# A Technique for Invariant Generation*

**To be presented at Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2001, Genova, Italy, April 2001.**


A. Tiwari, H. Rueß, H. Saïdi, and N. Shankar

SRI International,
333 Ravenswood Ave,
Menlo Park, CA, U.S.A
{tiwari,ruess,saidi,shankar}@csl.sri.com

**Abstract.** Most of the properties established during verification are either invariants or depend crucially on invariants. The effectiveness of automated formal verification is therefore sensitive to the ease with which invariants, even trivial ones, can be automatically deduced. While the strongest invariant can be defined as the least fixed point of the strongest post-condition of a transition system starting with the set of initial states, this symbolic computation rarely converges. We present a method for invariant generation and strengthening that relies on the simultaneous construction of least and greatest fixed points, restricted widening and narrowing, and quantifier elimination. The effectiveness of the method is demonstrated on a number of examples.

## 1 Introduction

The majority of properties established during the verification of programs are either invariants or depend crucially on invariants. Indeed, safety properties can be reduced to invariant properties, and to prove progress one usually needs to establish auxiliary invariance properties too. Consequently, the discovery and strengthening of invariants is a central technique in the analysis and verification of both sequential programs and reactive systems, especially for infinite state systems.

Consider, for example, a program with state variables $pc$ and $x$. The program counter $pc$ is interpreted over the control locations $inc$ and $dec$, and $x$ is interpreted over the integers. Initially, the program counter $pc$ is set to $inc$ and $x$ to 0. The dynamics of the system is described in terms of the guarded commands:

$$pc = inc \;\longmapsto\; x := x + 2;\; pc := dec$$
$$pc = dec \wedge x > 0 \;\longmapsto\; x := x - 2;\; pc \in \{inc, dec\}$$

Suppose we are interested in establishing the invariant $pc = inc \rightarrow x = 0$. A naïve proof attempt fails, and consequently, the invariant needs to be strengthened to an inductive invariant $(pc = inc \rightarrow x = 0) \wedge (pc = dec \rightarrow x = 2)$. Such

* The research described in this paper was supported in part by NSF contract CCR-9712383 and DARPA/AFRL contract F33615-00-C-3043.

strengthenings are typically needed in induction proofs. In general, the main principle for proving that a predicate $\phi$ is an invariant of some program or system S, consists in finding an *auxiliary* predicate $\psi$ such that $\psi$ is stronger than $\phi$ and $\psi$ is inductive; i.e., every initial state of S satisfies $\psi$, and $\psi$ is preserved under all transitions. This rule is sound and (relatively) complete. On the other hand, finding a strengthening $\psi$ is not always obvious, and usually requires a microscopic examination of failed verification attempts.

Most approaches for generating and strengthening invariants are based on symbolic computation of the system at hand [4,10,15]. The *bottom-up* method performs an abstract forward propagation to compute the set of all reachable configurations, while the *top-down* method starts from an invariant candidate $\phi$ and performs an abstract backward propagation to compute a strengthened invariant $\psi$. There is, however, no guarantee for success in exact forward or backward propagation. This may be due either to infinite or unmanageably large configuration spaces or to the failure to detect convergence of the propagation methods altogether. Consequently, approximation techniques such as widening or narrowing [8] are needed to enforce termination of symbolic computation. The basic idea is to accelerate the convergence of symbolic computations in infinite abstract domains.

The framework of abstract interpretation with widening and narrowing as outlined in [8], however, is not immediately applicable to the discovery and strengthening of inductive invariants, since not every over-approximation of an inductive invariant is necessarily an inductive invariant. Our main contributions are: first, we provide an abstract description of the process of *inductive* invariant generation and strengthening based on computing under- and over-approximations of the reachable state set; second, this framework is instantiated with a novel technique based on combining concrete widening and narrowing operators. Our techniques can uniformly be used on a wide class of examples including transition systems where both forward and backward propagation do not converge. We demonstrate the effectiveness of our approach through a variety of examples.

Our algorithm is based on the symbolic computation of a sequence of under- and over-approximations of the reachable state set. These computations rely heavily on the elimination of quantifiers in the underlying theory. Quantifier elimination, however, is not required to return equivalent formulas, since our algorithm tolerates weakened quantifier-eliminated formulas. Whenever the computation of the sequence of under-approximations terminates, we get an inductive invariant. Moreover, since every element in the sequence of decreasing over-approximations is an inductive invariant, our algorithm can be stopped at *any time* and it outputs the best (strongest) inductive invariant computed up to this point. In the example above, our procedure yields the invariant $(pc = inc \to x = 0) \wedge (pc = dec \to x = 2)$.[1]

[1] This example can also be handled by some other invariant generation techniques based on forward reachability or abstraction [3,17].

The approach faces two problems. First, the computation of the sequence of under-approximations usually does not terminate. Second, the computation of the sequence of over-approximations terminates with very weak invariants, in practice. For instance, forward reachability does not converge in case the initial value for $x$ is unspecified in the example above. In order to overcome these problems we add specialized widening and narrowing operators to our algorithm. One of the distinguishing features of our algorithm is the use of unreachable configurations for detecting unreachable strongly connected components and computing corresponding narrowing operators. In this way, our algorithm terminates with the invariant $x > -2$ in case the initial value for $x$ is unspecified in our running example.

The paper is structured as follows. In Section 2 we introduce notation and definitions, Section 3 presents the theoretical framework that is used in Section 4 to obtain a procedure for generating invariants using affirmation and propagation rules along with widening and narrowing. Finally, we conclude in Section 5 with a short investigation of the relationship between invariant generation and abstract interpretation, and comparisons with related work.

## 2 Preliminaries

Let $\Sigma$ be a first-order language containing interpreted symbols for standard concrete domains like booleans, integers and reals. Let $\Re$ denote the (first-order) theory of interest over the language $\Sigma$. We fix the set $\mathcal{V} = \{x_1, \ldots, x_n\}$ of (typed) variables and denote by $\mathcal{F}$ the set of first-order formulas over $\Sigma$ with free variables contained in the set $\mathcal{V}$. A *transition system* S is a tuple $(\mathcal{V}, \Theta, \Phi)$, where $\Theta \in \mathcal{F}$ and $\Phi$ is a first-order formula over $\Sigma$ with free variables contained in the set $\mathcal{V} \cup \mathcal{V}'$, where $\mathcal{V}' = \{x'_1, \ldots, x'_n\}$. The formula $\Theta$ is called the *initial predicate* and the formula $\Phi$ a *transition predicate* of the system S. We shall denote the sequence $x_1, \ldots, x_n$ by $\boldsymbol{x}$ and the sequence $x'_1, \ldots, x'_n$ by $\boldsymbol{x'}$.

A *state* $\sigma$ of a transition system $\mathrm{S} = (\mathcal{V}, \Theta, \Phi)$ is a mapping from $\mathcal{V}$ to values from the corresponding domains. If $\rho$ is a state, we denote by $\rho'$ the mapping obtained by renaming variables $x_i$ to $x'_i$ in $\rho$. A formula $\phi(\boldsymbol{x})$ is interpreted as the set $[\![\phi(\boldsymbol{x})]\!]$ of all states $\sigma$ such that $\Re, \sigma \models \phi(\boldsymbol{x})$. We define the set $\mathit{Reach}(\Phi)(\Theta)$ of states *reachable* from the states represented by $\Theta$ via the transition predicate $\Phi$ as the smallest set such that (i) $[\![\Theta]\!] \subset \mathit{Reach}(\Phi)(\Theta)$ and (ii) the state $\sigma \in \mathit{Reach}(\Phi)(\Theta)$ whenever $\Re, \rho, \sigma' \models \Phi(\boldsymbol{x}, \boldsymbol{x'})$ for some $\rho \in \mathit{Reach}(\Phi)(\Theta)$. Since the theory $\Re$ is fixed, we shall not mention it explicitly when we talk about satisfiability and validity in $\Re$. Thus, validity in $\Re$ is denoted by $\models$.

A *formula transformer* $\Gamma$ is a function mapping formulas to formulas. The *strongest postcondition* transformer, denoted by $\mathtt{SP}(\Phi)$, is defined as $\mathtt{SP}(\Phi)(\phi(\boldsymbol{x})) = \exists \boldsymbol{y}.(\Phi(\boldsymbol{y}, \boldsymbol{x}) \wedge \phi(\boldsymbol{y}))$. The formula $\mathtt{SP}(\Phi)(\phi(\boldsymbol{x}))$ denotes the set of states reachable in one step from the set of states represented by $\phi$. Similarly, the *weakest precondition* transformer, $\mathtt{WP}(\Phi)$, is defined as $\mathtt{WP}(\Phi)(\phi(\boldsymbol{x})) = \forall \boldsymbol{y}.(\Phi(\boldsymbol{x}, \boldsymbol{y}) \rightarrow \phi(\boldsymbol{y}))$.

A *fixed point* of a formula transformer $\Gamma$ is a formula $\phi$ such that $\models \Gamma(\phi) \leftrightarrow \phi$. A formula transformer $\Gamma$ is *monotonic* if $\models \Gamma(\phi) \rightarrow \Gamma(\psi)$ whenever $\models \phi \rightarrow \psi$. A *least* fixed point of $\Gamma$, denoted by $\mu\psi.\Gamma(\psi)$, is a fixed point $\phi$ such that for any other fixed point $\psi$ of $\Gamma$, it is the case that $\models (\phi \rightarrow \psi)$. A *greatest* fixed point of $\Gamma$, denoted by $\nu\psi.\Gamma(\psi)$, is a fixed point $\phi$ such that for any other fixed point $\psi$ of $\Gamma$, it is the case that $\models (\psi \rightarrow \phi)$. Whenever the transition system $\langle \mathcal{V}, \Theta, \Phi \rangle$ is clear from the context, we define the transformer $\mathcal{I}$ by $\mathcal{I}(\phi) = \mathtt{SP}(\Phi)(\phi) \vee \Theta$. Note that the transformer $\mathcal{I}$ is monotonic. The least fixed point of this operator, $\mu\psi.\mathcal{I}(\psi)$, whenever it exists in the first-order language, represents the set $\mathit{Reach}(\Phi)(\Theta)$ of reachable states.

### 2.1 Invariants

A formula $\phi$ is an *S-invariant* if $\mathit{Reach}(\Phi)(\Theta) \subset [\![\phi]\!]$. Thus, an invariant describes an over-approximation of the set of reachable states. An *S-inductive invariant* is a formula $\phi$ such that (i) $\phi$ is an S-invariant, and (ii) $\phi$ is inductive, i.e., $\models \mathtt{SP}(\Phi)(\phi) \rightarrow \phi$. Condition (ii) can be equivalently stated as $\models \phi \rightarrow \mathtt{WP}(\Phi)(\phi)$. In other words, $\phi$ is an S-inductive invariant if $\models \mathcal{I}(\phi) \rightarrow \phi$. Note that the definition does not require an equivalence, but only an implication.

It is easy to establish that the set of reachable states $\mathit{Reach}(\Phi)(\Theta)$ of a system S represents the strongest (inductive) invariant. By this we mean that if $\psi$ is any other (inductive) invariant, then, $\mathit{Reach}(\mathrm{S})(\Theta) \subset [\![\psi]\!]$. However, note that if $\phi$ is an inductive invariant, and $\models (\phi \rightarrow \psi)$, then $\psi$ need not be an inductive invariant because $\psi$ might violate condition (ii). For purposes of this paper, we will only be interested in inductive invariants. Thus, we are *not* interested in just obtaining *any* over-approximation of the set of reachable states, but only those that also satisfy condition (ii). This is because the inductive property provides a sufficient local characterization of invariance property, which makes the task of proving easier.

Given a transition system $\mathrm{S} = (\mathcal{V}, \Theta, \Phi)$, the *converse* transition system $\mathrm{S}^{-1} = (\mathcal{V}, \Theta, \Phi^{-1})$ is defined by $\Phi^{-1}(\boldsymbol{x}, \boldsymbol{y}) = \Phi(\boldsymbol{y}, \boldsymbol{x})$. The following well-known theorem says that if none of the initial states is backward reachable from the states represented by $\phi$, then $\neg\phi$ is an invariant.

**Theorem 1.** *Let $S = \langle \mathcal{V}, \Theta, \Phi \rangle$ be a transition system and $\phi$ an arbitrary formula. If $\psi$ is such that $\models (\mathtt{SP}(\Phi^{-1})(\psi) \vee \phi) \rightarrow \psi$ and the formula $\Theta \wedge \psi$ is unsatisfiable, then $\neg\psi$ is an S-inductive invariant.*

**Corollary 1.** *If $\mathit{Reach}(\Phi^{-1})(\phi) \cap [\![\Theta]\!] = \emptyset$, then the formula corresponding to the complement of the set $\mathit{Reach}(\Phi^{-1})(\phi)$ is an S-inductive invariant.*

We remark here that although application of the $\mathtt{SP}(\Phi)$ transformer is called "forward propagation", the term "backward propagation" is typically used for the transformer $\mathtt{WP}(\Phi)$. But there is no anomaly here as the transformers $\mathtt{SP}(\Phi^{-1})$ and $\mathtt{WP}(\Phi)$ are duals in the sense that $\mathtt{SP}(\Phi^{-1})(\phi)$ is logically equivalent to $\neg\mathtt{WP}(\Phi)(\neg\phi)$. Hence, Theorem 1 can be stated in terms of $\mathtt{WP}(\Phi)$. It also follows that if formula $\phi$ is an invariant, then the formula $\nu\psi.\phi \wedge \mathtt{WP}(\Phi)(\psi)$ is an inductive invariant that is a strengthening of $\phi$[2]. Similarly, it is easy to see that there is a corresponding connection between the $\mathtt{SP}(\Phi)$ and $\mathtt{WP}(\Phi^{-1})$ transformers.

## 3 Inductive Invariant Generation

In this section, we discuss the problem of automatically generating some useful inductive invariants for a given transition system. It is a simple observation that the greatest fixed point $\nu\phi.\mathcal{I}(\phi)$, whenever it exists, is an S-inductive invariant.

**Lemma 1.** *Let $S = \langle \mathcal{V}, \Theta, \Phi \rangle$ be a transition system. Recursively define the sequence of formulas $\phi_0, \phi_1, \ldots,$ as follows.*

$$\phi_0 = \mathtt{true} \qquad\qquad \phi_{i+1} = \mathtt{SP}(\Phi)(\phi_i) \vee \Theta$$

*Then, every formula $\phi_i$ is an S-inductive invariant. Furthermore, every formula $\phi_i$ in the above sequence can be decomposed as $\psi_i \vee \chi_i$, where*

$$\begin{array}{ll} \psi_0 = \mathtt{false} & \qquad \psi_{i+1} = \mathtt{SP}(\Phi)(\psi_i) \vee \Theta \\ \chi_0 = \mathtt{true} & \qquad \chi_{i+1} = \mathtt{SP}(\Phi)(\chi_i). \end{array}$$

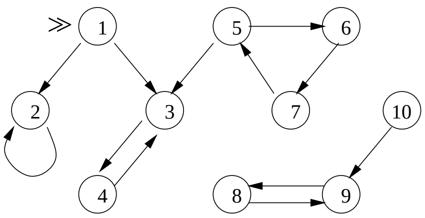
The sequence $\psi_0, \psi_1, \ldots,$ represents iterations in a least fixed point computation of the $\mathcal{I}$ transformer. The sequence $\chi_0, \chi_1, \ldots,$ represents the greatest fixed point component. The formulas $\psi_i$ provide successive under-approximations of the set $\mathit{Reach}(\Phi)(\Theta)$ of reachable states. The formulas $\phi_i$ are inductive over-approximations. The sequence $\psi_0, \psi_1, \ldots,$ usually does not terminate, whereas the sequence $\phi_0, \phi_1, \ldots,$ often terminates with very weak invariants.

It should be observed here that the greatest fixed point of the $\mathtt{SP}(\Phi)(\_) \vee \Theta$ transformer characterizes states $\sigma$ such that there exists a backward path starting from $\sigma$ which is either infinite, or contains some initial state. In case of finite state transition systems, this is exactly the set of states that either belong to a strongly connected component, or, that are reachable from either some initial state or some strongly connected component. Hence, the greatest fixed point may not be the strongest S-inductive invariant even in the case of finite systems. Despite its shortcomings, this simple method is attractive since (i) we do not need to detect that the iterations have converged[3], and (ii) every formula $\phi_i$ is an S-inductive invariant. Detecting convergence is difficult as it involves deciding if $\models \phi_i \leftrightarrow \phi_{i+1}$.

*Example 1.* Consider the transition system over ten states presented in Figure 1.

[2] It follows from this duality that the the least (greatest) fixed point iterations of $\mathtt{SP}(\Phi^{-1}) \vee \phi$ are logically equivalent to the negations of the greatest (least) fixed point iterations of $\mathtt{WP}(\Phi) \wedge \neg\phi$.

[3] If $\phi$ is an $S$-invariant, then every iteration in the greatest fixed point computation of $\mathtt{WP}(\Phi)(\_) \wedge \phi$ is also an $S$-invariant. But, if $\phi$ is inductive, then this method yields $\phi$.




States are represented by nodes with integer labels and transitions are represented by edges. State 1 is the initial state. Clearly, the set of reachable states is the set

$$\{1, 2, 3, 4\}.$$

The greatest fixed point of the $\mathtt{SP}(\Phi) \vee \Theta$ is the set consisting of states

$$\{1, 2, 3, 4, 5, 6, 7, 8, 9\}.$$

**Fig. 1.** A finite state transition system.

### 3.1 Widening and Narrowing

In the case when the state space is either infinite, or finite but too large, the symbolic computation of (greatest or least) fixed points of various transformers is restricted by the finite space and time resources available. A well-known solution to this problem is the use of *widening* and *narrowing* to respectively enhance the least and greatest fixed point computation (with gains obtained both in terms of space and time).

A *widening* operator $\nabla : \mathcal{F} \times \mathcal{F} \mapsto \mathcal{F}$ is a function such that for all formulas $\phi, \phi' \in \mathcal{F}$, $\models (\phi \vee \phi') \rightarrow \nabla(\phi, \phi')$. Similarly, a *narrowing* operator $\triangle : \mathcal{F} \times \mathcal{F} \mapsto \mathcal{F}$ is a function such that for all formulas $\phi, \phi' \in \mathcal{F}$, $\models \triangle(\phi, \phi') \rightarrow (\phi \wedge \phi')$. Thus, logical disjunction $\vee$ is a trivial widening operator, and logical conjunction $\wedge$ is a trivial narrowing operator.

The definitions of widening and narrowing are slightly different from the standard ones [8, 9]. First, we do not include any conditions to guarantee that increasing (decreasing) sequences are transformed to finite, hence converging, increasing (decreasing) sequences by widening (narrowing). Secondly, in the case of narrowing, the standard definition requires that whenever $\phi' \rightarrow \phi$, the formula $\triangle(\phi, \phi')$ is such that $\phi' \rightarrow \triangle(\phi, \phi')$ and $\triangle(\phi, \phi') \rightarrow \phi$. In our definition, $\triangle(\phi, \phi')$ is stronger than both $\phi$ and $\phi'$ as our interest is in the use of narrowing to obtain *under*-approximations of the greatest fixed point. But we have to be careful so as to not eliminate any reachable states by overly aggressive under-approximation, see Lemma 3.

A particularly simple narrowing operator, denoted by $\triangle(\psi)$, is defined by $\triangle(\psi)(\phi, \phi') = \phi \wedge \phi' \wedge \psi$, where $\psi$ is an arbitrary formula. Similarly, we can define $\nabla(\psi)(\phi, \phi') = \phi \vee \phi' \vee \psi$. Since we are interested in generating inductive invariants, it turns out that in order to guarantee correctness, we can use any arbitrary widening operator, but not any narrowing operator.

**Lemma 2.** *[Upward iteration sequence with widening] Let $\psi_0, \psi_1, \ldots,$ be a sequence of formulas such that $\psi_0$ is $\Theta$, and for every $i > 0$, either*
*(i) $\psi_i$ is $\mathtt{SP}(\Phi)(\psi_{i-1}) \vee \psi_{i-1}$, or*
*(ii) $\psi_i$ is $\nabla(\alpha_i)(\psi_{i-2}, \psi_{i-1})$, where $\alpha_i$ is any arbitrary formula.*

*Then, if for some $n > 0$, $\models \mathtt{SP}(\Phi)(\psi_n) \rightarrow \psi_n$, then the formula $\psi_n$ is an S-inductive invariant.*

**Lemma 3.** *[Downward iteration sequence with narrowing] Let $\phi_0, \phi_1, \ldots,$ be a sequence of formulas such that $\phi_0$ is* `true`*, and for every $i > 0$, either*
*(i) $\phi_i$ is $\mathtt{SP}(\Phi)(\phi_{i-1}) \vee \Theta$, or*
*(ii) $\phi_i$ is $\triangle(\beta_i)(\phi_{i-2}, \phi_{i-1})$, where $\beta_i$ is some S-inductive invariant.*
*Then, for every $i$, $\phi_i$ is an S-inductive invariant[4] such that $\models \phi_i \rightarrow \beta_i$.*

Lemma 3 extends the greatest fixed point iterations in Lemma 1 by a narrowing operator. Similarly, Lemma 2 extends the least fixed point computation that is hidden inside the iterations in Lemma 1 by a widening operator.

We obtain the formula $\beta_i$ used in Lemma 3 by identifying strongly connected components consisting of unreachable states. This is achieved using backward propagation from an unreachable state, as outlined in Theorem 1. These unreachable states are not automatically eliminated by the greatest fixed point computation outlined in Lemma 1. Furthermore, an S-inductive invariant obtained using Lemma 2 can be used in Step (ii) of Lemma 3. Thus, Lemma 3 gives a method for systematically strengthening known invariants.

*Example 2.* Following up on Example 1, let $N = \{1, 2, \ldots, 10\}$ denote the set of all states. In order to strengthen the over-approximation, viz. $N - \{10\}$, of the set of reachable states obtained via the greatest fixed point computation, we can try removing certain states. But if we remove a subset of states that is not strongly connected, the subsequent fixed point computation may no longer be monotonic, and could fail to converge.

For instance, removing state 5 from the above set gives a new set $N_1 = N - \{5, 10\}$. Now, $\mathtt{SP}(\Phi)(\phi_{N_1}) \vee \Theta$, where $\phi_{N_1}$ is the characteristic predicate of $N_1$, represents the set $N_2 = N - \{6, 10\}$. Clearly, $N_2 \not\subseteq N_1$, and hence the sequence of formulas obtained in the greatest fixed point computation is no longer monotonic. Note that all formulas in the sequence are invariants, but they are not inductive.

In order to identify unreachable states, we note that if we start with the set $N_3 = \{7, 8\}$, and we assign $\phi$ in Theorem 1 to the characteristic predicate $\phi_{N_3}$ of $N_3$. The least fixed point of $\mathtt{SP}(\Phi^{-1}) \vee \phi_{N_3}$ represents the set $N_4 = \{5, 6, 7, 8, 9, 10\}$. Now, since the formula $\Theta \wedge \phi_{N_4}$ is unsatisfiable (i.e. the set $\{1\} \cap N_4 = \emptyset$), it follows from Theorem 1 that the set $N_5 = \{1, 2, 3, 4\}$ represented by $\neg\phi_{N_4}$ is an S-inductive invariant.

## 4 An Any-time Algorithm for Generating Inductive Invariants

The transition predicate $\Phi$ of a transition system $\mathrm{S} = (\mathcal{V} = \{x_1, \ldots, x_n\}, \Theta, \Phi)$ is typically specified using a finite set of *guarded transitions*, where a guarded

[4] Note that the lemma also holds if we drop the word "inductive" from the statement.

transition consists of a guard $\gamma \in \mathcal{F}$, and a finite set of assignments $\{x_1 := e_1(\boldsymbol{x}), \ldots, x_n := e_n(\boldsymbol{x})\}$. A guarded transition $\tau$ is written as

$$\gamma \;\longmapsto\; x_1 := e_1(\boldsymbol{x}); \ldots; x_n := e_n(\boldsymbol{x})$$

where $e_i$ is some expression with free variables in the set $\boldsymbol{x}$. We shall also use the compact notation $\boldsymbol{x} := e(\boldsymbol{x})$ to represent the above assignments.

A typical specification of a guarded transition system contains at least one control variable, usually the program counter $pc \in \{x_1, \ldots, x_n\}$, which takes values from a finite set, say $\{1, \ldots, p\}$. Control states are defined by formulas of the form $pc = i$, $i \in \{1, \ldots, p\}$. This transition system then has $p$ different control states. Additionally, we assume that the source states of each guarded transition belong to some *fixed* source control state, $pc = i$, (and similarly for the target states) so that each transition $\tau$ can be written as

$$pc = i \wedge \gamma \;\longmapsto\; \boldsymbol{x} := e(\boldsymbol{x}); pc := j$$

where $\boldsymbol{x}$ denotes variables in $\mathcal{V} - \{pc\}$. In this case, we define $src(\tau) = i$ and $tgt(\tau) = j$. By $\Phi_\tau(\boldsymbol{x}, \boldsymbol{x'})$, we denote the formula $\gamma(\boldsymbol{x}) \wedge \boldsymbol{x'} = e(\boldsymbol{x})$. If $\mathcal{T}$ is a set of such transitions, then the transition predicate $\Phi$ is itself defined by $\bigvee_{\tau \in \mathcal{T}} pc = src(\tau) \wedge pc' = tgt(\tau) \wedge \Phi_\tau$. Similarly, we assume that $\models \Theta \rightarrow pc = 1$.

Whenever such a decomposition of the state space into finitely many control states is available such that every transition has a unique source and target control state, the S-invariant can be maintained as a conjunction of local invariants indexed by the control locations. We assume that every formula is represented as an array of formulas indexed by integers $\{1, \ldots, p\}$. Given an S-inductive invariant $\varphi$ (as an array of formulas), and a transition predicate $\Phi$, the function $\texttt{propagation}(\Theta, \Phi, \varphi, k)$ returns the strengthened S-inductive invariant $\mathcal{I}^k(\varphi)$.

$$
\begin{array}{l}
\texttt{function propagation}(\Theta, \Phi, \varphi, k)\ \{ \\
\quad \texttt{let}\ \Theta\ \texttt{be}\ pc = 1 \wedge \Theta'; \\
\quad \texttt{for}\ k\ \texttt{iterations do: for every}\ i\ \texttt{in parallel do}\ \{ \\
\qquad \mathcal{T}_i := \{\tau \in \mathcal{T} : tgt(\tau) = i\}; \\
\qquad \varphi[i] := \left\{ \begin{array}{ll} \bigvee_{\tau \in \mathcal{T}_i} \texttt{SP}(\Phi_\tau)(\varphi[src(\tau)]) \vee \Theta' & \texttt{if}\ i = 1 \\ \bigvee_{\tau \in \mathcal{T}_i} \texttt{SP}(\Phi_\tau)(\varphi[src(\tau)]) & \texttt{if}\ i \neq 1 \end{array} \right\}; \\
\qquad \varphi[i] := \Re\texttt{-simplify}(\varphi[i]); \\
\quad \} \\
\quad \texttt{return}(\varphi); \\
\}
\end{array}
$$

The function $\Re$-`simplify` performs quantifier-elimination and simplification in the theory $\Re$ and is described in Section 4.2.

**Lemma 4.** *Let $S = (\mathcal{V}, \Theta, \Phi)$ be a transition system and let $\varphi_0$ be an array of formulas initialized to* `true`*. Let $\varphi_k$ denotes the array* $\texttt{propagation}(\Theta, \Phi, \varphi_0, k)$ *of formulas (assuming $\Re$-*`simplify` *always returns equivalent formulas), and $\phi_k$ be as defined in Lemma 1. Then, for all $k \geq 0$, $\models \phi_k \leftrightarrow \bigwedge_{i=1}^{p}(pc = i \rightarrow \varphi_k[i])$. Consequently, the formula $\bigwedge_{i=1}^{p}(pc = i \rightarrow \varphi_k[i])$ is an S-inductive invariant, for every $k$.*

Notice that the formula $\bigwedge_{i=1}^{p}(pc = i \rightarrow \varphi[i])$ is equivalent to the formula $\bigvee_{i=1}^{p}(pc = i \wedge \varphi[i])$ under the assumption that $\bigvee_{i=1}^{p}(pc = i)$. The computations outlined in other lemmas and theorems can be suitably cast in terms of local invariants at control locations.

### 4.1 Combining $\mathtt{SP}(\Phi)$ and $\mathtt{SP}(\Phi^{-1})$ iterations

The basic algorithm for the automatic generation of inductive invariants consists of affirmation and propagation steps—the essence of which is captured in Lemma 1 and function `propagation`. In order to get stronger invariants, we propose the use of narrowing and widening.

The function `widening`$(\vartheta, \varphi, k)$ starts with a given under-approximation $\vartheta$ of the set of reachable states, and widens it using a subformula $\alpha$ of the over-approximation $\varphi$. If this widening yields an S-inductive invariant (see Lemma 2) in $k$ propagation steps, then the function returns this invariant, otherwise it just returns `true`[5].

```
function widening(ϑ, φ, k) {
    χ := ϑ;
    choose j ∈ {1,...,p} and a formula α s.t.
        φ[j] is of the form φ' ∨ α, and ϑ[j] ∧ α is satisfiable;
    χ[j] := χ[j] ∨ α;                           /* widening */
    χ := propagation(Θ, Φ, χ, k);
    if (⊨ propagation(Θ, Φ, χ, 1)[i] → χ[i] for all i)
        return(χ);                              /* new invariant */
    return(true);
}
```

**Lemma 5.** *For any value of the constant $k$, if $\chi$ denotes the array of formulas returned by* `widening`$(\vartheta, \varphi, k)$, *then the formula $\bigwedge_{i=1}^{p} pc = i \rightarrow \chi[i]$ is an S-inductive invariant.*

Strongly connected components of unreachable states are detected using backward propagation, and if successful, this information is used for strengthening the current invariant. The subroutine `narrowing`$(\vartheta, \varphi, k)$ chooses a subformula $\beta$ of the over-approximation $\varphi$ which could possibly represent unreachable states. Thereafter, it computes the set of states that are backward reachable from the conjectured unreachable states $\beta$ and if we successfully terminate without intersecting $\Theta$ (see Theorem 1), then we again have an S-inductive invariant.

```
function narrowing(ϑ, φ, k) {
   choose j ∈ {1,...,p} and a formula β s.t.
       φ[j] is of the form φ' ∨ β, and ϑ[j] ∧ β is unsatisfiable;
   χ := propagation(pc = j ∧ β, Φ^{-1}, false, k);
```

[5] We shall overload `true` (`false`) to also denote arrays in which every element is `true` (`false`), and use assignments between arrays to mean element-wise copying.

```
    if (⊨propagation(pc = j ∧ β, Φ⁻¹, χ, 1)[i] → χ[i] for all i)
       if (⊨ ¬(χ ∧ Θ))
          return(Invariant(¬χ));
       else if (χ ∧ Θ is satisfiable)        /* β is reachable */
          return(Reachable(pc = j ∧ β));
    return(Invariant(true));
}
```

The return value `Reachable(`$\psi$`)` of the function `narrowing(`$\vartheta, \varphi, k$`)` says that the states represented by $\psi$ are reachable, and the return value `Invariant(`$\psi$`)` denotes that the formula represented by $\psi$ is an inductive invariant.

**Lemma 6.** *For any value of the constant* $k$*, if the function* `narrowing(`$\vartheta, \varphi, k$`)` *returns* `Reachable(`$\psi$`)`*, then* $[\![\psi]\!] \subset Reach(\Phi)(\Theta)$*. Similarly, for any value of the constant* $k$*, if* `narrowing(`$\vartheta, \varphi, k$`)` *returns* `Invariant(`$\psi$`)`*, then the formula* $\bigwedge_{i=1}^{p}(pc = i \rightarrow \psi[i])$ *is an S-inductive invariant.*

Finally, we outline a procedure that uses the various functions described above by combining the least fixed point and greatest fixed point computations with narrowing and widening. In the procedure, the formula $\psi$ always stores an under-approximation of the set of reachable states, and the formula $\phi$ always stores an S-inductive invariant. The procedure essentially consists of doing one of four different steps—(i) Augmenting $\psi$ using `propagation(`$\Theta, \Phi, \psi, k$`)`, where $k$ is some constant; (ii) Strengthening the current invariant $\phi$ using the function `propagation(`$\Theta, \Phi, \phi, k$`)`; (iii) Use of widening on the under-approximation for generating an invariant; and, (iv) Use of narrowing to detect and eliminate unreachable states from the over-approximation.

```
/* Given: S = (V, Θ, Φ), a transition system with p control states.
   The transition predicate Φ is indexed by guarded transitions.
   k is an upper bound on the number of iterations. */
Procedure InvGen:
   φ, ψ: Array [1...p] of formula
   Initialization:
       φ := true;
       ψ := false;
   repeatedly do the following {{
       ψ := propagation(Θ, Φ, ψ, k);
       if (⊨ propagation(Θ, Φ, ψ, 1)[i] → ψ[i] for all i)
           φ := ψ; terminate the program;
   } OR {
       φ := propagation(Θ, Φ, φ, k);
   } OR {
       φ := φ ∧ widening(ψ, φ, k);
   } OR {
       if (narrowing(ψ, φ, k) returns Reachable(β))
           ψ[j] := ψ[j] ∨ χ where β is pc = j ∧ χ;
```

```
    else (assuming narrowing returns Invariant(β))
        φ[i] := φ[i] ∧ β[i] for all i;
}}
```

**Theorem 2.** *Let $\phi$ be the array of formulas in the procedure* `InvGen`*. Then, at any stage of the procedure, the formula $\bigwedge_i (pc = i \to \phi[i])$ is an S-inductive invariant.*

Our procedure does not consider the control structure of the transition graph to generate invariants. Though specific control structures, like loops, are not relevant for correctness of the basic procedure, they can be important in choosing specific points for widening or narrowing [6]. We wish to point out that the procedure is tolerant to theorem proving failures and only assumes a refutationally complete prover. In particular, note that the satisfiability test in `widening` can be eliminated.

### 4.2 Quantifier Elimination and Simplification

We remark here that implementation of propagation requires elimination of existential quantifiers. The existential quantifier in $\mathtt{SP}(\Phi^{-1})(\phi)$ and the universal quantifier in $\mathtt{WP}(\Phi)(\phi)$ can both be easily eliminated using substitutions. The quantifiers in $\mathtt{SP}(\Phi)(\phi)$ and $\mathtt{WP}(\Phi^{-1})(\phi)$ cannot be eliminated so easily in general. But in special cases, for instance when the transition is "reversible" (for example, the effect of assignment $x := x + y$ can be reversed by the assignment $x := x - y$), quantifier elimination reduces to substitution again. In cases where exact quantifier elimination is not possible, we can still get a correct procedure using a quantifier elimination procedure that returns a "weaker" formula, i.e., we do *not* need an equivalence preserving quantifier elimination procedure.

Let $\Re$-`simplify` be a function such that $\models \phi \to \Re\text{-}\mathtt{simplify}(\phi)$. We shall denote the formula $\Re\text{-}\mathtt{simplify}(\phi)$ by $\overline{\phi}$ in the next theorem.

**Theorem 3.** *Let $\psi_0, \psi_1, \ldots, \psi_i$ be an upward iteration sequence with widening and $\phi_0, \phi_1, \ldots, \phi_i$ be a downward iteration sequence with narrowing (see Lemmas 2 and 3). Then the sequence $\psi_0, \psi_1, \ldots, \psi_i, \overline{\psi_i}$ is also an upward iteration sequence with widening. Similarly, the sequence $\phi_0, \phi_1, \ldots, \phi_{i-1}, \phi_i'$, where $\phi_i'$ is $\phi_{i-1} \wedge \overline{\phi_i}$, is also a downward iteration sequence with narrowing.*

Note that the formula $\phi_i'$ in Theorem 3 can be seen as results of "narrowing" in the sense of [9]. Theorem 3 makes it possible for simple (and possibly incomplete) quantifier elimination procedures to suffice for our purposes. For instance, when it is not possible to eliminate the existential quantifier from $\exists x.p(x) \wedge q(x)$, we could weaken this to $\exists x.p(x) \wedge \exists x.q(x)$ and perform quantifier elimination on atomic formulas. With suitable modifications as outlined in Theorem 3, our procedure continues to be correct. In fact, such simplifications help in the convergence of the iterations as well.

Finally, as pointed out in Lemma 1, implementation of the above procedure can be optimized by combining the arrays $\psi$ and $\phi$ into a single array, say $\varphi$.

If individual formulas $\varphi[i]$ are always stored in disjunctive normal form, then we can distinguish the disjuncts that would appear in $\psi[i]$ by marking them. In this way, a single propagation step can be used to update both $\psi$ and $\phi$. The implementation of the above procedure is being done in the framework of SAL [1], which is a collection of different tools for analyzing concurrent systems.

### 4.3 Illustrative Examples

We shall provide certain simple examples to illustrate the procedure. The theory of interest is the theory of linear arithmetic, and we assume that we have an exact quantifier elimination procedure.

*Example 3.* Consider the example outlined in Section 1. In this case, the least fixed point sequence converges in two steps. In particular, we obtain the invariant $pc = inc \rightarrow x = 0 \wedge pc = dec \rightarrow x = 2$.

*Example 4.* A simplified version of the Bakery mutual exclusion protocol S $= (\mathcal{V}, \Theta, \Phi)$ for two processes $p1$ and $p2$ accessing a critical section $cs$ is given by $\mathcal{V} = \{y1 : int, y2 : int, pc1 : \{1,2,3\}, pc2 : \{1,2,3\}\}$, $\Theta$ is $pc1 = 1 \wedge pc2 = 1 \wedge y1 = 0 \wedge y2 = 0$, and $\Phi$ is defined by the following set of guarded transitions:

$$\begin{array}{rcll}
pc1 = 1 & \longmapsto & y1 := y2 + 1; pc1 := 2; & \text{// p1: try} \\
pc1 = 2 \wedge (y2 = 0 \vee y1 \leq y2) & \longmapsto & pc1 := 3; & \text{// p1: enter cs} \\
pc1 = 3 & \longmapsto & y1 := 0; pc1 := 1; & \text{// p1: exit cs} \\
pc2 = 1 & \longmapsto & y2 := y1 + 1; pc2 := 2; & \text{// p2: try} \\
pc2 = 2 \wedge (y1 = 0 \vee y2 < y1) & \longmapsto & pc2 := 3; & \text{// p2: enter cs} \\
pc2 = 3 & \longmapsto & y2 := 0; pc2 := 1; & \text{// p2: exit cs}
\end{array}$$

Since this system has an infinite number of reachable states, the least fixed point computation sequence does not converge. We choose to define 9 control locations based on the values of $pc1$ and $pc2$ variables, and we shall use the notation $\phi[i,j]$ to denote the current invariant at control location $pc1 = i \wedge pc2 = j$. After a few iterations, the greatest fixed point iterations yield a formula $\phi$, with the following three local invariants (due to space restrictions, we are not writing down the complete formula here):

$$\begin{array}{l}
\phi[3,1] : y2 = 0 \\
\phi[3,2] : (y2 = y1 + 1) \vee (y1 = 1 \wedge y2 = 0) \\
\phi[3,3] : (y1 = 0 \wedge y2 = 1) \vee (y1 = 1 \wedge y2 = 0)
\end{array}$$

The disjunct $\beta$, defined as $y1 = 0 \wedge y2 = 1$, in control location $pc1 = 3 \wedge pc2 = 3$ can be conjectured to be unreachable (as the formula $\psi[3,3]$ in the least fixed point iterations is always `false`) and for a suitable choice of $k$, the formula $\chi := \texttt{propagation}(pc1 = 3 \wedge pc2 = 3 \wedge \beta, \Phi^{-1}, \texttt{false}, k)$ contains the following strongly connected set of unreachable states,

$$\begin{array}{lll}
\chi[3,3] : y1 = 0 & \chi[3,2] : y1 = 0 & \chi[2,3] : y1 = 0 \\
\chi[3,1] : y1 = 0 & \chi[2,2] : y1 = 0 & \chi[2,1] : y1 = 0
\end{array}$$

Similarly, we can eliminate the other possibility $(y1 = 1 \wedge y2 = 0)$ at control location $pc1 = 3 \wedge pc2 = 3$. This proves mutual exclusion. We can also use a single widening step to obtain an inductive invariant strong enough to prove mutual exclusion. Note that it was pointed out in [5] that the computation of $\nu\phi.(\mathtt{WP}(\Phi)(\phi) \wedge (pc1 = 3 \wedge pc2 = 3 \rightarrow \mathtt{false}))$ terminates in a finite number of steps and yields an invariant that proves mutual exclusion.

*Example 5.* Consider the following transitions:

$$\begin{array}{lcl} pc = 1 & \longmapsto & x := x + 2; y := y + 2; pc := 2; \\ pc = 2 & \longmapsto & x := x - 2; y := y + 2; pc := 1; \end{array}$$

with initial state predicate $pc = 1 \wedge x = 0 \wedge y = 0$. Assuming that the variables $x$ and $y$ are declared to be integers, neither the least fixed point sequence, nor the greatest fixed point sequence converges. After a few iterations for computing the greatest fixed point, the formula $\phi$ we obtain is:

$$\begin{array}{l} pc = 1 \rightarrow (x = 0 \wedge y = 0) \vee (x = 0 \wedge y = 4) \vee (x \geq 0 \wedge y \geq 8) \\ pc = 2 \rightarrow (x = 2 \wedge y = 2) \vee (x = 2 \wedge y = 6) \vee (x \geq 2 \wedge y \geq 10) \end{array}$$

The predicate $\geq$ can be replaced by the predicates $=$ and $>$. Now, the disjunct $\beta$ can be chosen as $x > 0 \wedge y \geq 8$ and it can be conjectured to be unreachable. The formula $\mathtt{propagation}(pc = 1 \wedge \beta, \Phi^{-1}, \mathtt{false}, 2)$ contains the following strongly connected set of unreachable states,

$$pc = 1 \rightarrow x > 0 \wedge y \geq 8 \qquad pc = 2 \rightarrow x > 2 \wedge y \geq 6$$

Conjunction of the negation of this formula with the original invariant $\phi$ gives the following new invariant,

$$\begin{array}{l} pc = 1 \rightarrow (x = 0 \wedge y = 0) \vee (x = 0 \wedge y = 4) \vee (x = 0 \wedge y \geq 8) \\ pc = 2 \rightarrow (x = 2 \wedge y = 2) \vee (x = 2 \wedge y = 6) \vee (x = 2 \wedge y \geq 10) \end{array}$$

As before, in this case again widening can also be used to obtain a similar invariant.

## 5 Related Work and Concluding Remarks

Early work [10, 12] on generating invariant for sequential programs has been extended to the case of reactive systems in [2, 5, 11, 13, 16]. These methods are usually based on the propagation of invariants through the control structure of the different components and by combining local invariants of each component to construct global invariants of the system.

Forward and backward propagation using operators $\mathtt{SP}(\Phi)$ and $\mathtt{WP}(\Phi)$ is also used in [5] as the basic technique for generating invariants. In addition, over-approximations such as the convex hull of the union of polyhedra, are used for widening fixed point computations. Our approach differs in that we consider simultaneous forward and backward propagation for computing both lower and upper bounds of the reachable state sets. These bounds are also used for computing suitable narrowing and widening operators. The combination of these techniques usually yields much stronger invariants. Moreover, our algorithm is an any-time algorithm, in the sense that it can be interrupted at any time to yield the most refined inductive invariant computed up to the point of interruption.

The method of generalized reaffirmed invariance and propagation was introduced in [2] and is based on *affirming* local invariants of the form $\mathtt{SP}(\Phi(\tau))(\mathtt{true})$ and *propagating* these local invariants along all transitions. This process of affirmation and propagation, however, is performed only in the special case when all the existential quantifiers arising in the process are trivial, i.e., when the quantified variables do not occur in the rest of the formula; the *twos* example in the introduction does not possess this property. The technique presented in [2] also uses information about the control transition graph, especially knowledge about cycles and how variables are manipulated in the cycle transitions, to generate stronger invariants. In some cases, these stronger local invariants can be generated by repeated propagation (in the stronger sense defined in this paper). In general, however, the detection of unreachable cycles is crucial, as outlined in Theorem 1.

Techniques based on abstraction have also been proposed for generating invariants [3,14]. It appears attractive to first create (finite) abstractions for large programs and then to use standard propagation techniques to obtain the set of states reachable in the abstract system. This set can then be concretized to obtain invariants of the concrete system. Abstraction can be cast as a special widening strategy in our procedure. More specifically, let $(\alpha, \gamma)$ be an abstraction and concretization pair (Galois connection) for a transition system $\mathrm{S} = (\mathcal{V}, \Theta, \Phi)$. Let $\mathrm{S}_a = (\mathcal{V}_a, \Theta_a, \Phi_a)$ denote the abstract transition system. If

$$\psi_a^{(0)}, \psi_a^{(1)}, \psi_a^{(2)}, \ldots$$

is a least fixed point computation on the abstract transition system $\mathrm{S}_a$, then one obtains a corresponding fixed point computation with widening on the concrete system

$$\psi^{(0)}, \psi^{(1)}, \psi^{(1')}, \psi^{(2)}, \psi^{(2')}, \ldots$$

as follows: the formula $\psi^{(i)}$ is $\mathtt{SP}(\Phi)(\psi^{(i-1')}) \vee \psi^{(i-1')}$ (Step (i) of Lemma 2), and $\psi^{(i')}$ is $\psi^{(i)} \vee \gamma(\alpha(\psi^{(i)}))$ (Step (ii) of Lemma 2). Now, if $\models \gamma(\psi_a^{(i)}) \leftrightarrow \psi^{(i')}$, then it is also the case that $\models \gamma(\psi_a^{(i+1)}) \leftrightarrow \psi^{(i+1')}$. Thus, the fixed point computation on the abstract transition system can be suitably captured in the concrete system. We shall not prove this claim here, but refer to [9] for a similar result.

Note that the set of generated invariants is restricted to the ones expressible in the language of the theory $\Re$. A program that performs multiplication by repeated addition, for example, never uses the multiplication operator, but any expression that describes the set of reachable states typically would use the multiplication operator.

In summary, we present a technique for generation of inductive invariants using a combination of least and greatest fixed point computations of the forward and backward propagation operators. With obvious modifications, the results can be used to strengthen invariants. Thus, any technique for generation of invariants, inductive or not, can be incorporated with the techniques in this paper.

**Acknowledgements**. We would like to thank S. Bensalem, S. Owre, Y. Lakhnech, J. Rushby, J. Sifakis, and the referees for their helpful comments.

## References

[1] S. Bensalem, V. Ganesh, Y. Lakhnech, C. Muñoz, S. Owre, H. Rueß, J. Rushby, V. Rusu, H. Saïdi, N. Shankar, E. Singerman, and A. Tiwari. An overview of SAL. In C. M. Holloway, editor, *LFM 2000: Fifth NASA Langley Formal Methods Workshop*, pages 187–196, 2000. Available at http://shemesh.larc.nasa.gov/fm/Lfm2000/Proc/.

[2] S. Bensalem and Y. Lakhnech. Automatic generation of invariants. *Formal Methods in System Design*, 15:75–92, 1999.

[3] S. Bensalem, Y. Lakhnech, and S. Owre. Computing abstractions of infinite state systems compositionally and automatically. In *Proc. of the 9th Conference on Computer-Aided Verification, CAV'98*, LNCS. Springer Verlag, June 1998.

[4] S. Bensalem, Y. Lakhnech, and H. Saïdi. Powerful techniques for the automatic generation of invariants. In R. Alur and T. A. Henzinger, editors, *Computer-Aided Verification, CAV '96*, number 1102 in LNCS, pages 323–335. Springer-Verlag, 1996.

[5] N. Bjørner, A. Browne, and Z. Manna. Automatic Generation of Invariants and Intermediate Assertions. *Theoretical Computer Science*, 1997.

[6] F. Bourdoncle. Efficient chaotic iteration strategies with widenings. In *Proceedings of the Intl Conf on Formal Methods in Programming and their Applications*, volume 735 of *LNCS*, pages 128–141. Springer Verlag, 1993.

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

[8] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th POPL*, January 1977.

[9] P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In M. Bruynooghe and M. Wirsing, editors, *Proc. of the 4th Intl. Symposium on Programming Language Implementation and Logic Programming* (*PLILP '92*), volume 631 of *LNCS*, pages 269–295, Berlin, 1992. Springer-Verlag.

[10] S. M. German and B. Wegbreit. A synthesizer of inductive assertions. *IEEE Transactions on Software Engineering*, 1(1):68–75, March 1975.

[11] S. Graf and H. Saïdi. Verifying invariants using theorem proving. In *Conference on Computer Aided Verification CAV'96*, LNCS 1102, Springer Verlag, 1996.

[12] S. Katz and Z. Manna. Logical analysis of programs. *Communications of the ACM*, 19(4):188–206, April 1976.

[13] L. Lamport. The 'Hoare logic' of concurrent programs. In *Acta Informatica 14*, pages 21–37, 1980.

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

[15] Z. Manna and A. Pnueli. *The Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, 1995.

[16] S. Owicki and D. Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, 6:319–340, 1976.

[17] H. Saïdi and N. Shankar. Abstract and model check while you prove. In *Computer-Aided Verification, CAV '99*, Trento, Italy, July 1999.