–95 (1999)

Refining Model Checking by Abstract Interpretation

PATRICK COUSOT cousot@dmi.ens.fr,
RADHIA COUSOT rcousot@lix.polytechnique.fr
École Normale Supérieure, DMI, 45, rue d'Ulm, 75230 Paris cedex 05, France
CNRS & École Polytechnique, LIX, 91440 Palaiseau cedex, France

Abstract. Formal methods combining abstract interpretation and model-checking have been considered for automated analysis of software.

In abstract model-checking, the semantics of an infinite transition system is abstracted to get a finite approximation on which temporal-logic/μ-calculus model-checking can be directly applied.

The paper proposes two improvements of abstract model-checking which can be applied to infinite abstract transition systems:

– A new combination of forwards and backwards abstract fixed-point model-checking computations for universal safety. It computes a more precise result than that computed by conjunction of the forward and backward analyses alone, without needing to refine the abstraction;

– When abstraction is unsound (as can happen in minimum/maximum path-length problems), it is proposed to use the partial results of a classical combination of forward and backward abstract interpretation analyses for universal safety in order to reduce, on-the-fly, the concrete state space to be searched by model-checking.

Keywords: model-checking, abstract interpretation, static analysis, transition system, universal safety.

1. Introduction

In the design and development of software using model-based automatic analysis – such as model-checking or state space exploration – one is confronted with high complexity for very large systems and undecidability as soon as one has to consider infinite sets of states. Consequently, all properties of all systems cannot be automatically verified in finite or reasonable time. Some form of approximation has to be considered. For example syntax-driven proof techniques ultimately rely on some form of assistance from the user. Although one can prove very precise assertions with an interactive automatic theorem prover, the technique is necessarily approximate in the sense that the output of the theorem-prover may not be understandable by the user and/or the user's answers may mislead the theorem-prover into dead-ends. Model checking (Clarke et al., 1983) places no restriction on verifiable properties (CTL*, μ-calculus and the like) but consider only finite state systems (or finite abstraction of infinite systems). Program analysis by abstract interpretation (Cousot and Cousot, 1977, 1979, Cousot, 1996) places no restriction on systems/programming languages (which can be imperative, functional, logic, object-oriented, parallel) but places restrictions on verifiable properties since abstract properties are necessarily approximate. Both model-checking and abstract interpretation have benefited from mutual cross-fertilization. In particular model-checking can now consider infinite-state systems whereas in abstract interpretation it is common to consider properties significantly more complex than safety/invariance (see e.g. (Dams et al., 1997, Dill and Wong-Toi, 1995, Fernandez, 1993, Halbwachs, 1994) and (Steffen, 1991)). We would like to consider further combinations of abstract interpretation and universal safety model-checking.

Reduction by abstraction consists in approximating infinite or very large finite transition systems by finite ones, on which existing algorithms designed for finite automata are directly applicable. This semi-verification idea was first introduced by (Clarke et al., 1992) and progressively refined to cope with wider classes of temporal-logic (Kelb, 1994, Dams et al., 1997, Cleaveland et al., 1995) or μ-calculus formulæ (Graf and Loiseaux, 1993, Loiseaux et al., 1995, Cridlig, 1995, Cridlig, 1996). We extend this to abstract transition systems which are infinite. The algorithms designed for universal safety analysis of finite transition systems can be simply extended to the infinite case. One can use abstract interpretation techniques such as widening and narrowing to enforce, on the fly, the convergence of fixpoint computations. It is known in abstract interpretation that this is not as precise as could it be (Cousot, 1978, Cousot and Cousot, 1992a). Hence we propose a combination of forward and backward upper-approximate universal safety checking which is more precise than the mere conjunction of these forward and backward analyses (which are indeed equivalent for finite states).

We also suggest another possible interaction between abstract interpretation and model-based automatic analysis of infinite systems (Cousot, 1995). It is based on the remark that although the transition system is infinite, all behaviors considered in practice may be finite e.g. when there is a termination requirement, or more generally a liveness requirement excluding infinite behaviors. In this case, abstract interpretation may be used, on the infinite state system, to eliminate the impossible potentially infinite behaviors. In the favorable case, this preliminary analysis by abstract interpretation may be used to restrict the states which must be explored to a finite number. Even in the case of finite but very large state spaces, the method can be useful to reduce the part of the state graph which need to be explored for verification, before this verification or better in parallel with it, so as to avoid additional costs in time.

2. Definitions

A *poset* $\langle L, \sqsubseteq\rangle$ a set L with a partial order $\sqsubseteq$ (that is a reflexive, antisymmetric and transitive binary relation on L). A *complete partial order* (cpo) $\langle L, \sqsubseteq, \bot, \sqcup\rangle$ is a poset $\langle L, \sqsubseteq\rangle$ such that $\bot$ is the infimum and increasing chains $x_0 \sqsubseteq x_1 \sqsubseteq \ldots$ of elements of L have a least upper bound (lub) $\bigsqcup_{i\geq 0} x_i$.

A map $\mathcal{F} \in L \mapsto L$ of L into L is *monotonic* (written $\mathcal{F} \in L \overset{m}{\longmapsto} L$) if and only if:

$$\forall x, y \in L : x \sqsubseteq y \Longrightarrow \mathcal{F}(x) \sqsubseteq \mathcal{F}(y) .$$

If $\mathcal{F} \in L \overset{m}{\longmapsto} L$ is a monotonic map of L into L and $m \sqsubseteq \mathcal{F}(m)$ then $\operatorname{lfp}_m^{\sqsubseteq} \mathcal{F}$ denotes the $\sqsubseteq$*-least fixpoint of* $\mathcal{F}$ *which is* $\sqsubseteq$*-greater than or equal to* m:

$$\begin{aligned} \mathcal{F}(\operatorname{lfp}_m^{\sqsubseteq} \mathcal{F}) &= \operatorname{lfp}_m^{\sqsubseteq} \mathcal{F}, \\ m &\sqsubseteq \operatorname{lfp}_m^{\sqsubseteq} \mathcal{F}, \\ (m \sqsubseteq x) \wedge (\mathcal{F}(x) = x) &\Longrightarrow \operatorname{lfp}_m^{\sqsubseteq} \mathcal{F} \sqsubseteq x . \end{aligned}$$

$\mathrm{lfp}^{\sqsubseteq} \mathcal{F} \triangleq \mathrm{lfp}^{\sqsubseteq}_{\perp} \mathcal{F}$ is the least fixpoint of $\mathcal{F}$. The greatest fixpoint (gfp) is defined dually, replacing $\sqsubseteq$ by its inverse $\sqsupseteq$, the infimum $\perp$ by the supremum $\top$, the lub $\sqcup$ by the greatest lower bound (glb) $\sqcap$, etc.

As a generalization of Kleene/Knaster/Tarski fixpoint theorem, the transfinite iteration sequence is:

$$\begin{aligned} \bar{\mathcal{F}}^{0} &\triangleq m, \\ \bar{\mathcal{F}}^{\delta+1} &\triangleq \mathcal{F}(\bar{\mathcal{F}}^{\delta}) \quad \text{for successor ordinals,} \\ \bar{\mathcal{F}}^{\lambda} &\triangleq \bigsqcup_{\delta<\lambda} \bar{\mathcal{F}}^{\delta} \quad \text{for limit ordinals .} \end{aligned}$$

This increasing sequence is ultimately stationary and converges to $\mathrm{lfp}^{\sqsubseteq}_{m} \mathcal{F}$. This directly leads to an iterative algorithm which is finitely convergent when L satisfies the ascending chain condition (ACC)[1].

The *complement* $\neg P$ of a subset $P \subseteq S$ of a set S is $\{s \in S \mid s \notin P\}$. The *left-restriction* $P \rceil t$ of a relation t on S to $P \subseteq S$ is $\{\langle s,\ s'\rangle \in t \mid s \in P\}$. The *composition* of relations is $t \circ r \triangleq \{\langle s,\ s''\rangle \mid \exists s' \in S : \langle s,\ s'\rangle \in t \wedge \langle s',\ s''\rangle \in r\}$. The *iterates* of the relation t are defined inductively by:

$$\begin{aligned} t^{0} &\triangleq 1_S \triangleq \{\langle s,\ s\rangle \mid s \in S\} && \text{(that is identity on the set } S\text{),} \\ \text{and } t^{n+1} &\triangleq t \circ t^{n} = t^{n} \circ t, && \text{for } n \geq 0. \end{aligned}$$

The *reflexive transitive closure* $t^{\star}$ of the relation t is:

$$t^{\star} \triangleq \bigcup_{n \geq 0} t^{n} .$$

3. Framework

The considered (real-time) concurrent systems are assumed to be modeled by a *transition system*, that is tuple $\langle S,\ t,\ I,\ F\rangle$ where S is the set of *states*, $t \subseteq S \times S$ is the *transition relation*, $I \subseteq S$ is the set of *initial states* and $F \subseteq S$ is the set of *final states*. There is no finiteness restriction on the set S of states. Moreover initial and final states must be understood in a broad sense. For a terminating program this can be the states in which execution can start and end. For a non-terminating process this can be respectively the states in which a resource is requested and those in which it has later been allocated. For simplicity, we assume that initial and final states are disjoint ($I \cap F = \emptyset$). An example of transition system is given in Figure 1. Such transition systems have been used to introduce abstract interpretation in a language independent way, since they model small-step operational semantics of programs (Cousot and Cousot, 1979).

The *pre-image* $\mathrm{pre}[t]\,P$ of a set $P \subseteq S$ of states by a transition relation t is the set of states from which it is possible to reach a state in P by a transition t:

$$\mathrm{pre}[t]\,P \triangleq \{s \mid \exists s' : \langle s,\ s'\rangle \in t \wedge s' \in P\} .$$

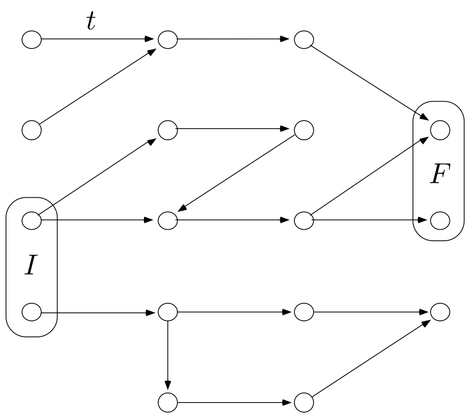
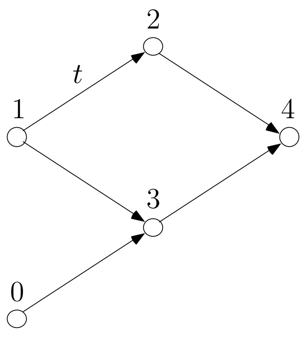


Figure 1. A (finite) transition system (○: state, ⟶▶: transition)



$$\begin{aligned}
\mathrm{pre}[t]\{3\} &= \{0, 1\}\\
\widetilde{\mathrm{pre}}[t]\{3\} &= \{0, 4\}\\
\mathrm{post}[t]\{1\} &= \{2, 3\}\\
\widetilde{\mathrm{post}}[t]\{1\} &= \{0, 1, 2\}
\end{aligned}$$

Figure 2. (Dual) pre- and post-images (○: state, ⟶▶: transition)

The *dual pre-image* $\widetilde{\mathrm{pre}}[t]\,P$ is the set of states from which any transition, if any, must lead to a state in P:

$$\widetilde{\mathrm{pre}}[t]\,P \;\stackrel{\triangle}{=}\; \{s \mid \forall s' : \langle s,\ s'\rangle \in t \Longrightarrow s' \in P\}\ .$$

The *post-image* $\mathrm{post}[t]\,P$ is the set of states which are reachable from $P \subseteq S$ by a transition t:

$$\mathrm{post}[t]\,P \;\stackrel{\triangle}{=}\; \{s' \mid \exists s : s \in P \wedge \langle s,\ s'\rangle \in t\}\ .$$

The *dual post-image* $\widetilde{\mathrm{post}}[t]\,P$ is the set of states which can only be reached, if ever possible, by a transition t from P:

$$\widetilde{\mathrm{post}}[t]\,P \;\stackrel{\triangle}{=}\; \neg\,\mathrm{post}[t](\neg P) \;=\; \{s' \mid \forall s : \langle s,\ s'\rangle \in t \Longrightarrow s \in P\}\ .$$

This is illustrated in Figure 2. We have the fixpoint characterizations (see e.g. (Cousot, 1978), (Cousot, 1981)):

$$\begin{aligned}
\text{pre}[t^\star]\,F &= \text{lfp}^{\subseteq}\,\lambda X \bullet F \cup \text{pre}[t]\,X &&= \text{lfp}^{\subseteq}_{F}\,\lambda X \bullet X \cup \text{pre}[t]\,X,\\
\widetilde{\text{pre}}[t^\star]\,F &= \text{gfp}^{\subseteq}\,\lambda X \bullet F \cap \widetilde{\text{pre}}[t]\,X &&= \text{gfp}^{\subseteq}_{F}\,\lambda X \bullet X \cap \widetilde{\text{pre}}[t]\,X,\\
\text{post}[t^\star]\,I &= \text{lfp}^{\subseteq}\,\lambda X \bullet I \cup \text{post}[t]\,X &&= \text{lfp}^{\subseteq}_{I}\,\lambda X \bullet X \cup \text{post}[t]\,X,\\
\widetilde{\text{post}}[t^\star]\,I &= \text{gfp}^{\subseteq}\,\lambda X \bullet I \cap \widetilde{\text{post}}[t]\,X &&= \text{gfp}^{\subseteq}_{I}\,\lambda X \bullet X \cap \widetilde{\text{post}}[t]\,X\ .
\end{aligned}$$

ALGORITHM 1 *The invariance/safety property, post$[t^\star]\,I \subseteq P$, can be checked by the following well-known fixpoint computation procedure (Cousot and Cousot, 1977, Cousot and Cousot, 1979, Cousot, 1981):*

```
function f-ai1 (I);
   X := I;
   repeat
      Y := X;
      X := Y ∪ post[t] Y;
   until X = Y;
   return X;
function f-mc1 (I, P);
   return f-ai1 (I) ⊆ P.
```

The correctness of this algorithm follows directly from the Kleene/Knaster/Tarski fixpoint theorem, stating that:

$$\begin{aligned}
\text{lfp}^{\subseteq}_{I}\,\lambda X \bullet X \cup post[t]\,X &= \bigcup_{n\geq 0} X^n, && \textit{where:}\\
X^0 &= I && \textit{and}\\
X^{n+1} &= X^n \cup post[t]\,X^n\ .
\end{aligned}$$

ALGORITHM 2 *Since post$[t^\star]\,I \subseteq P$ if and only if pre$[t^\star]\,\neg P \subseteq \neg I$, the forward model-checking procedure f-mc1 is equivalent*[2] *to the following backward one (Clarke and Emerson, 1981, Queille and Sifakis, 1982, Clarke et al., 1983):*

```
function b-ai1 (P);
   X := ¬P;
   repeat
      Y := X;
      X := Y ∪ pre[t] Y;
   until X = Y;
   return X;
function b-mc1 (I, P);
   return b-ai1 (P) ⊆ ¬I.
```

The state space exploration procedures “f-mc1” and “b-mc1” are effective for finite set of states only. During the last few years, they became practical and widely usable (Burch et al., 1992, Henzinger et al., 1992, Daws et al., 1996) thanks to *symbolic* implementations using compact symbolic formula representation of (the characteristic function of) sets of states. For example, the symbolic formula can be encoded by BDDs (Akers, 1978, Bryant, 1986) or by affine inequality relations (Cousot and Halbwachs, 1978). So it is understood that such symbolic representations of sets of states should be used in “f-mc1” and “b-mc1”.

4. Abstract Interpretation Based Model Checking

Abstract interpretation is a theory of semantic approximation (Cousot, 1996) where “approximation” means logical implication i.e. subsets of states inclusion. Here, the semantics to be approximated is the *forward collecting semantics* $\mathrm{post}[t^\star]\,I$, the *backward collecting semantics* $\mathrm{pre}[t^\star]\,F$ (Cousot, 1978, Cousot and Cousot, 1979) or $\mathrm{post}[t^\star]\,I \cap \mathrm{pre}[t^\star]\,F$ that is the set $\mathrm{post}[t^\star]\,I$ of descendants of the initial states I which are in the set $\mathrm{pre}[t^\star]\,F$ of ancestors of the final states F (Cousot, 1978, Cousot and Cousot, 1992a). We briefly recall how the upper-approximations D of $\mathrm{post}[t^\star]\,I$ and A of $\mathrm{pre}[t^\star]\,F$ can be automatically computed by abstract interpretation. The upper/over approximation U of $\mathrm{post}[t^\star]\,I \cap \mathrm{pre}[t^\star]\,F$ will be considered in Section 5.2. We only consider upper approximations (U is an upper approximation of E meaning $E \subseteq U$) since lower/under approximations (L is a lower approximation of E meaning $L \subseteq E$) are dual, both approximations being useful in abstract interpretation (see e.g. (Dams et al., 1997) and (Dill and Wong-Toi, 1995) in the context of abstract model checking).

4.1. *Forward Abstract Interpretation Based Model Checking*

In order to obtain an upper approximation D of the forward collecting semantics $\mathrm{post}[t^\star]\,I = \mathrm{lfp}^{\subseteq}\,\lambda X \bullet I \cup \mathrm{post}[t]\,X = \mathrm{lfp}^{\subseteq}_{I}\,\lambda X \bullet X \cup \mathrm{post}[t]\,X$ one considers a *Galois connection*[3]:

$$\langle \wp(S),\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L,\ \sqsubseteq\rangle,$$

that is, by definition, a pair of maps $\alpha \in \wp(S) \mapsto L$ and $\gamma \in L \mapsto \wp(S)$ from the powerset $\wp(S)$ ordered by subset inclusion $\subseteq$ into the poset $\langle L,\ \sqsubseteq\rangle$ of abstract properties[4] partially ordered by $\sqsubseteq$ such that:

$$\forall P \in \wp(S) : \forall Q \in L : \alpha(P) \sqsubseteq Q \Longleftrightarrow P \subseteq \gamma(Q)\ .$$

EXAMPLE: Given a relation r on a set S, a classical example of Galois connection is:

$$\langle \wp(S),\ \subseteq\rangle \underset{\mathrm{pre}[r]}{\overset{\widetilde{\mathrm{post}}[r]}{\leftrightarrows}} \langle \wp(S),\ \subseteq\rangle\ .$$

The same way, we have:

$$\langle \wp(S),\ \subseteq\rangle \underset{\mathrm{post}[r]}{\overset{\widetilde{\mathrm{pre}}[r]}{\leftrightarrows}} \langle \wp(S),\ \subseteq\rangle\ .$$

For an application proving the equivalence of the model-checking algorithms "f-mc1" and "b-mc1", we have post$[t^\star]\, I \subseteq P$ if and only if $I \subseteq \widetilde{\text{pre}}[t^\star]\, P$ that is, by definition of $\widetilde{\text{pre}}$, $I \subseteq \neg\, \text{pre}[t^\star]\, \neg P$ or equivalently $\text{pre}[t^\star]\, \neg P \subseteq \neg I$. □

In a Galois connection $\langle \wp(S),\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L,\ \sqsubseteq\rangle$, α preserves existing lubs and γ preserves glbs hence both are monotonic. Moreover any concrete property $P \in \wp(S)$ has a best (i.e. most precise) upper approximation $\alpha(P)$ in L, such that $P \subseteq \gamma(\alpha(P))$. When α is surjective (or equivalently γ is injective or $\alpha \circ \gamma = 1_L$ is the identity on L) we have a *Galois insertion* and we write $\langle \wp(S),\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L,\ \sqsubseteq\rangle$. In this case, the poset $\langle L,\ \sqsubseteq\rangle$ is necessarily a complete lattice $\langle L,\ \sqsubseteq,\ \bot,\ \top,\ \sqcup,\ \sqcap\rangle$ with $\alpha(\wp(S)) = L$. $\alpha(P)$ should be machine-representable which, in general, may not be the case of P.

The appropriate choice of the abstract domain L is problem dependent. The design and composition of convenient abstract domains has been extensively developed in the abstract interpretation literature and will not be further considered here. Often the abstraction can be defined by stages. For example one can consider a first abstraction $\langle \wp(S),\ \subseteq\rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle L,\ \sqsubseteq\rangle$ for a class of program properties which is then composed with a second problem-specific abstraction $\langle L,\ \sqsubseteq\rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle M,\ \preceq\rangle$. The correctness follows from the fact that Galois connection compose: $\langle \wp(S),\ \subseteq\rangle \underset{\alpha_2 \circ \alpha_1}{\overset{\gamma_1 \circ \gamma_2}{\leftrightarrows}} \langle M,\ \preceq\rangle$. An abstraction can also be parameterized so as to allow for problem dependent instantiations, as shown by the following example.

EXAMPLE: (Clarke et al., 1992), (Cleaveland et al., 1995), (Dams et al., 1997), (Jackson, 1994) and others implicitly consider a restricted form of abstraction given by the Galois connection $\langle \wp(S),\ \subseteq\rangle \underset{\alpha[h]}{\overset{\gamma[h]}{\leftrightarrows}} \langle \wp(S^\sharp),\ \subseteq\rangle$, where S is the set of concrete states, $S^\sharp$ is the set of abstract states, $\alpha[h](X) \stackrel{\triangle}{=} \{h(x) \mid x \in X\}$ and $\gamma[h](Y) \stackrel{\triangle}{=} \{x \mid h(x) \in Y\}$ where $h \in S \mapsto S^\sharp$ is the state approximation mapping. If h is surjective (as assumed e.g. in (Jackson, 1994)), then so is $\alpha[h]$ whence $\langle \wp(S),\ \subseteq\rangle \underset{\alpha[h]}{\overset{\gamma[h]}{\leftrightarrows}} \langle \wp(S^\sharp),\ \subseteq\rangle$. However this specific form of abstraction is restrictive and cannot e.g. account for relational analyses such as (Halbwachs et al., 1994). □

Once an abstraction $\langle \alpha,\ \gamma\rangle$ has been chosen, we then use the fact that if $\langle M,\ \preceq,\ 0,\ \vee\rangle$ is a cpo, the pair $\langle \alpha,\ \gamma\rangle$ is a Galois connection $\langle M,\ \preceq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L,\ \sqsubseteq\rangle$, $\mathcal{F} \in M \overset{\text{m}}{\longmapsto} M$ and $\mathcal{F}^\sharp \in L \overset{\text{m}}{\longmapsto} L$ are monotonic and

$$\begin{array}{lll} & & \forall y \in L : \alpha \circ \mathcal{F} \circ \gamma(y) \preceq \mathcal{F}^\sharp(y) \\ & \text{or else} & \forall x \in M : \alpha \circ \mathcal{F}(x) \sqsubseteq \mathcal{F}^\sharp \circ \alpha(x) \\ & \text{or else} & \forall y \in L : \mathcal{F} \circ \gamma(y) \preceq \gamma \circ \mathcal{F}^\sharp(y) \\ \text{then} & & \\ & & \text{lfp}^{\preceq}\, \mathcal{F} \ \preceq\ \gamma(\text{lfp}^{\sqsubseteq}\, \mathcal{F}^\sharp) \\ & \text{and equivalently} & \alpha(\text{lfp}^{\preceq}\, \mathcal{F}) \ \sqsubseteq\ \text{lfp}^{\sqsubseteq}\, \mathcal{F}^\sharp, \end{array}$$

see (Cousot and Cousot, 1979).

4.1.1. Abstract Domains Satisfying the Ascending Chain Condition When the abstract domain satisfies the ascending chain condition (such as e.g. the linear equality relations of (Karr, 1976)), the fixpoints can be computed iteratively.

ALGORITHM **3** *The model-checking algorithm "f-mc1" can be approximated using an abstract domain* $\langle L, \sqsubseteq \rangle$*, a Galois connection* $\langle \wp(S), \subseteq \rangle \xleftarrow[\alpha]{\gamma} \langle L, \sqsubseteq \rangle$*, an abstract initial state predicate* $I^\sharp$ *such that* $\alpha(I) \sqsubseteq I^\sharp$ *and an abstract forward predicate transformer* $\mathcal{F}^\sharp \in L \overset{m}{\longmapsto} L$ *such that* $\alpha \circ (\lambda X \bullet X \cup post[t]\, X) \circ \gamma \sqsubseteq \mathcal{F}^\sharp$*, pointwise:*

```
function f-ai2 ($I^\sharp$);                function f-mc2 ($I^\sharp$, $P$);
    $X := I^\sharp$;                              $X :=$ f-ai2($I^\sharp$);
    repeat                                  if $\gamma(X) \subseteq P$ then
        $Y := X$;                                 return true
        $X := \mathcal{F}^\sharp(Y)$;                      else
    until $X = Y$;                                return unknown.
    return $X$;
```

If P *is not computer representable or the test* $\gamma(X) \subseteq P$ *is not computable then a weaker computable form should be used instead, such as* $X \sqsubseteq P^\sharp$ *where* $\gamma(P^\sharp) \subseteq P$.

Since the successive values X^i, $i \geq 0$ *of the variable* X *form a* $\sqsubseteq$*-increasing chain, the test* $\gamma(X) \not\subseteq P$ *can be added in the loop of "f-ai2" to exit sooner from "f-mc2" with the "unknown" result. However the result "true" can only be returned upon termination.*

ALGORITHM **4** *Strictly speaking the formal verification algorithm "f-mc2" is not in the model-checking realm because no abstract transition system is explicitly used. However "f-ai2" can be refined into an* abstract model-checking algorithm *using the specific state abstraction of Example 4.1, by considering an abstract transition system* $\langle S^\sharp, t^\sharp, I^\sharp, F^\sharp \rangle$ *preserving universal properties:*

$$\alpha(I) \subseteq I^\sharp,$$
$$\forall s, s' \in S : \langle s, s' \rangle \in t \implies \langle h(s), h(s') \rangle \in t^\sharp .$$

This implies that:

$$\forall X \subseteq S : \alpha(X \cup post[t]\, X) \subseteq \alpha(X) \cup post[t^\sharp]\, \alpha(X),$$

whence

$$\alpha(\mathrm{lfp}^{\subseteq}_{I}\ \lambda X \bullet X \cup post[t]\, X) \ \subseteq\ \mathrm{lfp}^{\subseteq}_{I^\sharp}\ \lambda X \bullet X \cup post[t^\sharp]\, X .$$

It follows that

$$post[t^\star]\, I \ \subseteq\ \gamma(\mathrm{lfp}^{\subseteq}_{I^\sharp}\ \lambda X \bullet X \cup post[t^\sharp]\, X)$$

so that

$$\gamma(\mathrm{lfp}^{\subseteq}_{I^\sharp}\ \lambda X \bullet X \cup post[t^\sharp]\, X) \ \subseteq\ P$$

implies

$$post[t^\star]\, I \subseteq P\ .$$

This reduction by abstraction *leads to the following model-checking algorithm (Kelb, 1994, Dams et al., 1997) which considers exact properties of an approximate semantics, hence, in general, is incomplete (see however (Cleaveland et al., 1995)):*

```
function f-ai3 ($I^\sharp$);                function f-mc3 ($I^\sharp$, $P$);
  $X$ := $I^\sharp$;                           $X$ := f-ai3 ($I^\sharp$);
  repeat                                       if $\gamma(X) \subseteq P$ then
    $Y$ := $X$;                                   return true
    $X$ := $Y \cup post[t^\sharp]\, Y$;          else
  until $X = Y$;                                  return unknown.
  return $X$;
```

Observe that "f-mc3" amounts to "f-mc1" when the Galois connection $\langle\alpha,\ \gamma\rangle$ is an isomorphism. So the only difference between symbolic model-checking and abstract interpretation based model-checking is the possibility of information loss when necessary i.e. when concrete symbolic model-checking fails.

4.1.2. *Abstract Domains Not Satisfying the Ascending Chain Condition*

In general however, the iterates $\bar{\mathcal{F}}^\delta$, $\delta \geq 0$ of $\mathcal{F}^\sharp \in L \stackrel{\mathrm{m}}{\longmapsto} L$ in "f-ai2" do not converge to $\mathrm{lfp}^{\sqsubseteq}\, \mathcal{F}^\sharp$ in finitely many steps (as in e.g. (Cousot and Halbwachs, 1978), (Henzinger and Ho, 1995)). Hence, one must resort to a *widening operator* $\triangledown$ which can be used both to upper-approximate non-existent lubs (as in e.g. (Cousot and Halbwachs, 1978)) and to enforce finite convergence of increasing iterations (Cousot and Cousot, 1977, Cousot and Halbwachs, 1978). The widening operator $\triangledown \in L \times L \mapsto L$ is defined so as to be an upper bound:

$$\begin{array}{ll} \forall x, y \in L\ :\ x \sqsubseteq x \triangledown y, & \text{and} \\ \forall x, y \in L\ :\ y \sqsubseteq x \triangledown y, & \end{array}$$

and, if termination is required, to enforce finite convergence:

For all increasing chains $x^0 \sqsubseteq x^1 \sqsubseteq \ldots \sqsubseteq x^i \sqsubseteq \ldots$ the increasing chain defined by $y^0 = x^0, \ldots, y^{i+1} = y^i \triangledown x^{i+1}, \ldots$ is not strictly increasing.

Examples of widenings are given by (Halbwachs, 1993), (Halbwachs, 1994) for affine inequality relations and (Mauborgne, 1998) for BDDs.

The *upward iteration sequence with widening* for $\mathcal{F}^\sharp$ *from* $I^\sharp$ (with $I \subseteq \gamma(I^\sharp)$) is:

$$\begin{array}{rcll} \hat{\mathcal{F}}^0 & \stackrel{\Delta}{=} & I^\sharp, & \\ \hat{\mathcal{F}}^{i+1} & \stackrel{\Delta}{=} & \hat{\mathcal{F}}^i, & \text{if } \mathcal{F}^\sharp(\hat{\mathcal{F}}^i) \sqsubseteq \hat{\mathcal{F}}^i \\ \hat{\mathcal{F}}^{i+1} & \stackrel{\Delta}{=} & \hat{\mathcal{F}}^i \triangledown \mathcal{F}^\sharp(\hat{\mathcal{F}}^i), & \text{otherwise.} \end{array} \qquad (1)$$

By definition of the widening ∇, this upward iteration sequence is ultimately stationary and its limit $\hat{\mathcal{F}}$ is a sound upper approximation of $\mathrm{lfp}^{\sqsubseteq}\, \mathcal{F}^\sharp$ in that:

$$\mathrm{lfp}^{\sqsubseteq}\, \mathcal{F}^\sharp \ \sqsubseteq\ \hat{\mathcal{F}}\ .$$

If $\mathcal{F}^\sharp(\hat{\mathcal{F}}) \sqsubset \hat{\mathcal{F}}$ and the iterates:

$$\begin{aligned} \check{\mathcal{F}}^0 &\triangleq \hat{\mathcal{F}}, \\ \check{\mathcal{F}}^{\delta+1} &\triangleq \mathcal{F}^\sharp(\check{\mathcal{F}}^\delta), && \text{for successor ordinals} \\ \check{\mathcal{F}}^\lambda &\triangleq \bigsqcap_{\delta<\lambda} \check{\mathcal{F}}^\delta, && \text{for limit ordinals} \end{aligned}$$

do not finitely converge, we use a narrowing operator $\triangle$ to speed up the convergence. A *narrowing operator* $\triangle \in L \times L \mapsto L$ is defined such that:

$$\forall x, y \in L : x \sqsubseteq y \Longrightarrow x \sqsubseteq x \triangle y \sqsubseteq y$$

and, if termination is required, to enforce finite convergence:

For all decreasing chains $x^0 \sqsupseteq x^1 \sqsupseteq \ldots$ the decreasing chain defined by $y^0 = x^0, \ldots, y^{i+1} = y^i \triangle x^{i+1}, \ldots$ is not strictly decreasing.

So, if $\mathcal{F}^\sharp(X) = X \sqsubseteq \mathcal{F}(\hat{\mathcal{F}}) \sqsubseteq \hat{\mathcal{F}}$ then the *downward iteration sequence with narrowing* is defined by:

$$\begin{aligned} \check{\mathcal{F}}^0 &\triangleq \hat{\mathcal{F}}, \\ \check{\mathcal{F}}^{i+1} &\triangleq \check{\mathcal{F}}^i, && \text{if } \mathcal{F}^\sharp(\check{\mathcal{F}}^i) = \check{\mathcal{F}}^i, \\ \check{\mathcal{F}}^{i+1} &\triangleq \check{\mathcal{F}}^i \triangle \mathcal{F}^\sharp(\check{\mathcal{F}}^i), && \text{otherwise.} \end{aligned} \tag{2}$$

This downward iteration sequence is ultimately stationary and its limit $\check{\mathcal{F}}$ is a sound upper approximation of the fixpoint which is better than the one $\hat{\mathcal{F}}$ obtained by widening:

$$X \ \sqsubseteq\ \check{\mathcal{F}} \ \sqsubseteq\ \hat{\mathcal{F}}$$

In conclusion

$$\mathrm{lfp}^{\sqsubseteq}\, \mathcal{F}^\sharp \sqsubseteq \check{\mathcal{F}} \sqsubseteq \hat{\mathcal{F}}$$

so that by monotony

$$\mathrm{post}[t^\star]\, I \ =\ \mathrm{lfp}^{\subseteq}\, \lambda X \bullet I \cup \mathrm{post}[t]\, X \ \subseteq\ \gamma(\check{\mathcal{F}}) \ \subseteq\ \gamma(\hat{\mathcal{F}})\ .$$

It follows that we can choose the upper approximation D of $\mathrm{post}[t^\star]\, I$ to be:

$$D \ \triangleq\ \gamma(\check{\mathcal{F}})\ .$$

ALGORITHM **5** *If the set $S^\sharp$ of abstract states is infinite, $\langle \wp(S^\sharp), \subseteq\rangle$ does not satisfies the ascending chain condition, so that, in general, "f-ai3" may not converge, converge too slowly or require too much memory. In such cases, rapid convergence can be enforced using widening/narrowing operators, as follows:*

```
function f-ai4 ($I^\sharp$);
    $X$ := $I^\sharp$;
    loop
        $Y$ := $X$;
        $X$ := post[$t^\sharp$] $Y$;
        exit if $X \sqsubseteq Y$;
        $X$ := $Y \triangledown X$;
    repeat;
    while $X \neq Y$ do
        $Y$ := $X$;
        $X$ := $Y \vartriangle$ post[$t^\sharp$] $Y$;
    od;
    return $X$;
```

```
function f-mc4 ($I^\sharp$, $P$);
    $X$ := f-ai4 ($I^\sharp$);
    if $\gamma(X) \subseteq P$ then
        return true
    else
        return unknown.
```

Observe that "f-mc4" amounts to "f-mc3" when the abstract domain satisfies the ascending chain condition in which case $\triangledown = \sqcup$ and the narrowing phase (with $\vartriangle = \sqcap$) is not executed (since $X = Y$) hence is useless.

Because the least fixpoint is overshot, it is no longer possible to anticipate the "unknown" result within the loops of "f-ai2" as was the case for "f-ai2" and "f-ai3".

As already mentioned, the design of the abstract algebra $\langle L, \sqsubseteq, \bot, \top, \sqcup, \sqcap, \triangledown, \vartriangle, f_1, \dots, f_n\rangle$ and of the transformer $\mathcal{F}^\sharp$ (usually composed out of the primitives $f_1, \dots, f_n$) by structural induction on the syntax of programs are problem dependent and will not be further considered here.

4.2. *Backward Abstract Interpretation Based Model Checking*

Because pre[t] = post[t^{-1}], the situation is similar for computing an upper approximation A of the backward collecting semantics pre[$t^\star$] F = $\mathrm{lfp}^{\subseteq}\ \lambda X \bullet F \cup \mathrm{pre}[t]\, X$ using $\mathcal{B}^\sharp \in L \xrightarrow{m} L$ such that $\alpha \circ (\lambda X \bullet F \cup \mathrm{pre}[t]\, X) \circ \gamma \sqsubseteq \mathcal{B}^\sharp$, pointwise[5] (Cousot and Cousot, 1979).

ALGORITHM **6** *"b-mc1" can be abstracted using $\bar{P}^\sharp$ such that $\alpha(\neg P) \sqsubseteq \bar{P}^\sharp$ and an abstract backward predicate transformer $\mathcal{B}^\sharp \in L \xrightarrow{m} L$ such that $\alpha \circ (\lambda X \bullet X \cup \mathrm{pre}[t]\, X) \circ \gamma \sqsubseteq \mathcal{B}^\sharp$:*

```
function b-ai2 ($\bar{P}^\sharp$);
    $X$ := $\bar{P}^\sharp$;
    repeat
        $Y$ := $X$;
        $X$ := $\mathcal{B}^\sharp(Y)$;
    until $X = Y$;
    return $X$;
```

```
function b-mc2 ($I$, $\bar{P}^\sharp$);
    $X$ := b-ai2 ($\bar{P}^\sharp$);
    if $\gamma(X) \subseteq \neg I$ then
        return true
    else
        return unknown.
```

ALGORITHM 7 *The abstraction of Example 4.1 and the conditions:*

$$\alpha(\neg P) \subseteq \bar{P}^\sharp,$$
$$\forall s, s' \in S : \langle s,\ s'\rangle \in t \Longrightarrow \langle h(s),\ h(s')\rangle \in t^\sharp,$$

imply that:

$$\forall X \subseteq S :\ \alpha(X \cup pre[t]\, X)\ \subseteq\ \alpha(X) \cup pre[t^\sharp]\, \alpha(X),$$

which leads, by reduction by abstraction, to the following abstract version of the model-checking procedure "b-mc1":

```
function b-ai3 ($\bar{P}^\sharp$);
    $X := \bar{P}^\sharp$;
    repeat
        $Y := X$;
        $X := Y \cup pre[t^\sharp]\, Y$;
    until $X = Y$;
    return $X$;

function b-mc3 ($I$, $\bar{P}^\sharp$);
    $X :=$ b-ai3 ($\bar{P}^\sharp$);
    if $\gamma(X) \subseteq \neg I$ then
        return true
    else
        return unknown.
```

Observe again that "b-mc3" amounts to "b-mc1" when the Galois connection $\langle\alpha,\ \gamma\rangle$ is an isomorphism.

For abstract domains not satisfying the ascending chain condition, one first uses an upward iteration sequence with widening converging to $\hat{\mathcal{B}}$ followed by a downward iteration sequence with narrowing converging to $\check{\mathcal{B}}$ such that $\mathrm{lfp}^{\sqsubseteq}\, \mathcal{B}^\sharp \sqsubseteq \check{\mathcal{B}} \sqsubseteq \hat{\mathcal{B}}$ whence by monotony $\mathrm{pre}[t^\star]\, F = \mathrm{lfp}^{\subseteq}\, \lambda X \bullet F \cup \mathrm{pre}[t]\, X \subseteq \gamma(\check{\mathcal{B}}) \subseteq \gamma(\hat{\mathcal{B}})$. It follows that we can choose the upper approximation A of $\mathrm{pre}[t^\star]\, F$ to be $A \stackrel{\Delta}{=} \gamma(\check{\mathcal{B}})$.

ALGORITHM 8 *The widening/narrowing based version of the abstract model checking algorithm of "b-mc3" is (the assignment "Over := true;" and "**send**(...)" command should be ignored at the moment, i.e. **send**(...) returns a void value):*

```
function b-ai4 ($\bar{P}^\sharp$);
    $X := \bar{P}^\sharp$;
    loop
        $Y := X$;
        $X := pre[t^\sharp]\, Y$;
        exit if $X \sqsubseteq Y$;
        $X := Y \triangledown X$;
    repeat;
    Over := true;
    while $X \neq Y$ do
        send($\gamma(X)$);
        $Y := X$;
        $X := Y \vartriangle pre[t^\sharp]\, Y$;
    od;
    return $X$;

function b-mc4 ($I$, $\bar{P}^\sharp$);
    $X :=$ b-ai4 ($\bar{P}^\sharp$);
    if $\gamma(X) \subseteq \neg I$ then
        return true
    else
        return unknown.
```

Observe again that “b-mc4” amounts to “b-mc3” when the abstract domain satisfies the ascending chain condition in which case $\triangledown = \sqcup$ and the narrowing phase (with $\vartriangle = \sqcap$) is useless.

The approximation of $\widetilde{\mathrm{post}}[t^\star]\, I$ and $\widetilde{\mathrm{pre}}[t^\star]\, F$ can also be handled in the same way to yield dual abstract model-checking algorithms. .

4.3. *Combining Forward and Backward Abstract Interpretation Based Model Checking*

Observe that “f-mc1” and “b-mc1” are equivalent[6] whereas, because of the abstraction, “f-mc2” and “b-mc2”, “f-mc3” and “b-mc3” as well as “f-mc4” and “b-mc4” are not and may produce different answers. One algorithm might produce an affirmative answer whereas the other might be inconclusive. Therefore it is natural to combine the forward and backward information[7].

For that purpose, let us consider any sequence X^n, $n \geq 0$ defined simultaneously with A^n, $n \geq 0$ and D^n, $n \geq 0$ such that:

- $X^0 \supseteq I$,
- $\mathrm{post}[t^\star]\, X^n \subseteq D^n$,
  $\mathrm{pre}[t^\star](D^n - P) \subseteq A^n$,
  $X^{n+1} \supseteq X^n \cap A^n$,

Let us call a state “safe” if all its descendants by t satisfy P. The X^n, $n \geq 0$ are (smaller and smaller) sets of unsafe (ideally initial) states. D^n is an upper approximation of the descendants of X^n. A^n is an upper approximation of the ancestors of D^n not in P.
THEOREM **1** $\forall n \geq 0 : \mathit{post}[t^\star](I - X^n) \subseteq P$.

Proof: This proof is by recurrence on n.
For $X^0 \supseteq I$, we have $\mathrm{post}[t^\star](I - X^0) = \mathrm{post}[t^\star]\emptyset = \emptyset \subseteq P$.
Assume, by induction hypothesis, that $\mathrm{post}[t^\star](I - X^n) \subseteq P$. We have:

$$\begin{array}{lll}
& \mathrm{pre}[t^\star](D^n - P) \ \subseteq\ A^n & \\
\Longleftrightarrow & \neg A^n \ \subseteq\ \neg\, \mathrm{pre}[t^\star](D^n - P) & \text{by } X \subseteq Y \Longleftrightarrow \neg Y \subseteq \neg X, \\
\Longleftrightarrow & \neg A^n \ \subseteq\ \neg\, \mathrm{pre}[t^\star](D^n \cap (\neg P)) & \text{by def. } X - Y \overset{\vartriangle}{=} X \cap (\neg Y), \\
\Longleftrightarrow & \neg A^n \ \subseteq\ \widetilde{\mathrm{pre}}[t^\star]((\neg D^n) \cup P)) & \text{since } \neg(X \cap (\neg Y)) = \neg(X) \cup Y, \\
\Longleftrightarrow & \mathrm{post}[t^\star](\neg A^n) \ \subseteq\ ((\neg D^n) \cup P)) & \text{since } \langle \mathrm{post}[t^\star],\ \widetilde{\mathrm{pre}}[t^\star]\rangle \text{ is a Galois} \\
& & \text{connection.}
\end{array}$$

We then have:

$$\begin{array}{lll}
& \mathrm{post}[t^\star](I - X^{n+1}) & \\
\subseteq & \mathrm{post}[t^\star](I - (X^n \cap A^n)) & \text{since post}[t^\star] \text{ is monotonic,} \\
= & \mathrm{post}[t^\star]((I - X^n) \cup (I \cap (X^n - A^n))) & \\
= & \mathrm{post}[t^\star](I - X^n) \cup \mathrm{post}[t^\star](I \cap (X^n - A^n)) & \text{since post}[t^\star] \text{ preserves lubs,} \\
\subseteq & P \cup \mathrm{post}[t^\star](I \cap (X^n - A^n)) & \text{by induction hypothesis,} \\
\subseteq & P \cup \mathrm{post}[t^\star](X^n - A^n) & \text{since post}[t^\star] \text{ is monotonic,}
\end{array}$$

$$\begin{array}{lll}
= & P \cup \mathrm{post}[t^\star](X^n \cap (\neg A^n)) & \text{by def. } X - Y \overset{\triangle}{=} X \cap (\neg Y), \\
\subseteq & P \cup (\mathrm{post}[t^\star]\, X^n \cap \mathrm{post}[t^\star](\neg A^n)) & \text{since post}[t^\star] \text{ is monotonic,} \\
\subseteq & P \cup (D^n \cap \mathrm{post}[t^\star](\neg A^n)) & \text{by hypothesis,} \\
\subseteq & P \cup (D^n \cap ((\neg D^n) \cup P)) & \text{by the above lemma,} \\
= & P \cup (D^n \cap P) & \\
= & P & \text{since } (D^n \cap P) \subseteq P.
\end{array}$$

■

COROLLARY **1** *If* $\exists n \geq 0 : X^n \cap I = \emptyset$ *then* $\mathrm{post}[t^\star]\, I \subseteq P$.

Proof: If $X^n \cap I = \emptyset$ then $I - X^n = I$ so $\mathrm{post}[t^\star](I) \subseteq P$ by Theorem 1. ■

ALGORITHM **9** *Assuming that* $I^\sharp$, $\bar{P}^\sharp$ *and* $\bot$ *are respective abstractions of* I, $\neg P$ *and* $\emptyset$, *more precisely:*

$$\begin{array}{rrcl}
 & \alpha(I) & \subseteq & I^\sharp, \\
 & \gamma(\bar{P}^\sharp) & \supseteq & \neg P, \\
\text{and} & \gamma(\bot) & = & \emptyset,
\end{array} \tag{3}$$

this leads to the following abstract model-checking algorithm (where "f-ai5" and "b-ai5" are any one of the forward and backward abstract interpretation algorithms considered previously, such as "f-ai4" and "b-ai4"):

```
function mc5 ($I^\sharp$, $\bar{P}^\sharp$);
  $Y := I^\sharp$;
  repeat
    $X := Y$;
    $D :=$ f-ai5($X$);
    $A :=$ b-ai5($D \sqcap \bar{P}^\sharp$);
    $Y := X \vartriangle A$;
  until $X = Y$;
  if $I^\sharp \sqcap X = \bot$ then
    return true
  else
    return unknown.
```

THEOREM **2 (Correctness of the abstract model-checking algorithm "mc5")**
If (3) holds and "mc5 ($I^\sharp$, $\bar{P}^\sharp$)" returns "true" then $\mathrm{post}[t^\star]\, I \subseteq P$.

Proof: We have $I \subseteq \gamma(I^\sharp)$ so $X^0 = \gamma(I^\sharp) \supseteq I$.

At the n-th iteration of the loop, we let $X^n = \gamma(X)$ so that $D^n = \gamma(D) \supseteq \mathrm{post}[t^\star]\, X^n$ by correctness of the abstract interpretation function "f-ai5". γ preserves glbs, hence we have $\gamma(D \sqcap \bar{P}^\sharp) = \gamma(D) \cap \gamma(\bar{P}^\sharp) \supseteq D^n \cap (\neg P)$ and $\mathrm{pre}[t^\star](\gamma(D \sqcap \bar{P}^\sharp)) \supseteq \gamma(A)$ by correctness of the abstract interpretation function "b-ai5" whence $A^n = \gamma(A) \supseteq \mathrm{pre}[t^\star](D^n \cap (\neg P)$ by monotony. Moreover $X^{n+1} = \gamma(Y) = \gamma(X \vartriangle A) \supseteq \gamma(X) \cap \gamma(A)$.

By definition of the narrowing operation $\vartriangle$, the sequence of values of the variable X cannot be strictly $\sqsubseteq$-decreasing so that the loop must terminate.

Upon termination, after n iterations, we have:

$$\begin{array}{ll l}
 & I^\sharp \sqcap X^n = \bot & \\
\Longrightarrow & \gamma(I^\sharp \sqcap X^n) = \gamma(\bot) & \\
\Longrightarrow & \gamma(I^\sharp) \cap \gamma(X^n) = \gamma(\bot) & \text{since } \gamma \text{ preserves lubs,} \\
\Longrightarrow & \gamma(I^\sharp) \cap \gamma(X^n) = \emptyset & \text{by hypothesis } \gamma(\bot) = \emptyset, \\
\Longrightarrow & I \cap \gamma(X^n) = \emptyset & \text{by hypothesis } \alpha(I) \subseteq I^\sharp \Longleftrightarrow I \subseteq \gamma(I^\sharp),
\end{array}$$

proving, by Theorem 1, the correctness post$[t^\star]\,I \subseteq P$ of the abstract model-checking algorithm "mc5". ∎

A possible improvement of "mc5" would be to initialize D^0 to the supremum $\top$ and to run in parallel the computations of

$$\begin{array}{rrcl}
 & D^{n+1} & := & \underline{\text{f-ai5}}(X^n) \\
\text{and} & A^{n+1} & := & \underline{\text{b-ai5}}(D^n \sqcap \bar{P}^\sharp) \,.
\end{array}$$

If $\gamma(D^1) \subseteq P$ then we can conclude positively (as would have done the forward algorithm "f-mc4"). If $\gamma(A^1) \subseteq \neg I$ then we can also conclude positively (as would have done the backward algorithm "b-mc4"). Otherwise "mc5" cannot conclude on the first iteration (but in this case neither "f-mc4" nor "b-mc4" would be conclusive). If further iterations are needed to conclude then the result is obtained highly automatically whereas a manual adjustment of the abstraction would be needed for "f-mc4" or "b-mc4" to get the same result. Experience with abstract interpretation shows that a few iterates only are needed for convergence (typically 3, see e.g. (Bourdoncle, 1993)).

5. Combining Abstract Interpretation and Model Checking when Abstract Model Checking is Inadequate

We now consider the verification of infinite state concurrent systems for which neither the exact symbolic nor the abstract model-checking algorithms are applicable.

5.1. Verification Problems For Which Abstract Model Checking is Inadequate

5.1.1. Minimum Delay Problem The *minimum delay problem* (see e.g. (Halbwachs, 1993)) for a transition system $\langle S,\ t,\ I,\ F\rangle$ consists in computing the length ℓ of (i.e. number of edges in) a shortest path from an initial state in I to a final state in F.

$$\ell \ \stackrel{\triangle}{=}\ \min\{n \mid \exists s \in I, s' \in F : \langle s,\ s'\rangle \in t^n\},$$

where the minimum $\min \emptyset$ of the empty set $\emptyset$ is chosen to be infinity ∞:

$$\min \emptyset \ \stackrel{\triangle}{=}\ \infty \,.$$

An example of transition system and corresponding minimum delays is given in Figure 3.

The following model-checking *minimum delay algorithm* is due to (Campos et al., 1995):

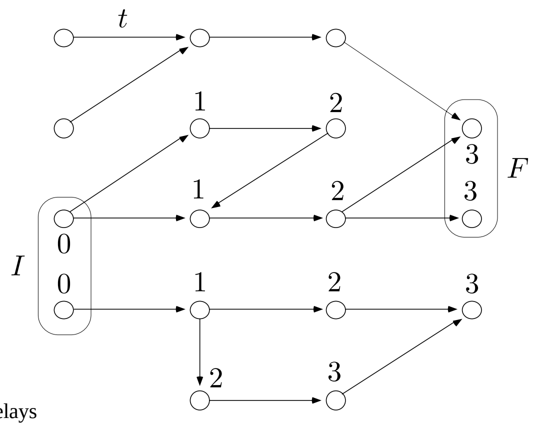


Figure 3. Minimum delays

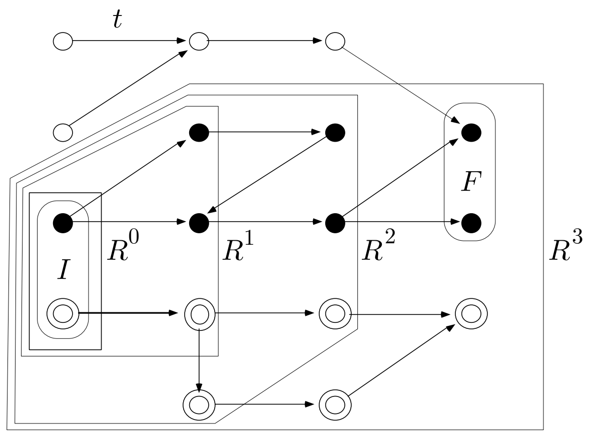


Figure 4. Execution trace of algorithm "minimum1"

```
function minimum1 ($I$, $F$);
$R$ := $I$;
$n$ := 0;
stable := ($R \cap F \neq \emptyset$);
while ¬stable do
    $R'$ := $R \cup \text{post}[t]\, R$;
    $n$ := $n + 1$;
    stable := ($R = R'$) $\vee$ ($R' \cap F \neq \emptyset$);
    $R$ := $R'$;
od;
return if ($R \cap F \neq \emptyset$) then $n$ else $\infty$.
```

An example of execution trace of the "minimum1" algorithm is given in Figure 4. In order to consider infinite state sets, it is necessary to enforce finite convergence. Abstract model-checking techniques, with abstractions of transitions, are not applicable since they would either lead to erroneous results (unless a lower or upper bound of the minimum delay is acceptable) or state space reduction would be precluded[8]. Classical symbolic methods for speeding up model-checking algorithms such as BDDs to encode boolean formulas representing sets of states, the transition relation, and so on or "on-the-fly" property checking, without state graph generation are applicable in this case. However, there is a serious potential inefficiency problem because of useless exploration of dead-end states which are reachable but cannot lead to a final state. These dead-end states are marked ◎ in Figure 4.

However, we can still use abstract interpretation to cut down the size of the model-checking search space by determining, as shown in Section 4.1, a super-set A of the ancestors of the final states:

$$\text{pre}[t^\star]\, F \;\subseteq\; A\,.$$

The states in the set $\text{pre}[t^\star]\, F$ of ancestors of the final states F are marked ● in Figure 5. This information can then be used to restrict the exploration of the transition graph for computing the minimum delay. The *revisited minimum delay algorithm* is now (the **receive**(...) commands should be ignored at the moment, i.e. **receive**(...) returns a void value):

```
function minimum2 ($I$, $F$);
$R := I$;
$n := 0$;
stable := $(R \cap F \neq \emptyset)$;
while $\neg$stable do
    receive($A$);
    $R' := R \cup \boxed{(\text{post}[t]\, R \cap A)}$;
    $n := n + 1$;
    stable := $(R = R') \vee (R' \cap F \neq \emptyset)$;
    $R := R'$;
od;
return if $(R \cap F \neq \emptyset)$ then $n$ else $\infty$.
```

A trace of this algorithm "minimum2" is given in Figure 5.

Observe that:

– any upper-approximate solution $\text{pre}[t^\star]\, F \subseteq A$ can be used in algorithm "minimum2";
– the upper approximation A of $\text{pre}[t^\star]\, F$ which is used in the loop can be different at each iteration; and
– in the worst possible case, when the analysis by abstract interpretation is totally unfruitful, we have $A = S$ in which case algorithm "minimum2" simply amounts to algorithm "minimum1".

5.1.2. Maximum Delay Problem The *maximum delay problem* consists in computing the length m of (i.e. number of edges in) a longest path from an initial state in I to a final

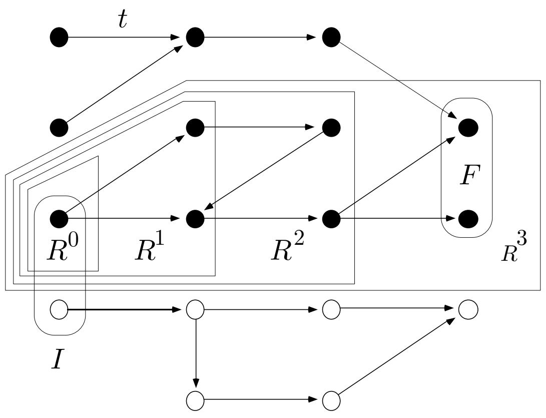


Figure 5. Execution trace of algorithm "minimum2"

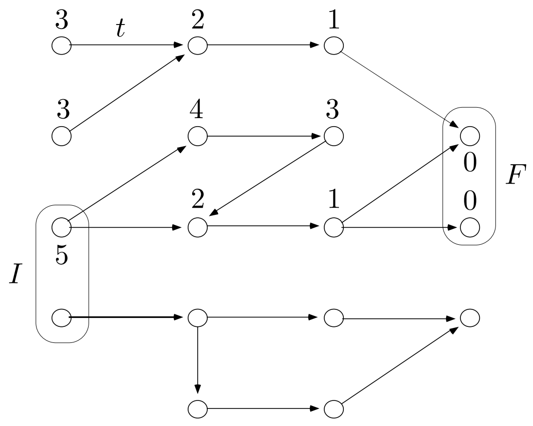


Figure 6. Maximum delays

state in F:

$$m \triangleq \max\{n \mid \exists s \in I, s' \in F : \langle s,\ s'\rangle \in (\neg F \rceil t)^n\},$$

where the left-restriction $\neg F \rceil t$ of a relation t to $\neg F \subseteq S$ has been defined as $\{\langle s,\ s'\rangle \in t \mid s \notin F\}$ and the maximum $\max \mathbb{N}$ of the set $\mathbb{N}$ of natural numbers is chosen to be infinity ∞:

$$\max \mathbb{N} \triangleq \infty\ .$$

An example of maximum delays is given in Figure 6. The following *maximum delay algorithm* has been proposed by (Campos et al., 1995):

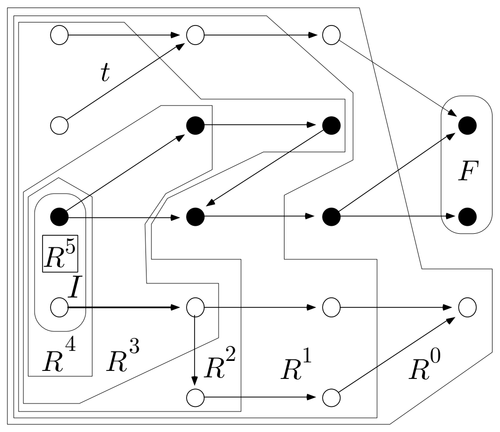


Figure 7. Execution trace of the "maximum1" algorithm

```
**function** maximum1 ($I$, $F$);
$R' := S$;
$n := 0$;
$R := (S - F)$;
**while** ($R \neq R' \wedge R \cap I \neq \emptyset$) **do**
    $R' := R$;
    $n := n + 1$;
    $R := \mathrm{pre}[t]\ R' \cap (S - F)$;
**od**;
**return if** ($R' = R$) **then** $\infty$ **else** $n$.
```

An example of an execution trace of the "maximum1" algorithm is given in Figure 7.

Although this is left unspecified by (Campos et al., 1995), the correctness of this "maximum1" delay algorithm relies on several hypotheses. First the sets of initial states I and final states F must be nonempty and disjoint. Second, there exists at least one path from some initial state to some final state. Third, there is no path starting from an initial state, ending in a blocking state (with no successor by the transition relation) never passing through a final state. Fourth and finally, there is no infinite or endless cyclic path starting from an initial state and never passing through a final state. If one of these hypotheses is not satisfied, the algorithm maximum1 returns an upper bound of the maximal path length.

Once again abstraction of the transition system would also provide an upper bound of the maximal path length hence would be incorrect. Exact symbolic methods have a potentially serious inefficiency problem because of useless exploration of dead-end states (marked ○ in Figure 7) which are not reachable from initial states or cannot lead to a final state. Observe that partial-order methods (Valmari, 1993), which are based on the fact that in concurrent systems, the total effect of a set of actions is often independent of the order in which the actions are taken, would locally reduce the number of considered paths, but would not perform a global elimination of the remaining paths that are useless for the verification.

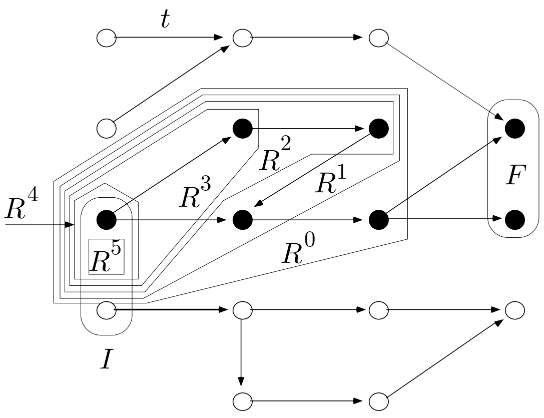


Figure 8. Execution trace of the "maximum2" algorithm

Once again an automatic analysis by abstract interpretation can determine a super-set U of the descendants of the initial states I which are ancestors of the final states F (the principle of determination of U by abstract interpretation will be precisely defined in Section 5.2):

$$\begin{aligned} U &\supseteq \text{post}[t^\star]\, I \cap \text{pre}[t^\star]\, F, \\ &= \{s \mid \exists s' \in I, s'' \in F : \langle s',\ s\rangle \in t^\star \wedge \langle s,\ s''\rangle \in t^\star\}\ . \end{aligned}$$

The states in the set of descendants of the initial states I which are ancestors of the final states F are marked ● in Figures 7 and 8. This leads to a *revisited maximum delay algorithm*, as follows (the **receive**(...) commands should be ignored at the moment, i.e. **receive**(...) returns a void value):

```
function maximum2 $(I, F)$;
$R' := S$;
$n := 0$;
receive$(U)$;
$R := \boxed{(U - F)}$;
 while $(R \neq R' \wedge R \cap I \neq \emptyset)$ do
      $R' := R$;
      $n := n + 1$;
      receive$(U)$;
      $R := \text{pre}[t]\, R' \cap \boxed{(U - F)}$;
od;
return if $(R' = R)$ then $\infty$ else $n$.
```

An example of execution trace of the "maximum2" algorithm is given in Figure 8.

Observe that any upper-approximation $\text{post}[t^\star]\, I \cap \text{pre}[t^\star]\, F \subseteq U$ of the descendants of the initial states I which are ancestors of the final states F is correct, since in the worst possible case, when $U = S$, algorithm "maximum2" simply amounts to "maximum1". Moreover, a different upper approximation U of $\text{post}[t^\star]\, I \cap \text{pre}[t^\star]\, F$ can be used at each iteration in the loop. Notice also that this restriction idea applies both to exhaustive and on-the-fly state space exploration techniques.

In the case of symbolic model-checking, say with BDDs (or polyhedron or set of polyhedra), the intersection $\text{pre}[t]\, R' \cap (U - F)$ may be a BDD (or polyhedron or set of polyhedra) of much greater size than $\text{pre}[t]\, R'$, although it describes a smaller set of states. In this case, the computation of the intersection is not mandatory, the information being still useful for simplifying the BDD (or polyhedra), e.g. by pruning, in order to reduce its size. Several such operators have been suggested such as the *cofactor* (Touati et al., 1990), *constrain* (Coudert, Berthet, and Madre, 1990) or *restrict* (Coudert, Madre, and Berthet, 1990) operators on BDDs and the *polyhedron simplification* of (Halbwachs and Raymond, 1996).

5.2. Combining Forward and Backward Analysis by Abstract Interpretation

We are left with the problem of computing an upper-approximation of $\text{post}[t^\star]\, I \cap \text{pre}[t^\star]\, F$. We use an upper-abstraction $\mathcal{F}^\sharp = \lambda X \bullet I^\sharp \sqcup \text{post}[t^\sharp]\, X$ of $\lambda X \bullet I \cup \text{post}[t]\, X$ so as to compute an abstraction $\check{\mathcal{F}} = \underline{\text{f-ai4}}\ (I^\sharp)$ of $\text{post}[t^\star]\, I$, as defined in Section 4.1.2. The same way, using an upper-abstraction $\mathcal{B}^\sharp = \lambda Y \bullet \text{pre}[t^\sharp]\, Y \sqcup F^\sharp$ of $\lambda Y \bullet \text{pre}[t]\, Y \cup F$, we can compute an abstraction $\check{\mathcal{B}} = \underline{\text{b-ai4}}\ (F^\sharp)$ of $\text{pre}[t^\star]\, F$, as defined in Section 4.2. We then have the trivial solution $\text{post}[t^\star]\, I \cap \text{pre}[t^\star]\, F \subseteq \gamma(\check{\mathcal{F}} \sqcap \check{\mathcal{B}})$.

This can be improved using the following approximation sequence which limit is always more precise than or equal to $\check{\mathcal{F}} \sqcap \check{\mathcal{B}}$ (Cousot, 1978, Cousot and Cousot, 1992a):

– $\dot{U}^0$ is the limit of the upward iteration sequence with widening (1) for the function $\mathcal{F}^\sharp$ and $\ddot{U}^0$ is the limit of the corresponding downward iteration sequence with narrowing[9] (2);
– . . .
– $\dot{U}^{2n+1}$ is the limit of the upper upward iteration sequence with widening for $\lambda X \bullet (\ddot{U}^{2n} \sqcap \mathcal{F}^\sharp(X))$ and $\ddot{U}^{2n+1}$ is the limit of the corresponding downward iteration sequence with narrowing;
– $\dot{U}^{2n+2}$ is the limit of the upward iteration sequence with widening for the function $\lambda X \bullet (\ddot{U}^{2n+1} \sqcap \mathcal{B}^\sharp(X))$ and $\ddot{U}^{2n+2}$ is the limit of the corresponding downward iteration sequence with narrowing.
– . . .

Observe that the sequence $\dot{U}^0$, $\ddot{U}^0$, $\dot{U}^1$, $\ddot{U}^1$, . . . , $\dot{U}^{2n}$, $\ddot{U}^{2n}$, $\dot{U}^{2n+1}$, $\ddot{U}^{2n+1}$, . . . is a descending chain and the concretization of any element in this sequence is a $\subseteq$-upper approximation of $\text{post}[t^\star]\, I \cap \text{pre}[t^\star]\, F$.

Observe that if no abstraction is performed (that is $\mathcal{F}^\sharp = \lambda X \bullet I \cup \text{post}[t]\, X$, $\mathcal{B}^\sharp = \lambda Y \bullet \text{pre}[t]\, Y \cup F$, $\nabla = \cup$, $\triangle = \cap$) and the set of states is finite then $\dot{U}^0 = \ddot{U}^0 = \text{post}[t^\star]\, I$ and $\dot{U}^1 = \ddot{U}^1 = \text{post}[t^\star]\, I \cap \text{pre}[t^\star]\, F$. So in this particular case, convergence to the exact solution is immediate. However in general the abstraction introduces a loss of information and the additional computational work of computing the limit of the sequence $\dot{U}^i$, $\ddot{U}^i$, $i \geq 0$ is justified by the improved precision.

EXAMPLE: The same phenomenon appears in abstract model-checking algorithm “mc5”, where the computation of $A := \underline{\text{b-ai5}}(D \sqcap \bar{P}^\sharp)$ will be more precise when intersecting with D. The same way, the computation of $D := \underline{\text{f-ai5}}(X)$ can use an abstract intersection $\sqcap$ with A, initially $\top = \alpha(S)$. ☐

ALGORITHM **10** *The concretization of the result of "ai6" below is a $\sqsubseteq$-upper approximation of post$[t^\star]$ $I \cap$ pre$[t^\star]$ F. We use a global counter $n \geq 0$ so as to indicate by comments $\{\dot{U}^{2n} = \ldots\}$, $\{\ddot{U}^{2n} = \ldots\}$, $\{\dot{U}^{2n+1} = \ldots\}$ and $\{\ddot{U}^{2n+1} = \ldots\}$ how and when the elements of the sequence $\dot{U}^0$, $\ddot{U}^0$, $\dot{U}^1$, $\ddot{U}^1$, $\ldots$, $\dot{U}^{2n}$, $\ddot{U}^{2n}$, $\dot{U}^{2n+1}$, $\ddot{U}^{2n+1}$, $\ldots$ have been computed. The corresponding* ***send***(. . .) *commands should be ignored at the moment, i.e.* ***send***(. . .) *returns a void value.*

```
function f-ai6 (A);
  X := ⊥;
  loop
    Y := X;
    X := $\mathcal{F}^\sharp(Y) \sqcap A$;
    exit if $X \sqsubseteq Y$
    X := $(Y \triangledown X) \sqcap A$;
  forever;
  {$\dot{U}^{2n} = Y$}
  send($\gamma(Y)$);
  while $X \neq Y$ do
    Y := X;
    X := $\mathcal{F}^\sharp(Y) \sqcap A$;
    X := $(Y \vartriangle X) \sqcap A$;
  od;
  return X.
```

```
function b-ai6 (D);
  X := ⊥;
  loop
    Y := X;
    X := $\mathcal{B}^\sharp(Y) \sqcap D$;
    exit if $X \sqsubseteq Y$
    X := $(Y \triangledown X) \sqcap D$;
  forever;
  {$\dot{U}^{2n+1} = Y$}
  send($\gamma(Y)$) %
  while $X \neq Y$ do
    Y := X;
    X := $\mathcal{B}^\sharp(Y) \sqcap D$;
    X := $(Y \vartriangle X) \sqcap D$;
  od;
  return X.
```

```
function ai6 ();
  D := ⊤;
  A := ⊤;
  n := 0;
  repeat
    D := f-ai6 (A);
    {$\ddot{U}^{2n} = D$}
    send($\gamma(D)$);
    A := b-ai6 (D);
    {$\ddot{U}^{2n+1} = A$}
    send($\gamma(A)$);
    n := $n + 1$;
  until A = D;
  return $\gamma(A)$.
```

Observe that if the lattice L does not satisfy the descending chain condition (DCC)[10]*, this approximation sequence may be infinite. We can enforce finite convergence using a narrowing as suggested in (Cousot, 1978, Cousot and Cousot, 1992a) ("A := b-ai6 (D);" being replaced by "A := D $\vartriangle$ b-ai6 (D);").*

We are now in position to explain how the verification by model-checking can interact with the analysis of the system by abstract interpretation. The general idea is to improve the efficiency of symbolic model-checking algorithms for verifying systems by using properties of the system that can be automatically inferred by abstract interpretation.

5.3. *Sequential Combination of Abstract Interpretation and Model Checking*

A simple interaction of an analysis of the system by abstract interpretation with a verification by model-checking algorithm "maximum2" consists in first running "ai6" so as to get an upper-approximation "U = ai6 ()" of post$[t^\star]$ $I \cap$ pre$[t^\star]$ F which is then used in "maximum2". The benefit is that the state space to be searched can be reduced to a finite set (else "maximum2" fails anyway). The inconvenience is the additional cost of the preliminary analysis.

5.4. *Parallel Combination of Abstract Interpretation and Model Checking*

5.4.1. *Maximum Delay Problem*

This untimeliness can be remedied by running the abstract interpretation and model-checking algorithms in parallel. Intermediate abstract interpretation results can be used, as they become available, to reduce the size of the state space to be explored during parallel model-checking. The parallel algorithm is then:

ALGORITHM **11**

$$\boxed{[\![\ \underline{ai6}\,()\ \parallel\ \underline{maximum2}\,(I, F)\ ]\!]}$$

*The semantics of the "**send**(V);" and "**receive**(U);" commands is that of an asynchronous one-place buffered communication where the buffer is initialized to the supremum $\top = \alpha(S)$, "**send**(V);" replaces the current value of the buffer with V while "**receive**(U);" assigns to U the current value of the buffer which is left unchanged.*

So execution of algorithm "maximum2" is started in parallel with the computation of the upper approximation sequence $\top$, $\dot{U}^0$, $\ddot{U}^0$, ..., $\dot{U}^{2n}$, $\ddot{U}^{2n}$, $\dot{U}^{2n+1}$, $\ddot{U}^{2n+1}$, ... At each iteration of the main loop in "maximum2"*, one can chose U as the element in this sequence which is currently available*[11]*. A double-buffering system can be considered to minimize the mutually exclusive accesses to read and write the shared buffer. Then there is no synchronization cost since the parallel computation is completely asynchronous. This computation should be stopped as soon as the execution of* "maximum2" *terminates.*

Finally, it should be observed that initially the model-checking algorithm manipulates small sets while the information $\top \sqsupseteq \dot{U}^0 \sqsupseteq \ddot{U}^0 \sqsupseteq \ldots$ provided by abstract interpretation is rough. While the parallel computations go on, the model-checking algorithm manipulates larger and larger sets while the information U provided by abstract interpretation $\ldots \sqsupseteq \dot{U}^{2n} \sqsupseteq \ddot{U}^{2n} \sqsupseteq \dot{U}^{2n+1} \sqsupseteq \ddot{U}^{2n+1} \sqsupseteq \ldots$ is more and more precise, so that the restriction is more efficient. It follows that the parallel strategy is adequate since the precise information will be available when most strongly needed.

5.4.2. Minimum Delay Problem In the case of algorithm "minimum2", the first iterates $\hat{\mathcal{B}}^0 = \emptyset$, $\hat{\mathcal{B}}^1$, ... of the upward iteration sequence with widening for $\mathcal{B}^\sharp = \lambda X \bullet F^\sharp \sqcup \text{pre}[t^\sharp]\,X$ are not upper approximations of $\text{pre}[t^\star]\,F$. It follows that one has to choose $A = S$ while waiting for their limit $\hat{\mathcal{B}}$ to be computed. Once available, one can use the iterates $\check{\mathcal{B}}^0 = \hat{\mathcal{B}}, \check{\mathcal{B}}^1, \ldots$ of the corresponding downward iteration sequence with narrowing as successive values of A in "minimum2" (where the assignment "Over := true;" should be ignored in "b-ai4"):

ALGORITHM **12**

$$\boxed{[\![\ \underline{b\text{-}ai4}\,(\alpha(F))\ \parallel\ \underline{minimum2}\,(I, F)\ ]\!]}$$

However, while waiting for $\hat{\mathcal{B}}$ to be available, the successive values of A can be chosen as the downward iterates for the greatest fixpoint $\text{gfp}^{\sqsubseteq}\,\mathcal{B}^\sharp$ since they are all upper approximations of $\text{lfp}^{\sqsubseteq}\,\mathcal{B}^\sharp$ and more precise than S:

ALGORITHM **13**

```
$Over := false;$
$[\![$ b-ai7 $(\alpha(F))$ $\parallel$ b-ai4 $(\alpha(F))$ $\parallel$ minimum2 $(I, F)$ $]\!]$
```

where termination is driven by that of "minimum2" and "b-ai7" is defined as:

```
function b-ai7 $(F^\sharp)$;
  $X := \top;$
  while $\neg\ Over$ do
    $X := F^\sharp \sqcup pre[t^\sharp]\, X;$
    send$(X)$;
  od;
```

Finally observe that weak fairness ensures that the abstract interpretation and model checking cooperate effectively. However in an unfair computation where "b-ai7" and "b-ai4" are blocked, the buffer will always contains the supremum $\top$ so that the computation of "minimum2" will simply amount to that of "minimum1".

6. Conclusion

We have proposed refinements of universal safety model-checking by abstract interpretation with infinite approximation of transition systems and sets of states:

– The combination of forwards and backwards abstract fixed-point model-checking computations for universal safety computes a more precise result than that computed by conjunction of the forward and backward analyses alone, without needing to refine the abstraction;

– When abstraction is unsound (as can happen in minimum/maximum path-length problems), the partial results of a classical combination of forward and backward abstract interpretation analyses for universal safety can be used to reduce, on-the-fly, the concrete state space to be searched by model-checking.

Other forms of restrictions have been proposed by (Halbwachs and Raymond, 1996) which are amenable to parallelization in a similar way. Such methods, which make no approximation on the states and transitions of the model, are nevertheless partial since it is not guaranteed that the reduction always leads to a finite state exploration sub-graph. Because of its precision, it should be tried first or in parallel. In case of computational verification costs which remain prohibitive despite the restriction, one can always later resort to the more classical property and transition abstraction.

Remarkably enough, the method then remains applicable to the more abstract model of properties and/or transitions. Indeed, by (Cousot and Cousot, 1992c), the abstract interpretation of the refined model will always be more precise than the analysis of the abstract model. Consequently the preliminary analysis has not been done for nothing. It follows that the idea can *always* be applied, and thanks to an abstract interpretation performed in parallel with the model-checking verification, should have a marginal cost only.

Similar restriction ideas apply to bisimulation equivalence checking (see e.g. (Bouajjani et al., 1992, Fernandez, 1993)). They seem indispensable to cope with infinite state systems, real-time systems (Halbwachs, 1994) and hybrid systems (Halbwachs et al., 1994), in particular to take possible values of variables, messages, queues, and the like into account, which would be a significant step in the automated analysis of software.

Acknowledgments

We thank the anonymous referees for their helpful suggestions on the presentation.

Notes

1. L satisfies the ACC if and only if any strictly ascending chain $x_0 \sqsubset x_1 \sqsubset \cdots$ of elements of L is necessarily finite.
2. Equivalence means that if the two procedures terminate then they return exactly the same result.
3. Strictly speaking this is a *semi-dual Galois connection* since, as observed in (Cousot and Cousot, 1979), the original definition corresponds to $\langle \wp(S),\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L,\ \sqsupseteq\rangle$.
4. Weaker models of abstract interpretation can be considered (Cousot and Cousot, 1992b), which are mandatory when considering abstract properties with no best approximation (e.g. (Cousot and Halbwachs, 1978)).
5. More generally one could consider a different abstract domain for backward analysis, the generalization being immediate.
6. Again, equivalence means that if the two procedures terminate then they return exactly the same result.
7. Notice that the proposed combination is inspired from the technique of (Cousot, 1978, Cousot and Cousot, 1992a) to upper approximate $\text{post}[t^\star]\, I \cap \text{pre}[t^\star]\, F$ as briefly recalled in Section 5.2 but is different, as shown by the proof of Theorem 1. Another forward-backward combination was also proposed by (Dill and Wong-Toi, 1995), which consists in computing separate upper and lower approximations of both $\text{post}[t^\star]\, I$ and $\widetilde{\text{pre}}[t^\star]\, P$
8. For example a referee suggested that "One must simply make the restriction that the abstraction function not collapse two states that are separated by a non-cyclic edge". For an acyclic graph with "non-cyclic edges" only (as in Figure 3), no abstraction is possible with this restriction.
9. Depending on the problem under consideration, it might be semantically equivalent but more efficient to start with $\mathcal{B}^\sharp$ instead of $\mathcal{F}^\sharp$.
10. L satisfies the DCC if and only if any strictly descending chain $x_0 \sqsupset x_1 \sqsupset \cdots$ of elements of L is necessarily finite.
11. Observe that all iterates of the downward iteration with narrowing to compute $\ddot{U}^k$ from $\dot{U}^k$ could also have been included in this sequence.

References

Akers, S. 1978. Binary decision diagrams. *IEEE Trans. Comput. C-27*, 6, 509–516.

Bouajjani, A., Fernandez, J.-C., Halbwachs, N., Raymond, P., and Ratel, C. 1992. Minimal state graph generation. *Sci. Comput. Prog. 18*, 247–269.

Bourdoncle, F. 1993. Abstract Debugging of Higher-Order Imperative Languages. In *Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation*, ACM Press, 46–55.

Bryant, R. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput. C-35*, 8, 677–691.

Burch, J., Clarke, E., McMillan, K., Dill, D., and Hwang, L. 1992. Symbolic model-checking: 10^{20} states and beyond. *Inf. & Comp. 98*, 2, 142–170.

Campos, S., Clarke, E., Marrero, W., and Minea, M. 1995. Verus: A tool for quantitative analysis of finite-state real-time systems. In *Proc. ACM SIGPLAN 1995 Workshop on Languages, Compilers & Tools for Real-Time Systems*, 75–83.

Clarke, E., and Emerson, E. 1981. Synthesis of synchronization skeletons for branching time temporal logic. In *Logics of Programs: Workshop*, LNCS 131, Springer-Verlag.

Clarke, E., Emerson, E., and Sistla, A. 1983. Automatic verification of finite-state concurrent systems using temporal logic specifications. In *10th POPL* (1983) and *Trans. Prog. Lang. Sys.*, 8:244–263 (1986). ACM Press.

Clarke, E., Grumberg, O., and Long, D. 1992. Model checking abstraction. In *19th POPL*, 343–354. ACM Press.

Cleaveland, R., Iyer, P., and Yankelevitch, D. 1995. Optimality in abstractions of model-checking. In A. Mycroft Ed., *Proc. SAS '95*, LNCS 983, 51–63. Springer-Verlag.

Coudert, O., Berthet, C., and Madre, J. 1990. Verification of synchronous sequential machines based on symbolic execution. In J. Sifakis Ed., *Proc. Int. Work. on Automatic Verification Methods for Finite State Systems*, LNCS 407, 365–373. Springer-Verlag.

Coudert, O., Madre, J., and Berthet, C. 1990. Verifying temporal properties of sequential machines without building their state diagrams. In E. Clarke and R. Kurshan Eds., *Computer Aided Verification '90*, Number 3 in DIMACS Volume Series, 75–84. AMS.

Cousot, P. 1978. Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique de programmes. Ph. D. thesis, Université scientifique et médicale de Grenoble, Grenoble, FRA.

Cousot, P. 1981. Semantic foundations of program analysis. In S. Muchnick and N. Jones Eds., *Program Flow Analysis: Theory and Applications*, Chapter 10, 303–342. Prentice-Hall.

Cousot, P. 1995. Abstract model-checking, invited lecture. In *7th Int. Conf. CAV '95* (Liège, BEL, 5 jul 1995).

Cousot, P. 1996. Abstract interpretation. *Symposium on Models of Programming Languages and Computation, ACM Comput. Surv. 28*, 2, 324–328.

Cousot, P. and Cousot, R. 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th POPL*, 238–252. ACM Press.

Cousot, P. and Cousot, R. 1979. Systematic design of program analysis frameworks. In *6th POPL*, 269–282. ACM Press.

Cousot, P. and Cousot, R. 1992a. Abstract interpretation and application to logic programs ⋆. *J. Logic Prog. 13*, 2–3, 103–179.

Cousot, P. and Cousot, R. 1992b. Abstract interpretation frameworks. *J. Logic and Comp. 2*, 4 (aug), 511–547.

Cousot, P. and Cousot, R. 1992c. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, invited paper. In M. Bruynooghe and M. Wirsing Eds., *Proc. Int. Work. PLILP '92*, LNCS 631, 269–295. Springer-Verlag.

Cousot, P. and Halbwachs, N. 1978. Automatic discovery of linear restraints among variables of a program. In *5th POPL*, 84–97. ACM Press.

Cridlig, R. 1995. Semantic analysis of shared-memory concurrent languages using abstract model-checking. In *Proc. PEPM '95*. ACM Press.

Cridlig, R. 1996. Semantic analysis of concurrent ML by abstract model-checking. In B. Steffen and T. Margaria Eds., *Proc. Int. Work. on Verification of Infinite State Systems*, vol. MIP-9614. Universität Passau, GER.

Dams, D., Gerth, R., and Grumberg, O. 1997. Abstract interpretation of reactive systems. *Trans. Prog. Lang. Sys.*, 19(2):253–291. ACM Press.

Daws, C., Olivero, A., Tripakis, S., and Yovine, S. 1996. The tool KRONOS. In R. Alur, T. Henzinger, and E. Sontag Eds., *Hybrid Systems III, Verification and Control*, LNCS 1066, 208–219. Springer-Verlag.

Dill, D.L., and Wong-Toi, H. 1995. Verification of real-time systems by successive over and under approximations. In P. Wolper Ed., *Proc. 7th Int. Conf. CAV '95*, LNCS 939, 409–422. Springer-Verlag.

Fernandez, J.-C. 1993. Abstract interpretation and verification of reactive systems. In P. Cousot, P. Falaschi, G. Filé, and A. Rauzy Eds., *Proc. 3rd Int. Work. WSA'93 on Static Analysis*, LNCS 724, 60–71. Springer-Verlag.

Graf, S. and Loiseaux, C. 1993. A tool for symbolic program verification and abstraction. In C. Courcoubatis Ed., *Proc. 5th Int. Conf. CAV '93*, LNCS 697, 71–84. Springer-Verlag.

Halbwachs, N. 1993. Delays analysis in synchronous programs. In C. Courcoubatis Ed., *Proc. 5th Int. Conf. CAV '93*, LNCS 697, 333–346. Springer-Verlag.

Halbwachs, N. 1994. About synchronous programming and abstract interpretation. In B. Le Charlier Ed., *Proc. SAS '94*, LNCS 864, 179–192. Springer-Verlag.

Halbwachs, N., Proy, J.-É., and Raymond, P. 1994. Verification of linear hybrid systems by means of convex approximations. In B. Le Charlier Ed., *Proc. SAS '94*, LNCS 864, 223–237. Springer-Verlag.

Halbwachs, N. and Raymond, P. 1996. On the use of approximations in symbolic model-checking. Tech. rep. SPECTRE L21 (jan), VERIMAG laboratory, Grenoble, FRA.

Henzinger, T. and Ho, P.-H. 1995. Algorithmic analysis of nonlinear hybrid systems. In P. Wolper Ed., *Proc. 7th Int. Conf. CAV '95*, LNCS 939, 225–238. Springer-Verlag.

Henzinger, T., Nicollin, X., Sifakis, J., and Yovine, S. 1992. Symbolic model-checking for real-time systems. In *Proc. 5th LICS'92*. IEEE Comp. Soc. Press.

Jackson, D. 1994. Abstract model-checking of infinite specifications. In M. Naftalin, T. Denvir, and M. Bertran Eds., *2nd Int. Symp. of Formal Methods Europe FME '94: Industrial Benefit of Formal Methods*, LNCS 873, 519–531. Springer-Verlag.

Karr, M. 1976. Affine relationships among variables of a program. *Acta Inf. 6*, 133–151.

Kelb, P. 1994. Model checking and abstraction: A framework approximating both truth and failure information. Technical report, University of Oldenburg.

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

Mauborgne, L. 1998. Abstract interpretation using typed decision graphs. *Sci. Comput. Prog.*, 31(1):91–112.

Queille, J.P. and Sifakis, J. 1982. Specification and verification of concurrent systems in *Cesar*. In *Int. Symp. on Programming*, LNCS 137. Springer-Verlag.

Steffen, B. 1991. Data flow analysis as model-checking. In T. Ito and A. Meyer Eds., *Proc. Int. Conf. TACS '91*, 346–364. Springer-Verlag.

Touati, H., Savoj, H., Lin, B., Brayton, R., and Sangiovanni-Vincentelli, S. 1990. Implicit state enumeration of finite state machines using BDDs. In A. Sangiovanni-Vincentelli and S. Goto Eds., *Proc. Int. Conf. ICCAD '90*, 130–133. IEEE Comp. Soc. Press.

Valmari, A. 1993. On-the-fly verification with stubborn sets. In C. Courcoubatis Ed., *Proc. 5th Int. Conf. CAV '93*, LNCS 697, 397–96. Springer-Verlag.

[*] The editor of JLP has mistakenly published the unreadable galley proof. For a correct version of this paper, see `http://www.dmi.ens.fr/~cousot`.