Parallel Combination of Abstract Interpretation and Model-Based Automatic Analysis of Software

Patrick Cousot
École Normale Supérieure
DMI, 45, rue d'Ulm
75230 Paris cedex 05
France
cousot@dmi.ens.fr
http://www.ens.fr/~cousot

and

Radhia Cousot
CNRS & École Polytechnique
LIX
91440 Palaiseau cedex
France
rcousot@lix.polytechnique.fr
http://lix.polytechnique.fr/~radhia

Abstract

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

A first category concerns *symbolic methods* where properties of the system are approximated using abstract domains. In this case, one considers approximated representations of sets of states.

A second category concerns *abstract model checking* where 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. In this other case, one considers approximated representations of sets of transitions.

The objective of this paper is to develop a third complementary possibility of interaction between abstract interpretation and model-checking based software analysis methods.

Here no approximation is made on sets of states or sets of transitions. Instead one performs an analysis of the system by abstract interpretation. This information is used to restrict the space of states and transitions which need to be explored during the verification process. The computational overhead of computing an abstract interpretation of a model to be checked can be avoided by doing the computation in parallel with the model checking and using intermediate abstract interpretation results as they become available.

AAS'97
First ACM SIGPLAN Workshop
on Automatic Analysis of Software
Paris, France, 14 January 1997

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 (quasi)-finite state 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. 1994; Fernandez 1993; Halbwachs 1994 and Steffen 1991).

We would like to consider here *abstract model-based automatic analysis* that is the model-based automatic analysis methods which are related to abstract interpretation and suggest further possible interactions.

First, *symbolic verification* [Burch et al. 1992; Henzinger et al. 1992; Daws et al. 1996] makes use of a 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]. Such abstract domains are of very common use when abstract interpretation is applied to program static analysis. Some symbolic abstract domains satisfy the chain condition [Karr 1976] and this directly guarantees the finite convergence of the analysis. However, most symbolic domains are very large or infinite so that, if one does not want to abandon the formal verification for lack of space or time, some form of widening [Cousot and Cousot 1992c] must ultimately be used to enforce rapid convergence of the analysis algorithms. Examples of widenings are given by Halbwachs 1993, 1994 for affine inequality relations and Mauborgne 1994 for BDDs. In this case, one does not consider a faithful symbolic description of the software properties but instead an approximation of sets of states. The corresponding loss of information may be without consequences for the verification [Henzinger and Ho 1995; Jackson 1994], else it fails.

In a second form of *reduction by abstraction*, one considers exact properties of an approximate semantics. More precisely, one does not consider a faithful description of the software runtime behavior but instead an approximation of this semantical behavior. Once again, abstract interpretation has been used to obtain such sound approximations. Here, the main idea for model checking or state exploration of infinite or very large finite transition systems is to use an abstract conservative finite transition system on which existing algorithms designed for finite automata are directly applicable. In this context, conservative means upper-approximation for safety ($\forall$) properties and lower-approximation for liveness ($\exists$) properties. 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. 1994; Cleaveland et al. 1995] or μ-calculus formulæ [Graf and Loiseaux 1993; Loiseaux et al. 1995; Cridlig 1995, 1996]. Partial-order approaches can be understood in this way, the loss of information being in this case without consequences on the completeness [Valmari 1993].

We would like here to suggest a new third 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, in parallel with this verification, that is at almost no cost in time.

2 Combining Abstract Interpretation and Model-Checking

The general idea is to improve the efficiency of symbolic model checking algorithms for verifying concurrent systems by using properties of the system that can be automatically inferred by abstract interpretation.

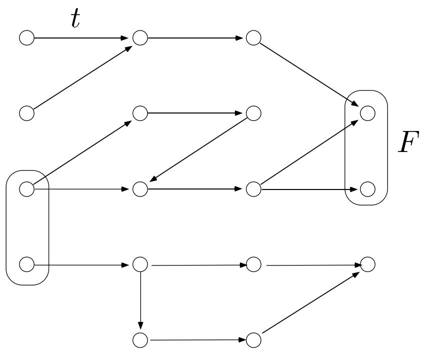


Figure 1: A (finite) transition system (o:state, →:transition)

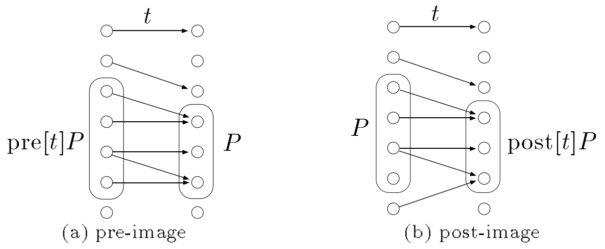


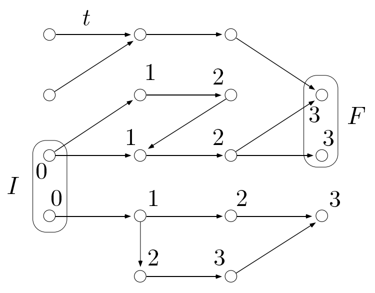
Figure 2: Pre- and post-image

2.1 Transition Systems

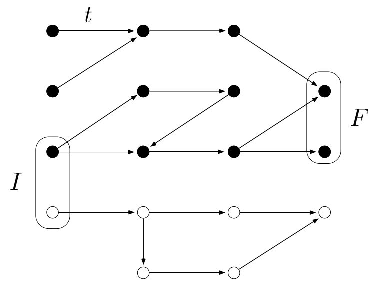
The considered (real-time) concurrent system is assumed to be modeled by a *transition system*, that is tuple $\langle S, t, I, F\rangle$ where S is the set of *states*, $t \subseteq S \times S$ is the *transition relation*, $I \subseteq S$ is the set of *initial states* and $F \subseteq S$ is the set 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 *complement* $\neg P$ of a set of states $P \subseteq S$ is $\{s \in S \mid s \notin P\}$. The *left-restriction* $P \rceil t$ of a relation t 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 transition relation t are defined inductively by $t^0 \triangleq 1_S \triangleq \{\langle s, s\rangle \mid s \in S\}$ (that is identity on states S) and $t^{n+1} \triangleq t \circ t^n$ for $n \geq 0$. The *reflexive transitive closure* $t^\star$ of the transition relation t is $t^\star \triangleq \bigcup_{n\geq 0} t^n$.

The *pre-image* $\text{pre}[t]\,P$ of a set $P \subseteq S$ of states by a transition relation t is $\text{pre}[t]\,P \triangleq \{s \mid \exists s' : \langle s, s'\rangle \in t \wedge s' \in P\}$. The *post-image* $\text{post}[t]\,P$ of a set $P \subseteq S$ of states by a transition relation t is $\text{post}[t]\,P \triangleq \{s' \mid \exists s : s \in P \wedge \langle s, s'\rangle \in t\}$. This is illustrated in Figure 2(a) and Figure 2(b). We have the least fixpoint characterizations $\text{pre}[t^\star]\,P = \text{lfp}^{\subseteq}\,\lambda X \bullet P \cup$

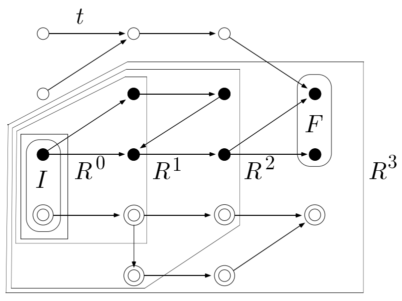


(a) Minimum delays

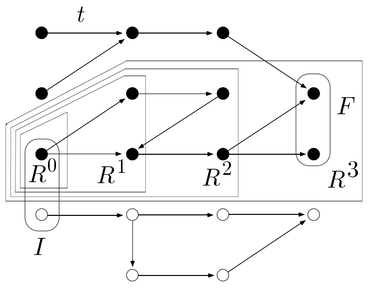


(b) Ascendants of the final states $\mathrm{pre}[t^\star]\,F$

Figure 3



(a) Algorithm "minimum1"



(b) Algorithm "minimum2"

Figure 4: Execution trace of minimum algorithm

$\mathrm{pre}[t]\,X$ and $\mathrm{post}[t^\star]\,P = \mathrm{lfp}^{\subseteq}\,\lambda X \bullet P \cup \mathrm{post}[t]\,X$ (see e.g. Cousot 1978, 1981).

2.2 Minimum Delay Problem

The *minimum delay problem* (see e.g. Halbwachs 1993) 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.

$$\begin{aligned}\ell &\triangleq \min\{n \mid \exists s \in I, s' \in F : \langle s,\, s'\rangle \in t^n\}\\ \min\emptyset &\triangleq \infty\end{aligned}$$

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

The following symbolic model checking *minimum delay algorithm* is due to Campos et al. 1995:

```
procedure minimum1 ($I$, $F$);
$R := I$;
$n := 0$;
stable := $(R \cap F \neq \emptyset)$;
while ¬stable do
    $R' := R \cup \mathrm{post}[t]\,R$;
    $n := n + 1$;
    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(a). 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 lead to erroneous results. Only a lower or upper bound of the minimum delay can be obtained in this way. 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(a).

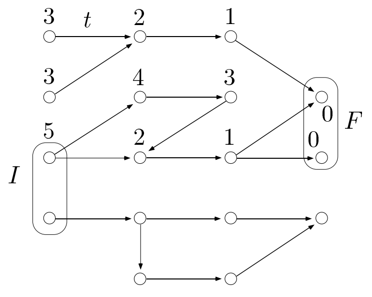
However, we can still use abstract interpretation to cut down the size of the model-checking search space by determining a super-set A of the ascendants of the final states (the principle of determination of A by abstract interpretation will be precisely defined in Section 4.1):

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

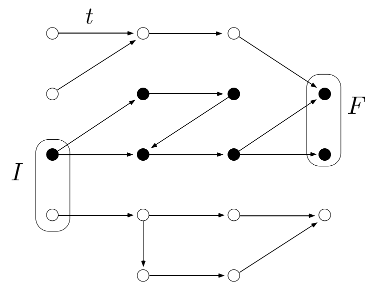
as illustrated in Figure 3(b), which can then be used to restrict the exploration of the transition graph for computing the minimum delay. The *revisited minimum delay algorithm* is now:

```
procedure minimum2 ($I$, $F$);
$R := I$;
$n := 0$;
stable := $(R \cap F \neq \emptyset)$;
while ¬stable do
    $R' := R \cup \boxed{(\mathrm{post}[t]\,R \cap A)}$;
    $n := n + 1$;
    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 4(b).

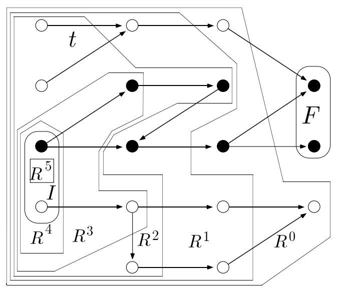


(a) Maximum delays

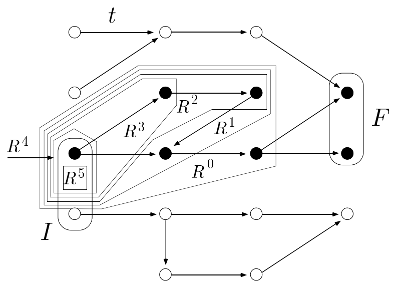


(b) Descendants of the initial states I which are ascendants of the final states F

Figure 5



(a) Algorithm "maximum1"



(b) Algorithm "maximum2"

Figure 6: Execution trace of maximum algorithm

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".

2.3 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 state in F:

$$\begin{aligned} m &\triangleq \max\{n \mid \exists s \in I, s' \in F : \langle s,\, s'\rangle \in (\neg F \rceil\, t)^n\} \\ \max \mathbb{N} &\triangleq \infty \end{aligned}$$

An example of maximum delays is given in Figure 5(a). The following *maximum delay algorithm* has been proposed by Campos et al. 1995:

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

An example of execution trace of the "maximum1" algorithm is given in Figure 6(a). 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 $\circ$ in Figure 6(a)) 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.

Once again an automatic analysis by abstract interpretation can determine a super-set U of the descendants of the initial states I which are ascendants of the final states F (the principle of determination of U by abstract interpretation will be precisely defined in Section 4.3):

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

The set of descendants of the initial states I which are ascendants of the final states F is illustrated by Figure 5(b). This leads to a *revisited maximum delay algorithm*, as follows:

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

An example of execution trace of the "maximum2" algorithm is given in Figure 6(b). Observe that any upper-approximation $\mathrm{post}[t^\star]\, I \cap \mathrm{pre}[t^\star]\, F \subseteq U$ of the descendants of the initial states I which are ascendants 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 $\mathrm{post}[t^\star]\, I \cap \mathrm{pre}[t^\star]\, F$ 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 polyhedra), the intersection $\mathrm{pre}[t]\, R' \cap (U - F)$ may be a BDD (or polyhedra) of much greater size than $\mathrm{pre}[t]\, R'$, 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].

3 Classical Abstract Interpretation Problems

Finding upper (or dually lower) approximations of the sets of:

- descendants $\mathrm{post}[t^\star]\, I$ of the initial states I [Cousot 1981];
- ascendants $\mathrm{pre}[t^\star]\, F$ of the final states F [Cousot 1981]; and
- descendants $\mathrm{post}[t^\star]\, I \cap \mathrm{pre}[t^\star]\, F$ of the initial states I which are ascendants of the final states F [Cousot 1978; Cousot and Cousot 1992a];

are classical problems in abstract interpretation with applications to:

- optimizing compilers;
- parallelization, vectorization, partial evaluation, program transformation;
- abstract debugging, and the like.

The following example of PASCAL program analysis of the descendants of the initial states using an interval approximation of set of possible values of variables [Cousot and Cousot 1977; Cousot 1981] has been given by Bourdoncle 1993. All comments { ...} have been generated automatically by the analyzer. They clearly show that non-trivial information about the infinite state space (in this case run-time values of variables at each program points) can be determined automatically by abstract interpretation:

```
program Variant_of_function_91_of_McCarthy;
   var X, Y : integer;
   function F(X : integer) : integer;
   begin
      if X > 100 then
         F := X - 10
         { F ∈ [91, maxint - 10] }
      else
         F := F(F(F(F(X + 33))));
         { F ∈ [91, 93] }
      { F ∈ [91, maxint - 10] }
   end;
begin
   readln(X);
   Y := F(X);
   { Y ∈ [91, maxint - 10] }
end.
```

It has been proved automatically that the result of F is necessarily greater than or equal to 91, if the call ever terminates.

The following example of approximation of the descendants of the initial states which are ascendants of the final states using interval analysis has also been given by Bourdoncle 1993. The comment {% true? %} has been included by the programmer. It is an intermittent assertion specifying the final states in that it stipulates that program execution should definitely terminate.

```
program Variant_of_function_91_of_McCarthy;
   var X, Y : integer;
   function F(X : integer) : integer;
   begin
      if X > 100 then F := X - 10
      else F := F(F(F(F(F(F(F(F(F(X + 90)))))))));
   end;
begin
   readln(X);
   { X > 100 }
   Y := F(X);
   {% true? %}
end.
```

The other comment { X > 100 } has been automatically generated by the abstract debugger (for short, the other automatically generated comments are not shown). If not satisfied, the program must necessarily go wrong either because of an inevitable run-time error (such as out of memory) or because of certain nontermination. This is precisely the case because of cycles such as F(100) → F(190) → F(180) → F(170) → F(160) → F(150) → F(140) → F(130) → F(120) → F(110) → F(100) → and so on. The intended constant was not 90 but 91! This shows that the restriction of the set of states of a transition system to those that lie on a path from an initial state to a final state by a preliminary abstract interpretation can cut down infinite paths.

4 Parallel Combination of Abstract Interpretation and Model Checking

Abstract interpretation is a theory of semantic approximation [Cousot 1996]. Here approximation means logical implication i.e. subsets of states inclusion. Moreover 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 the descendants $\mathrm{post}[t^\star]\, I \cap \mathrm{pre}[t^\star]\, F$ of the initial states I which are ascendants of the final states F [Cousot 1978; Cousot and Cousot 1992a]. We briefly recall how the upper-approximations D of $\mathrm{post}[t^\star]\, I$, A of $\mathrm{pre}[t^\star]\, F$ and U of $\mathrm{post}[t^\star]\, I \cap \mathrm{pre}[t^\star]\, F$ can be automatically computed by abstract interpretation. This is necessary to show how 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.

4.1 Forward Program Analysis by Abstract Interpretation

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

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

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[1] partially ordered by $\sqsubseteq$ such that:

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

It follows that α and γ are necessarily monotonic. Moreover any concrete property $P \in \wp(S)$ has a best (i.e. most precise) upper approximation $\alpha(P)$ in L, such that $P \subseteq \gamma(\alpha(P))$. We write $\langle \wp(S),\ \subseteq\rangle \ \underset{\alpha}{\overset{\gamma}{\begin{smallmatrix}\longleftarrow\\ \twoheadrightarrow\end{smallmatrix}}}\ \langle L,\ \sqsubseteq\rangle$ when α is surjective (or equivalently γ is injective or $\alpha \circ \gamma = 1_L$ is the identity on L). In this case, the poset $\langle L,\ \sqsubseteq\rangle$ is necessarily is a complete lattice $\langle L,\ \sqsubseteq,\ \bot,\ \top,\ \sqcup,\ \sqcup\rangle$ 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. For example, Clarke et al. 1992, Cleaveland et al. 1995, Dams et al. 1994 and others implicitly consider the Galois connection $\langle \wp(S),\ \subseteq\rangle \ \underset{\alpha}{\overset{\gamma}{\leftrightarrows}}\ \langle \wp(A),\ \subseteq\rangle$, where S is the set of concrete states and A is the set of abstract states, is necessarily of the form $\alpha(X) \triangleq \{h(x) \mid x \in X\}$ and $\gamma(Y) \triangleq \{x \mid h(x) \in Y\}$ where $h \in S \mapsto A$ is the approximation mapping. If h is surjective (as assumed e.g. in Jackson 1994), then so is α whence $\langle \wp(S),\ \subseteq\rangle \ \underset{\alpha}{\overset{\gamma}{\begin{smallmatrix}\longleftarrow\\ \twoheadrightarrow\end{smallmatrix}}}\ \langle \wp(A),\ \subseteq\rangle$.

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 $\langle M,\ \preceq\rangle \ \underset{\alpha}{\overset{\gamma}{\leftrightarrows}}\ \langle L,\ \sqsubseteq\rangle$, $T \in M \vdash\!\mathrm{m}\!\rightarrow M$ and $T^\sharp \in L \vdash\!\mathrm{m}\!\rightarrow L$ are monotonic and $\forall y \in L : \alpha \circ T \circ \gamma(y) \preceq T^\sharp(y)$ then $\mathrm{lfp}^{\preceq}\, T \preceq \gamma(\mathrm{lfp}^{\sqsubseteq}\, T^\sharp)$ and equivalently $\alpha(\mathrm{lfp}^{\preceq}\, T) \sqsubseteq \mathrm{lfp}^{\sqsubseteq}\, T^\sharp$, see Cousot and Cousot 1979. So let $\mathcal{F} \in L \vdash\!\mathrm{m}\!\rightarrow L$ be such that $\alpha \circ (\lambda X \bullet I \cup \mathrm{post}[t]\, X) \circ \gamma \sqsubseteq \mathcal{F}$, pointwise. The transfinite iteration sequence $\bar{\mathcal{F}}^0 \triangleq \alpha(\emptyset)$, $\bar{\mathcal{F}}^{\delta+1} \triangleq \mathcal{F}(\bar{\mathcal{F}}^\delta)$, $\bar{\mathcal{F}}^\lambda \triangleq \bigsqcup_{\delta<\lambda} \bar{\mathcal{F}}^\delta$ for limit ordinals is ultimately stationary and converges to $\mathrm{lfp}^{\sqsubseteq}\, \mathcal{F}$. This directly leads to an iterative algorithm which is finitely convergent when L satisfies the ascending chain condition[2].

In general however, the iterates $\bar{\mathcal{F}}^\delta$, $\delta \geq 0$ do not converge to $\mathrm{lfp}^{\sqsubseteq}\, \mathcal{F}$ in finitely many steps, so that one must resort to a *widening operator* ∇ which can be used both to upper-approximate inexisting lubs [as in e.g. Cousot and Halbwachs 1978] and to enforce finite convergence of increasing iterations [Cousot and Cousot 1977]. The widening operator $\nabla \in L \times L \mapsto L$ should be an upper bound (that is $\forall x, y \in L : x \sqsubseteq x \nabla y$ and $\forall x, y \in L : y \sqsubseteq x \nabla y$) and enforce finite convergence (for all increasing chains $x^0 \sqsubseteq x^1 \sqsubseteq \ldots \sqsubseteq x^i \sqsubseteq \ldots$ the increasing chain defined by $y^0 = x^0, \ldots, y^{i+1} = y^i \nabla x^{i+1}, \ldots$ is not strictly increasing).

The *upward iteration sequence with widening* is $\hat{\mathcal{F}}^0 \triangleq \alpha(\emptyset)$, $\hat{\mathcal{F}}^{i+1} \triangleq \hat{\mathcal{F}}^i$ if $\mathcal{F}(\hat{\mathcal{F}}^i) \sqsubseteq \hat{\mathcal{F}}^i$ and $\hat{\mathcal{F}}^{i+1} \triangleq \hat{\mathcal{F}}^i \nabla \mathcal{F}(\hat{\mathcal{F}}^i)$ otherwise. It is ultimately stationary and its limit $\hat{\mathcal{F}}$ is a sound upper approximation of $\mathrm{lfp}^{\sqsubseteq}\, \mathcal{F}$ in that $\mathrm{lfp}^{\sqsubseteq}\, \mathcal{F} \sqsubseteq \hat{\mathcal{F}}$.

If $\mathcal{F}(\hat{\mathcal{F}}) \sqsubset \hat{\mathcal{F}}$ and the iterates $\check{\mathcal{F}}^0 \triangleq \hat{\mathcal{F}}$, $\check{\mathcal{F}}^{\delta+1} \triangleq \mathcal{F}(\check{\mathcal{F}}^\delta)$ and $\check{\mathcal{F}}^\lambda \triangleq \bigsqcap_{\delta<\lambda} \check{\mathcal{F}}^\delta$ for limit ordinals 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 such that $\forall x, y \in L : x \sqsubseteq y \Longrightarrow x \sqsubseteq x \triangle y \sqsubseteq y$ and 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 $\mathcal{F}(X) = X \sqsubseteq \mathcal{F}(\hat{\mathcal{F}}) \sqsubseteq \hat{\mathcal{F}}$ then the *downward iteration sequence with narrowing* is defined by $\check{\mathcal{F}}^0 \triangleq \hat{\mathcal{F}}$, $\check{\mathcal{F}}^{i+1} \triangleq \check{\mathcal{F}}^i$ if $\mathcal{F}(\check{\mathcal{F}}^i) = \check{\mathcal{F}}^i$ and $\check{\mathcal{F}}^{i+1} \triangleq \check{\mathcal{F}}^i \triangle \mathcal{F}(\check{\mathcal{F}}^i)$ otherwise. This iteration sequence is ultimately stationary and its limit $\check{\mathcal{F}}$ is a sound upper approximation of the fixpoint $X \sqsubseteq \check{\mathcal{F}} \sqsubseteq \hat{\mathcal{F}}$ which is better than the one $\hat{\mathcal{F}}$ obtained by widening. In conclusion $\mathrm{lfp}^{\sqsubseteq}\, \mathcal{F} \sqsubseteq \check{\mathcal{F}} \sqsubseteq \hat{\mathcal{F}}$ 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 $D \triangleq \gamma(\check{\mathcal{F}})$.

As already mentioned design of the abstract algebra $\langle L,\ \sqsubseteq,\ \bot,\ \top,\ \sqcup,\ \sqcap,\ \nabla,\ \triangle,\ f_1,\ \ldots,\ f_n\rangle$ and of the transformer $\mathcal{F}$ (usually composed out of the primitives $f_1,\ \ldots,\ f_n$) are problem dependent and will not be further considered here.

4.2 Backward Program Analysis by Abstract Interpretation

The situation is similar for computing an upper approximation A of $\mathrm{pre}[t^\star]\, F = \mathrm{lfp}^{\subseteq}\, \lambda X \bullet F \cup \mathrm{pre}[t]\, X$ using $\mathcal{B} \in L \vdash\!\mathrm{m}\!\rightarrow L$ such that $\alpha \circ (\lambda X \bullet F \cup \mathrm{pre}[t]\, X) \circ \gamma \sqsubseteq \mathcal{B}$, pointwise[3]. 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 $\mathrm{lfp}^{\sqsubseteq}\, \mathcal{B} \sqsubseteq \check{\mathcal{B}} \sqsubseteq \hat{\mathcal{B}}$ 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 $A \triangleq \gamma(\check{\mathcal{B}})$.

[1]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].

[2]Any strictly ascending chain $x_0 \sqsubset x_1 \sqsubset \cdots$ of elements of L is necessarily finite.

[3]More generally one could consider a different abstract domain for backward analysis, the generalization being immediate.

4.3 Combining Forward and Backward Program Analysis by Abstract Interpretation

In order to upper approximate $\mathrm{post}[t^\star]\, I \cap \mathrm{pre}[t^\star]\, F$, we use $\mathcal{F}$ which is $\lambda X \bullet I \cup \mathrm{post}[t]\, X$ up to abstraction and $\mathcal{B}$ which is $\lambda Y \bullet \mathrm{pre}[t]\, Y \cup F$ up to abstraction. The following approximation sequence 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 for $\mathcal{F}$ and $\ddot{U}^0$ is the limit of the corresponding downward iteration sequence with narrowing[4];
- ...
- $\dot{U}^{2n+1}$ is the limit of the upper upward iteration sequence with widening for $\lambda X \bullet (\ddot{U}^{2n} \sqcap \mathcal{F}(X))$ 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 $\lambda X \bullet (\ddot{U}^{2n+1} \sqcap \mathcal{B}(X))$ and $\ddot{U}^{2n+2}$ is the limit of the corresponding downward iteration sequence with narrowing.
- ...

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

4.4 Parallel Abstract Interpretation and Model Checking

We are now in position to explain how the verification by model-checking can interact in parallel with the analysis of the system by abstract interpretation.

Consider for example algorithm "maximum2". Execution of algorithm "maximum2" is started in parallel with the computation of the upper approximation sequence $\top$, $\dot{U}^0$, $\ddot{U}^0$, ..., $\dot{U}^{2n+1}$, $\ddot{U}^{2n+1}$, $\dot{U}^{2n+2}$, $\ddot{U}^{2n+2}$, ... At each iteration of the main loop in "maximum2", one can chose U as the element in this sequence which is currently available[5]. Observe that if the lattice L does not satisfy the descending chain condition[6], this approximation sequence may be infinite. We can enforce finite convergence using a narrowing as suggested in [Cousot 1978; Cousot and Cousot 1992a] or more simply stop the computation as soon as the parallel asynchronous 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 $\ldots \sqsupseteq \dot{U}^{2n+1} \sqsupseteq \ddot{U}^{2n+1} \sqsupseteq \dot{U}^{2n+2} \sqsupseteq \ddot{U}^{2n+2} \sqsupseteq \ldots$ 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.

[4]Depending of the considered problem, it might be semantically equivalent but more efficient to start with $\mathcal{B}$ instead of $\mathcal{F}$.

[5]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.

[6]Any strictly descending chain $x_0 \sqsupset x_1 \sqsupset \cdots$ of elements of L is necessarily finite.

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 $\mathcal{B}$ are not upper approximations of $\mathrm{pre}[t^\star]\, F$. 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 $\check{\mathcal{B}}^0 = \hat{\mathcal{B}}$, $\check{\mathcal{B}}^1$, ... of the corresponding downward iteration sequence with narrowing as successive values of A in "minimum2". 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 $\mathrm{gfp}^{\sqsubseteq}\, \mathcal{B}$ since they are all upper approximations of $\mathrm{lfp}^{\sqsubseteq}\, \mathcal{B}$ and better than S.

5 Conclusion

Existing combinations of model-checking and abstract interpretation have been concerned with the symbolic representation of abstract properties and the approximation of the state transition relation, in both cases with or without loss of information. Building upon [Cousot 1995], we have proposed another form of combination which, by a preliminary analysis of the system (in forward, backward or combined direction) or better, in parallel with verification, one can reduce the size of the part of the state graph that has to be explored (in the other direction) for verification by exhaustive or on-the-fly model-checking. The combination is at almost no cost since the parallel execution of the abstract interpreter and the model checker is asynchronous, abstract properties being used by the model checker as they become available. Other forms of restrictions have been proposed by Halbwachs and Raymond 1996 which are amenable to parallelization in a similar way.

This method, which makes no approximation on the states and transitions of the model, is 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 AAS '97 referees for their comments.

References

AKERS, S. 1978. Binary decision diagrams. *IEEE Trans. Comput. C-27*, 6.

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. PLDI*, pp. 46–55. ACM Press.

BRYANT, R. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput. C-35*, 8.

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* (La Jolla, Calif., jun 21–22, 1995), pp. 75–83.

CLARKE, E., EMERSON, E., AND SISTLA, A. 1983. Automatic verification of finite-state concurrent systems using temporal logic specifications. In *10th POPL* (jan 1983). ACM Press.

CLARKE, E., GRUMBERG, O., AND LONG, D. 1992. Model checking abstraction. In *19th POPL* (Albuquerque, N.M., 1992), pp. 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*, Glasgow UK, 25–27 sep 1995, LNCS 983, pp. 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, Grenoble, France, 1989*, LNCS 407, pp. 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, pp. 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, pp. 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* (Los Angeles, Calif., 1977), pp. 238–252. ACM Press.

COUSOT, P. AND COUSOT, R. 1979. Systematic design of program analysis frameworks. In *6th POPL* (San Antonio, Texas, 1979), pp. 269–282. ACM Press.

COUSOT, P. AND COUSOT, R. 1992a. Abstract interpretation and application to logic programs. *J. Logic Prog. 13*[7], 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*, Leuven, BEL, 13–17 aug 1992, LNCS 631, pp. 269–295. Springer-Verlag.

COUSOT, P. AND HALBWACHS, N. 1978. Automatic discovery of linear restraints among variables of a program. In *5th POPL* (Tucson, Ariz., 1978), pp. 84–97. ACM Press.

CRIDLIG, R. 1995. Semantic analysis of shared-memory concurrent languages using abstract model-checking. In *Proc. PEPM '95*, La Jolla, Calif. (21–23 jun 1995). ACM Press.

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

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 (aug 1996). Universität PassauGER.

DAMS, D., GRUMBERG, O., AND GERTH, R. 1994. Abstract interpretation of reactive systems: Abstractions preserving $\forall$CTL*, $\exists$CTL* and CTL*. In E. OLDEROG Ed., *Proc. IFIP WG2.1/WG2.2/WG2.3 Working Conf. on Programming Concepts, Methods and Calculi (PROCOMET)*, IFIP Transactions (jun 1994). North-Holland/Elsevier.

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 (1996), pp. 208–219. 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*, Padova, ITA, LNCS 724, pp. 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*, Elounda, GRE, LNCS 697, pp. 71–84. Springer-Verlag.

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

HALBWACHS, N. 1994. About synchronous programming and abstract interpretation. In B. LE CHARLIER Ed., *Proc. SAS '94*, Namur, BEL, 20–22 sep 1994, LNCS 864, pp. 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*, Namur, BEL, 20–22 sep 1994, LNCS 864, pp. 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, France.

HENZINGER, T. AND HO, P.-H. 1995. Algorithmic analysis of nonlinear hybrid systems. In P. WOLPER Ed., *Proc. 7th Int. Conf. CAV '95*, Liège, BEL, LNCS 939 (3–5 jul 1995), pp. 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*, Barcelona ESP, LNCS 873, pp. 519–531. Springer-Verlag.

KARR, M. 1976. Affine relationships among variables of a program. *ACTAI2 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. 1994. Abstract interpretation using TDGs. In B. LE CHARLIER Ed., *Proc. SAS '94*, Namur, BEL, 20–22 sep 1994, LNCS 864, pp. 363–379. Springer-Verlag.

STEFFEN, B. 1991. Data flow analysis as model checking. In T. ITO AND A. MEYER Eds., *Proc. Int. Conf. TACS '91*, Sendai, JPN, pp. 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* (Santa Clara, Calif., nov 1990), pp. 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*, Elounda, GRE, LNCS 697, pp. 397–96. Springer-Verlag.