In U. Montanari (ed.), *First International Conference on Principles and Practice of Constraint Programming*, LNCS, Cassis, France, September 1995

Automatic Generation of Invariants and Intermediate Assertions *

Nikolaj Bjørner, Anca Browne and Zohar Manna

Computer Science Department
Stanford University
Stanford, CA 94305

Abstract. Verifying temporal specifications of reactive and concurrent systems commonly relies on generating auxiliary assertions and strengthening given properties of the system. Two dual approaches find solutions to these problems: the *bottom-up* method performs an abstract forward propagation of the system, generating auxiliary assertions; the *top-down* method performs an abstract backward propagation to strengthen given properties. *Exact* application of these methods is complete but is usually infeasible for large-scale verification. An *approximate analysis* can often supply enough information to complete the verification.
The paper overviews some of the exact and approximate analysis methods to generate and strengthen assertions for the verification of invariance properties. By formulating and analyzing a generic safety verification rule, we extend these methods to the verification of general temporal safety properties.

Contents

1. Introduction
2. Preliminaries
3. Invariance
4. General Safety
5. Approximate Analysis
6. Systems of Linear Equations
7. Convex Polyhedra
8. Set-based Analysis

1 Introduction

The deductive verification of reactive and concurrent systems commonly relies upon finding suitable auxiliary assertions and strengthened properties to complete the proof [MP95]. This paper describes a number of systematic methods for generating invariants and intermediate assertions that can help in the verification of temporal safety properties.

An invariant of a system $\mathcal{S}$ is a first-order formula (an *assertion*) that holds at every state of its computations. Deductive methods provide a verification rule

* This research was supported in part by the National Science Foundation under grant CCR-92-23226, the Advanced Research Projects Agency under NASA grant NAG2-892, the United States Air Force Office of Scientific Research under grant F49620-93-1-0139, and the Department of the Army under grant DAAH04-95-1-0317.

that prove invariants by establishing first-order premises. These premises require an auxiliary assertion which strengthens the invariant being proved. We will examine two ways of finding the strengthening assertion: forward propagation (bottom-up method) is a symbolic forward execution of $\mathcal{S}$ that summarizes the set of reachable states as an assertion; backward propagation (top-down method) is a symbolic backward execution of $\mathcal{S}$ from the states satisfying the invariant being proved and also generates an assertion. We will see that both methods, in principle, generate adequate candidates for the auxiliary assertion. Early work on forward and backward propagation in program verification includes [GW75, KM76].

We formulate a verification rule applicable to arbitrary temporal safety formulas, and show how forward and backward propagation can be generalized to all safety formulas. General temporal safety formulas can express properties such as first-in-first-out ordering, causality and bounded overtaking.

Exact forward or backward propagation of a system $\mathcal{S}$ may not terminate when the state space of $\mathcal{S}$ is infinite or unmanageably large. This motivates the study of approximate, but decidable, propagation methods over different domains. We overview techniques from linear algebra, linear programming and monadic second-order logic in a general framework. Table 1 categorizes the methods discussed in this paper. Linear algebra has long been applied to the automatic discovery of linear equalities between system variables [Lau73, Kar76]. Linear programming was proposed in [CH78] to express linear constraints among system variables in the form of convex polyhedra. It is also one of the most prominent examples of the abstract interpretation theory introduced in [CC77].[2] Monadic second-order logic applied in set-based program analysis [Hei92] provides unary constraints on the values of program variables.

Table 1. Approximation methods.

Domain	Constraint form	Methods	Propagation
Linear equations	$\mathbf{A}\overline{x} = \mathbf{b}$	Linear algebra	Finite
Polyhedra	$\mathbf{A}\overline{x} \leq \mathbf{b}$	Linear programming	Approximate
Set-based	$x \in S, y \in T$	Tree automata	Implicit

Section 2 summarizes preliminary notions such as the computational model of reactive systems, linear-time temporal logic and fixedpoints. In Section 3 we present the verification rule for invariance and introduce bottom-up and top-down methods. The methods are extended to handle general safety properties in Section 4. Section 5 introduces approximation as a tool to enable the automatic analysis presented in Sections 3 and 4. Finally, the approximate propagation methods are described in Sections 6, 7 and 8.

The methods described in this paper have proved useful as part of the verification system STeP, the *Stanford Temporal Prover* [MAB+94]. STeP uses linear-time temporal logic in the specification of reactive and concurrent systems,

[2] Abstract interpretation is also being applied in connection with model checking [CGL92, DGG94].

and combines model checking with deductive techniques. The tools presented are auxiliary routines that can be invoked from STeP's deductive environment.

2 Preliminaries

2.1 Transition Systems

Following [MP95], our computational model for reactive systems is that of a *transition system*, $\mathcal{S} = \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$[3], where $\mathcal{V}$ is a finite set of system variables, Θ is the initial condition, and $\mathcal{T}$ is a finite set of transitions. The vocabulary $\mathcal{V}$ contains data variables, control variables and auxiliary variables. The set of *states* (interpretations) over $\mathcal{V}$ is denoted by Σ. We assume a first-order assertion language $\mathcal{A}$ over $\mathcal{V}$. The initial condition Θ is an assertion in this language. A transition τ maps each state $s \in \Sigma$ into a (possibly empty) set of τ-successors, $\tau(s) \subseteq \Sigma$. It is defined by an assertion $\rho_\tau(\overline{x}, \overline{x}')$, called the *transition relation*, which relates the values $\overline{x}$ of the variables in state s and the values $\overline{x}'$ in a successor state $s' \in \tau(s)$. We require that $\mathcal{T}$ contain a transition τ_I, called the *idling transition*, such that $\tau(s) = \{s\}$ for every state s.

A computation of a system $\mathcal{S}$ is an infinite sequence of states $s_0, s_1, s_2, \ldots$, such that s_0 is an initial state satisfying Θ and for every $i \geq 0$ there is a transition $\tau \in \mathcal{T}$ satisfying $s_{i+1} \in \tau(s_i)$.

To facilitate the representation of systems, some of our examples are given in SPL (Simple Programming Language). SPL statements are translated into transitions in a straightforward manner. For example, the assignment statement

$$\ell_0 : x := y + 1;\ \ell_1 :$$

assigns $y + 1$ to x when control resides at location ℓ_0, and subsequently moves control to ℓ_1; it generates the transition τ with transition relation

$$\rho_\tau(\ell_0, \ell_1, x, y, \ell_0', \ell_1', x', y') :\ \ \ell_0 \ \wedge\ x' = y + 1 \ \wedge\ y' = y \ \wedge\ \ell_1' \ \wedge\ \neg \ell_0' \ .$$

2.2 Preconditions and Postconditions

The weakest precondition $\text{WP}(\tau, \varphi)(\overline{x})$ and strongest postcondition $\text{SP}(\tau, \varphi)(\overline{x})$ of an assertion $\varphi(\overline{x})$, relative to a transition τ, are defined by

$$\begin{array}{lcl} \text{WP}(\tau, \varphi)(\overline{x}) & \stackrel{\triangle}{=} & \forall \overline{x}' \cdot \rho_\tau(\overline{x}, \overline{x}') \rightarrow \varphi(\overline{x}') \\ \text{SP}(\tau, \varphi)(\overline{x}) & \stackrel{\triangle}{=} & \exists \overline{x}_0 \cdot \rho_\tau(\overline{x}_0, \overline{x}) \wedge \varphi(\overline{x}_0) \ . \end{array}$$

WP characterizes the states that must reach a φ-state (i.e., a state satisfying φ) by taking τ. SP characterizes the states reachable from a φ-state by taking τ. For example, transitions associated with guarded assignments of the form **if** $c(\overline{x})$ **then** $\overline{x} := e(\overline{x})$ have transition relations of the form $\rho_\tau : c(\overline{x}) \wedge \overline{x}' = e(\overline{x})$.

[3] Fairness assumptions are ignored when establishing safety properties.

Their weakest precondition, $\text{WP}(\tau,\varphi)(\overline{x}) : \forall \overline{x}'.c(\overline{x}) \ \wedge \ \overline{x}' = e(\overline{x}) \rightarrow \varphi(\overline{x}')$, can be simplified to $\text{WP}(\tau,\varphi)(\overline{x}) : \ c(\overline{x}) \ \rightarrow \ \varphi(e(\overline{x}))$.

The two operators are monotone in φ since all occurrences of φ are positive, i.e., they are under the scope of an even number of negations. Thus,

$$(\varphi \rightarrow \psi) \ \rightarrow \ (\text{WP}(\tau,\varphi) \rightarrow \text{WP}(\tau,\psi)), \qquad (\varphi \rightarrow \psi) \ \rightarrow \ (\text{SP}(\tau,\varphi) \rightarrow \text{SP}(\tau,\psi)).$$

Let τ^{-1} be the inverse of τ, i.e., $\rho_{\tau^{-1}}(\overline{x},\overline{y}) \stackrel{\triangle}{=} \rho_\tau(\overline{y},\overline{x})$. Then we observe

$$\begin{array}{ll} \text{WP}(\tau,\neg\varphi) \ \leftrightarrow \neg\text{SP}(\tau^{-1},\varphi) & \text{SP}(\tau,\neg\varphi) \ \leftrightarrow \neg\text{WP}(\tau^{-1},\varphi) \\ \text{WP}(\tau,\varphi\wedge\psi) \leftrightarrow \text{WP}(\tau,\varphi)\wedge\text{WP}(\tau,\psi) & \text{SP}(\tau,\varphi\vee\psi) \leftrightarrow \text{SP}(\tau,\varphi)\vee\text{SP}(\tau,\psi) \\ \text{WP}(\tau,\varphi\vee\psi) \leftarrow \text{WP}(\tau,\varphi)\vee\text{WP}(\tau,\psi) & \text{SP}(\tau,\varphi\wedge\psi) \rightarrow \text{SP}(\tau,\varphi)\wedge\text{SP}(\tau,\psi). \end{array}$$

We will also use the notation

$$\text{WP}(\mathcal{T},\varphi) \ \stackrel{\triangle}{=} \ \bigwedge_{\tau\in\mathcal{T}} \text{WP}(\tau,\varphi) \qquad \text{SP}(\mathcal{T},\varphi) \ \stackrel{\triangle}{=} \ \bigvee_{\tau\in\mathcal{T}} \text{SP}(\tau,\varphi) \ .$$

Notice that

$$\varphi \ \rightarrow \text{WP}(\mathcal{T},\psi) \quad \text{iff} \quad \text{SP}(\mathcal{T},\varphi) \ \rightarrow \ \psi \ .$$

2.3 Linear-Time Temporal Logic

We specify properties of reactive systems using linear-time temporal logic. A *temporal formula* is constructed from state formulas (called *assertions*), which are formulas from the first-order assertion language $\mathcal{A}$. To state formulas we apply boolean connectives (such as $\vee$, $\neg$), quantifiers ($\forall$, $\exists$) and temporal operators. The temporal operators used in this paper are future operators $\Box$ (*always in the future*), W (*waiting-for, unless*), $\bigcirc$ (*next*) and their past counterparts $\boxminus$ (*always in the past*), B (*back-to*) and $\ominus$ (*previously*).

A *model* for a temporal formula φ is an infinite sequence of states $\sigma : s_0, s_1, s_2, \ldots$, where each state s_j provides an interpretation for the variables occurring in φ. Given a model σ, we present an inductive definition of the notion of φ holding at position j, $j \geq 0$, in σ, denoted by $\langle\sigma, j\rangle \vDash \varphi$:

$$\begin{array}{lll} \langle\sigma,j\rangle \vDash \varphi & \Longleftrightarrow \ s_j \vDash \varphi & \text{if } \varphi \in \mathcal{A} \\ & \multicolumn{2}{l}{\text{That is, } \varphi \text{ is evaluated locally, using the interpretation in } s_j} \\ \langle\sigma,j\rangle \vDash \bigcirc\varphi & \Longleftrightarrow \ \langle\sigma,j+1\rangle \vDash \varphi & \\ \langle\sigma,j\rangle \vDash \Box\varphi & \Longleftrightarrow \ \forall j' : j' \geq j \cdot \langle\sigma,j'\rangle \vDash \varphi & \\ \langle\sigma,j\rangle \vDash \varphi \ W \ \psi & \Longleftrightarrow \ \forall j' : j' \geq j \cdot \langle\sigma,j'\rangle \vDash \varphi \quad \text{or} & \\ & \quad \exists k : k \geq j \cdot \langle\sigma,k\rangle \vDash \psi \text{ and } \forall j' : j \leq j' < k \cdot \langle\sigma,j'\rangle \vDash \varphi & \\ \langle\sigma,j\rangle \vDash \ominus\varphi & \Longleftrightarrow \ j > 0 \quad \text{and} \quad \langle\sigma,j-1\rangle \vDash \varphi & \\ \langle\sigma,j\rangle \vDash \boxminus\varphi & \Longleftrightarrow \ \forall j' : 0 \leq j' \leq j \cdot \langle\sigma,j'\rangle \vDash \varphi & \\ \langle\sigma,j\rangle \vDash \varphi \ B \ \psi & \Longleftrightarrow \ \forall j' : 0 \leq j' \leq j \cdot \langle\sigma,j'\rangle \vDash \varphi \quad \text{or} & \\ & \quad \exists k : 0 \leq k \leq j \cdot \langle\sigma,k\rangle \vDash \psi \text{ and } \forall j' : j \geq j' > k \cdot \langle\sigma,j'\rangle \vDash \varphi & \end{array}$$

Other temporal connectives are defined as abbreviations, e.g., $\Diamond\varphi = \neg\,\Box\neg\varphi$, $\Diamond\!\!-\varphi = \neg\,\boxminus\neg\varphi$.

A temporal formula φ is $\mathcal{S}$-*valid*, denoted $\mathcal{S} \models \varphi$, if for each computation σ of $\mathcal{S}$, $\langle \sigma, 0 \rangle \models \varphi$. A state s is said to be an $\mathcal{S}$-*reachable* state if it can be reached by some computation of $\mathcal{S}$. A state formula φ is $\mathcal{S}$-*state-valid* if $s \models \varphi$ for every $\mathcal{S}$-reachable state s of $\mathcal{S}$. See [MP95] for a more detailed discussion of linear-time temporal logic and reactive systems.

2.4 Fixedpoints

For a monotone operator $G : \mathcal{A} \mapsto \mathcal{A}$, such as WP and SP, we write the least fixedpoint as $\mu X.G(X)$ and the greatest fixedpoint of G as $\nu X.G(X)$. To give meaning to the fixedpoint operators μ and ν we introduce *denotations* $[\![\varphi]\!]$ of assertions φ in $\mathcal{A}$: $[\![\varphi]\!] = \{s \in \Sigma \mid s \models \varphi\}$. In particular $[\![G(X)]\!]_{[S/X]}$ denotes $G(X)$, where $[\![X]\!] = S$. The denotations of the fixedpoint operators are now defined as

$$\begin{array}{lcl} [\![\mu X.G(X)]\!] & \triangleq & \bigcap \{S \subseteq \Sigma \mid [\![G(X)]\!]_{[S/X]} \subseteq S\} \\ [\![\nu X.G(X)]\!] & \triangleq & \bigcup \{S \subseteq \Sigma \mid S \subseteq [\![G(X)]\!]_{[S/X]}\} \quad . \end{array}$$

The monotonicity of G ensures that $[\![\mu X.G(X)]\!]$ is the unique least fixedpoint, i.e., the least set $S \subseteq \Sigma$ such that $S = [\![G(X)]\!]_{[S/X]}$, and that $[\![\nu X.G(X)]\!]$ is the unique greatest fixedpoint.

Searching for fixedpoints in the assertion language $\mathcal{A}$ itself, we apply G repeatedly to F (false) and generate the ascending chain $\mathrm{F}, G(\mathrm{F}), G^{(2)}(\mathrm{F}), \ldots$, which, when it converges in a finite number of iterations, produces an assertion in $\mathcal{A}$, which is equivalent to $\mu X.G(X)$. Similarly, generating the descending chain $\mathrm{T}, G(\mathrm{T}), G^{(2)}(\mathrm{T}), \ldots$, starting from T (true), may yield an assertion in $\mathcal{A}$, which is equivalent to $\nu X.G(X)$. In general, the generated sequences are not guaranteed to converge to the fixedpoints in a finite number of steps.

3 Invariance

An assertion p is $\mathcal{S}$-*invariant* (*invariant* for short) if $\mathcal{S} \models \Box p$. To establish that a given assertion p is $\mathcal{S}$-invariant, we use the verification rule INV (Figure 1). This sound and relatively complete proof rule reduces the verification of $\Box p$ to first-order premises. For a given transition system $\mathcal{S}$ and assertion p, to prove that $\Box p$ is $\mathcal{S}$-valid, we have to find a strengthened assertion φ such that the first-order premises $I1$–$I3$ are $\mathcal{S}$-state-valid.

In premise $I3$ we use the notation

$$\{\varphi\}\ \mathcal{T}\ \{\psi\} = \bigwedge_{\tau \in \mathcal{T}} \{\varphi\}\tau\{\psi\}$$

where each *verification condition* $\{\varphi\}\tau\{\psi\}$ can be expressed equivalently as

$$\begin{array}{ll} \text{standard:} & \forall \overline{x} \forall \overline{x}' \cdot [\varphi(\overline{x}) \ \wedge \ \rho_\tau(\overline{x}, \overline{x}') \rightarrow \psi(\overline{x}')] \\ \text{weakest precondition:} & \forall \overline{x} \cdot [\varphi(\overline{x}) \ \rightarrow \ \mathrm{WP}(\tau, \psi)(\overline{x})] \\ \text{strongest postcondition:} & \forall \overline{x}' \cdot [\mathrm{SP}(\tau, \varphi)(\overline{x}') \ \rightarrow \ \psi(\overline{x}')] \quad . \end{array}$$

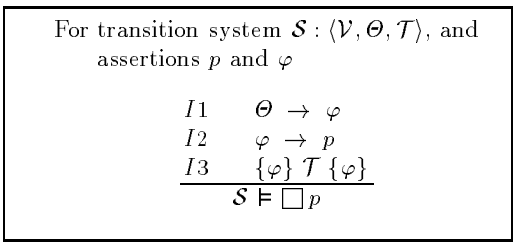


Fig. 1. Rule INV (invariance)

When establishing the first-order premises $I1-I3$ of rule INV it is sound to assert any previously established invariant ψ as an axiom. This will be used throughout the paper.

The main difficulty in using rule INV is finding the strengthened assertion φ. We now show that the strongest and weakest candidates for φ can be given fixedpoint characterizations.

3.1 Forward propagation

Define the operator $\mathcal{F}$ by

$$\mathcal{F}(X) \quad \stackrel{\triangle}{=} \quad \Theta \vee \mathrm{SP}(\mathcal{T}, X) \ .$$

A formula φ satisfies $I1$ and $I3$ iff $\mathcal{F}(\varphi) \rightarrow \varphi$. Since $\mathcal{F}$ is monotone, i.e., if $X_1 \rightarrow X_2$ then $\mathcal{F}(X_1) \rightarrow \mathcal{F}(X_2)$, the fixedpoint formula

$$\varphi_{\mathcal{F}} : \quad \mu X.\mathcal{F}(X)$$

provides the strongest assertion φ satisfying $I1$ and $I3$. We have therefore

$$(\mathrm{F1}) \quad \Theta \rightarrow \varphi_{\mathcal{F}} \qquad\qquad (\mathrm{F2}) \quad \mathrm{SP}(\mathcal{T}, \varphi_{\mathcal{F}}) \rightarrow \varphi_{\mathcal{F}} \ .$$

Notice that $\varphi_{\mathcal{F}}$ precisely characterizes the set of reachable states:

$$\text{state } s \text{ is } \mathcal{S}\text{-accessible} \quad \text{iff} \quad s \models \varphi_{\mathcal{F}} \ .$$

The implication from left to right holds for any formula satisfying $I1$ and $I3$; the converse holds because $\varphi_{\mathcal{F}}$ is the strongest such formula.

As $\mathcal{F}$ is monotone, if the sequence starting from F (false)

$$\underbrace{\mathrm{F}}_{\varphi_0} \rightarrow \underbrace{\mathcal{F}(\varphi_0)}_{\varphi_1} \rightarrow \underbrace{\mathcal{F}(\varphi_1)}_{\varphi_2} \rightarrow \cdots$$

converges in finitely many steps, i.e., $\varphi_n \leftrightarrow \varphi_{n+1}$ is $\mathcal{S}$-valid for some n, then its limit is $\varphi_{\mathcal{F}}$.

3.2 Backward propagation

Given an assertion p, define the monotone operator $\mathcal{B}$ by

$$\mathcal{B}(Y) \quad \stackrel{\triangle}{=} \quad p \wedge \mathrm{WP}(\mathcal{T}, Y) \ .$$

A formula satisfies satisfies $I2$ and $I3$ iff $\varphi \rightarrow \mathcal{B}(\varphi)$. The greatest fixedpoint

$$\varphi_{\mathcal{B}} : \quad \nu Y.\mathcal{B}(Y)$$

provides the weakest φ satisfying $I2$ and $I3$. We therefore have

$$\text{(B1)} \quad \varphi_{\mathcal{B}} \rightarrow p \qquad\qquad \text{(B2)} \quad \varphi_{\mathcal{B}} \rightarrow \mathrm{WP}(\mathcal{T}, \varphi_{\mathcal{B}}) \ .$$

$\varphi_{\mathcal{B}}$ precisely characterizes the *p-invariant states* i.e., the states where every computation of $\mathcal{S}$ starting in s satisfies $\Box p$.

Since $\mathcal{B}$ is monotone, if the sequence starting from T (true)

$$\underbrace{\mathrm{T}}_{\varphi_0} \quad \leftarrow \quad \underbrace{\mathcal{B}(\varphi_0)}_{\varphi_1} \quad \leftarrow \quad \underbrace{\mathcal{B}(\varphi_1)}_{\varphi_2} \quad \leftarrow \quad \cdots$$

converges in finitely many steps, then its limit is $\varphi_{\mathcal{B}}$.

3.3 The forward/backward duality

Abstract state space exploration from the set of initial states, also called forward propagation (*bottom-up analysis*), does not depend on the system property we want to establish. Backward propagation (*Top-down analysis*), on the other hand, explores the state space starting with the states that satisfy an invariant candidate p.

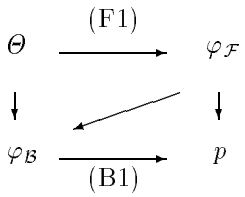
Since $\varphi_{\mathcal{F}}$ captures exactly the reachable state space, whereas $\varphi_{\mathcal{B}}$ only collects enough information to establish p, one can make the following observation:

Property 1 *The following are equivalent: 1:* $\Theta \rightarrow \varphi_{\mathcal{B}}$*, 2:* $\varphi_{\mathcal{F}} \rightarrow p$*, 3:* $\varphi_{\mathcal{F}} \rightarrow \varphi_{\mathcal{B}}$*.*

Proof Assume $\Theta \rightarrow \varphi_{\mathcal{B}}$. By the fixedpoint characterization (B2) of $\varphi_{\mathcal{B}}$ and by the equivalence $(X \rightarrow \mathrm{WP}(\mathcal{T}, Y)) \ \leftrightarrow \ (\mathrm{SP}(\mathcal{T}, X) \rightarrow Y)$, we may infer $\Theta \vee \mathrm{SP}(\mathcal{T}, \varphi_{\mathcal{B}}) \rightarrow \varphi_{\mathcal{B}}$. Since $\mathcal{F}(\varphi_{\mathcal{B}}) = \Theta \ \vee \ \mathrm{SP}(\mathcal{T}, \varphi_{\mathcal{B}})$ and $\varphi_{\mathcal{F}}$ is the strongest assertion satisfying $\mathcal{F}(\varphi) \rightarrow \varphi$ we have $\varphi_{\mathcal{F}} \rightarrow \varphi_{\mathcal{B}}$. This establishes *1⇒3*. The argument establishing *2⇒3* is analogous, using (F2).

To establish *3⇒1*, assume $\varphi_{\mathcal{F}} \rightarrow \varphi_{\mathcal{B}}$. By the fixedpoint characterization of $\varphi_{\mathcal{F}}$ we have (F1) and *3⇒1* follows by transitivity of implication. The implication *3⇒2* is established in a similar way. ∎

The correspondence is best illustrated by the commuting diagram:



where the horizontal implications are given by the fixedpoint equations. If one of the downwards directed implications is present, all the others must also be.

The diagram suggests that if p is an invariant, then the $\varphi_{\mathcal{F}}$ states are a subset of the $\varphi_{\mathcal{B}}$ states as reflected in Figure 2.

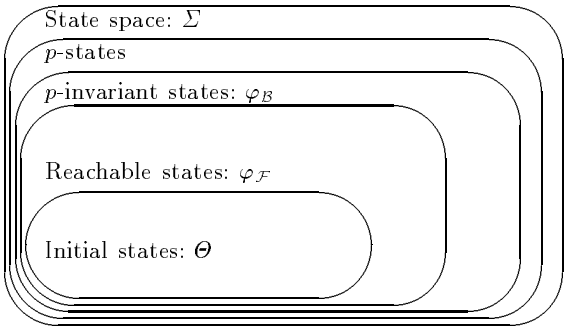


Fig. 2. State space.

As Figure 2 shows, the p-invariant states, given by $\varphi_{\mathcal{B}}$, are a subset of the p-states, which are those states where p holds but is not necessarily preserved by the transitions.

Example (forward and backward propagation) We analyze program BAKERY (Figure 3) using both forward and backward propagation. The program guarantees mutual exclusion, that is, ℓ_3 and m_3 are never reached at the same time. Synchronization is provided by the integer variables y_1 and y_2, which can be thought of as numbers used in waiting-lines at bakeries.

local y_1, y_2 : **integer where** $y_1 = y_2 = 0$

$$
P_1 :: \left[\begin{array}{l}\textbf{loop forever do} \\ \left[\begin{array}{ll} \ell_0\text{:} & \textbf{noncritical} \\ \ell_1\text{:} & y_1 := y_2 + 1 \\ \ell_2\text{:} & \textbf{await } (y_2 = 0 \vee y_1 \leq y_2) \\ \ell_3\text{:} & \textbf{critical} \\ \ell_4\text{:} & y_1 := 0 \end{array}\right]\end{array}\right] \;\|\; P_2 :: \left[\begin{array}{l}\textbf{loop forever do} \\ \left[\begin{array}{ll} m_0 : & \textbf{noncritical} \\ m_1 : & y_2 := y_1 + 1 \\ m_2 : & \textbf{await } (y_1 = 0 \vee y_2 < y_1) \\ m_3 : & \textbf{critical} \\ m_4 : & y_2 := 0 \end{array}\right]\end{array}\right]
$$

Fig. 3. Program BAKERY (Program BAKERY for mutual exclusion)

Forward propagation The method requires computing the sequence

$$\underbrace{\mathrm{F}}_{\varphi_0} \rightarrow \underbrace{\mathcal{F}(\varphi_0)}_{\varphi_1} \rightarrow \underbrace{\mathcal{F}(\varphi_1)}_{\varphi_2} \rightarrow \cdots$$

until a limit is found.

Figure 4 represents the iterations of $\mathcal{F}$ as layers in a directed graph, growing bottom-up from the initial condition $\varphi_1 : \Theta$. The i-th iteration of $\mathcal{F}$, represented by φ_i, is the disjunction of the nodes that are reachable from the source on a path of at most depth $i-1$.

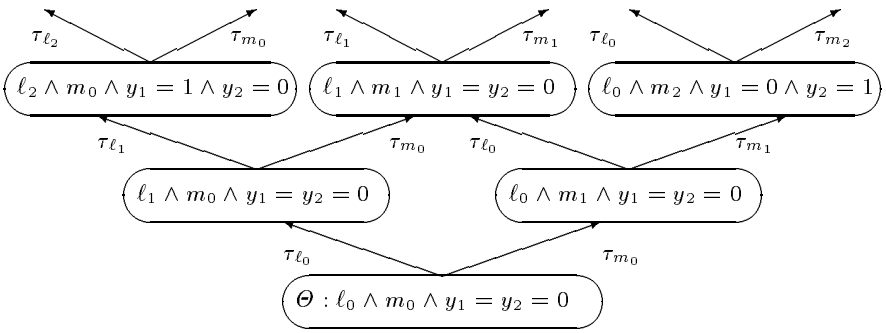


Fig. 4. Forward propagation

From the initial condition Θ we calculate

$$\begin{aligned} \mathrm{SP}(\mathcal{T}, \Theta) &= \textstyle\bigvee_{\tau \in \mathcal{T}} \mathrm{SP}(\tau, \Theta) \\ &= \mathrm{SP}(\tau_{\ell_0}, \Theta) \vee \mathrm{SP}(\tau_{m_0}, \Theta) \\ &= (\ell_1 \ \wedge \ m_0 \ \wedge \ y_1 = y_2 = 0) \quad \vee \quad (\ell_0 \ \wedge \ m_1 \ \wedge \ y_1 = y_2 = 0) \end{aligned}$$

which generates the two disjuncts just above Θ. One more iteration of $\mathcal{F}$ generates the three disjuncts at the third level. The sequence $\varphi_1, \varphi_2, \varphi_3, \ldots$ does not converge to a fixedpoint in a finite number of steps, since $\varphi_{5k} \wedge \ell_2 \wedge m_2$ is equivalent to

$$\ell_2 \ \wedge \ m_2 \quad \wedge \ |y_1 - y_2| = 1 \ \wedge \ 1 \leq y_1 \leq k+1 \ \wedge \ 1 \leq y_2 \leq k+1 \ .$$

However, it is possible to express the strongest invariant (representing the reachable state space) of the program by the assertion:

$$\begin{aligned} \varphi_{\mathcal{F}} : \ & \ell_{0,1} \wedge (m_{0,1} \wedge y_1 = 0 \wedge y_2 = 0 \ \vee \ m_{2,3,4} \wedge y_1 = 0 \wedge y_2 \geq 1) \\ & \vee \ \ell_{2,3,4} \wedge m_{0,1} \wedge y_1 \geq 1 \wedge y_2 = 0 \\ & \vee \ \ell_2 \wedge (m_2 \wedge y_1 \geq 1 \wedge y_2 \geq 1 \wedge |y_1 - y_2| = 1 \ \vee \ m_{3,4} \wedge y_2 \geq 1 \wedge y_1 - y_2 = 1) \\ & \vee \ \ell_{3,4} \wedge m_2 \wedge y_1 \geq 1 \wedge y_2 - y_1 = 1 \ . \end{aligned}$$

Backward propagation Backward propagation starts from an invariant candidate, in this case

$$\Box \neg (\ell_3 \ \wedge \ m_3),$$

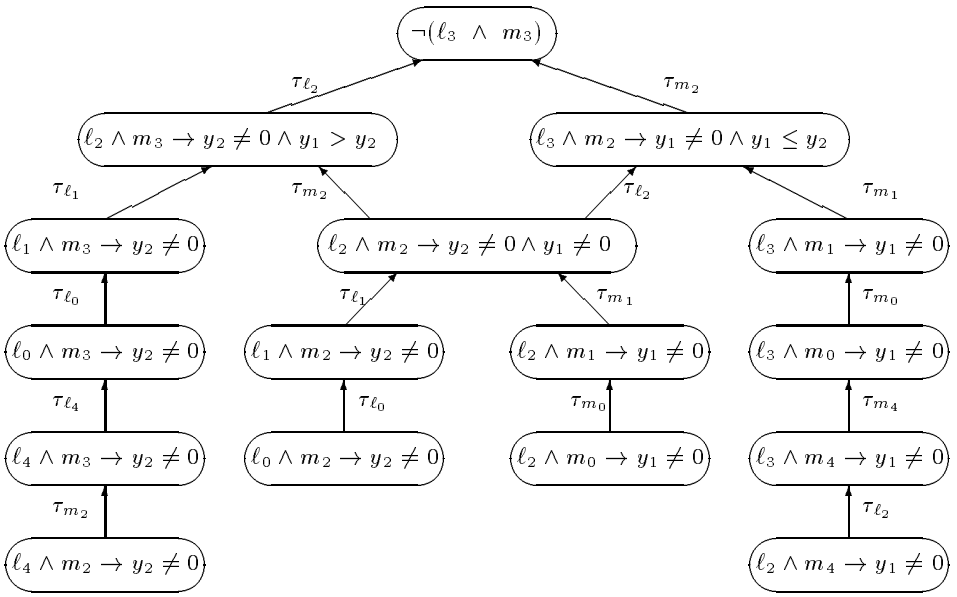


Fig. 5. Backward propagation from $\neg(\ell_3 \ \wedge \ m_3)$

which expresses mutual exclusion in the critical sections.

We compute the terms of the sequence

$$\underbrace{\mathrm{T}}_{\varphi_0} \leftarrow \underbrace{\mathcal{B}(\varphi_0)}_{\varphi_1} \leftarrow \underbrace{\mathcal{B}(\varphi_1)}_{\varphi_2} \leftarrow \cdots$$

until a limit is found. Applying $\mathcal{B}$ once generates $\varphi_1 : \neg(\ell_3 \ \wedge \ m_3)$. In the second iteration of $\mathcal{B}$ we calculate:

$$\begin{aligned} \mathrm{WP}(\mathcal{T}, \varphi_1) &= \textstyle\bigwedge_{\tau \in \mathcal{T}} \mathrm{WP}(\tau, \varphi_1) \\ &= \mathrm{WP}(\tau_{\ell_2}, \varphi_1) \ \wedge \ \mathrm{WP}(\tau_{m_2}, \varphi_1) \\ &= (\ell_2 \wedge m_3 \to y_2 \neq 0 \wedge y_1 > y_2) \wedge (\ell_3 \wedge m_2 \to y_1 \neq 0 \wedge y_1 \leq y_2). \end{aligned}$$

Continuing mechanically in this fashion we obtain the formulas shown in Figure 5. The conjunction of the formulas is the greatest fixedpoint $\varphi_{\mathcal{B}}$ of $\mathcal{B}$. By calculating $\mathrm{WP}(\tau, \varphi)$, where τ labels an edge pointing to a node labeled by φ, one obtains the assertion labeling the source of the edge. The auxiliary invariants $\Box(y_1 \geq 0)$ and $\Box(y_2 \geq 0)$ were used to simplify the examined conjuncts. These invariants can be generated automatically by several of the methods described later in this paper. Finally, since $\Theta : \ell_0 \wedge m_0 \wedge y_1 = y_2 = 0$ implies $\varphi_{\mathcal{B}}$, we have indeed established mutual exclusion of the critical sections.

The example shows the power of backward propagation: a completely automatic search terminates with a strengthened invariant. The analysis described

above is entirely automatic in STeP. Forward propagation, on the other hand, does not converge in finitely many steps because the program's set of states cannot be reached in finitely many iterations. ▟

4 General Safety

A *general safety* formula p is one that is equivalent to a temporal formula of the form

$$\Box\, q$$

where q is a past formula, i.e., it does not contain any future temporal operators.

Example (general safety formulas) The state-causality formula

$$\Box \underbrace{[p \;\rightarrow\; \diamondminus\, r]}_{\text{past}}$$

for assertions p and r, is a general safety formula. A model satisfies this formula if every p-position coincides with or is preceded by an r-position.

The nested waiting-for formula

$$\Box[p \;\rightarrow\; q_1 \;W\; (q_2 \;W\; r)]$$

for assertions p, q_1, q_2 and r is a general safety formula. A model satisfies this formula if every p-position initiates a q_1-interval, followed by a q_2-interval, that may be terminated by an r-position. Each q_i-interval, $i = 1, 2$, is a set of successive positions all of which satisfy q_i, and may be empty or extend to infinity. In the latter case, there need not be a following interval nor a terminating r-position. This is a general safety formula, since it is equivalent to the formula

$$\Box \underbrace{[(\neg q_2) \;\rightarrow\; q_1 \;B\; ((\neg p) \;B\; r)]}_{\text{past}} \;. \;\; ▟$$

An *assertion graph* $G = \langle N, N_0, E, \lambda \rangle$ is a labeled directed graph where

N	–	set of nodes
$N_0 \subseteq N$	–	initial nodes
$E \subseteq N \times N$	–	edges
$\lambda : N \mapsto \mathcal{A}$	–	mapping from nodes to assertions.

G is an *assertion graph for* p if it describes exactly the models of p, i.e., for every model $\sigma : s_0, s_1, \ldots$, $\langle \sigma, 0 \rangle \models p$ iff there exists a path $n_0 (\in N_0), n_1, \ldots$ in G such that $s_i \models \lambda(n_i)$ for every $i \geq 0$. An assertion graph for a given p can always be automatically generated ([MP95]). Conversely, any graph G is the assertion graph for some general safety formula p. More precisely, we have,

Lemma 1. *Let $G = \langle N, N_0, E, \lambda \rangle$ be a labeled graph. There exists a temporal operator C_G of arity $|N|$, which is monotone in every argument, such that G is the assertion graph of $C_G((\lambda(n))_{n \in N})$.*

For a node $n \in N$, we say that a state s is $(\mathcal{S}, n)$-*accessible* if there exists a computation $s_0, s_1, \ldots, s_k, \ldots$, of $\mathcal{S}$, where $s = s_k$, and a finite path $n_0, n_1, \ldots, n_k = n$ in G such that $n_0 \in N_0$ and $\forall i \in [0..k] \cdot s_i \models \lambda(n_i)$. An assertion φ is $(\mathcal{S}, n)$-*valid* if it holds for every $(\mathcal{S}, n)$-accessible state.

For transition system $\mathcal{S} = \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$,
general safety formula p,
assertion graph $G = \langle N, N_0, E, \lambda \rangle$ for p, and
assertions $\{\varphi_n\}_{n \in N}$

$$
\begin{array}{lll}
S1 & \Theta & \rightarrow \bigvee_{n \in N_0} \varphi_n \\
S2 & \varphi_n & \rightarrow \lambda(n) \qquad\qquad n \in N \\
S3 & \{\varphi_n\} \; \mathcal{T} \; \{\bigvee_{(n,m) \in E} \varphi_m\} & n \in N \\
\hline
 & \mathcal{S} \models p &
\end{array}
$$

Fig. 6. Rule SAFE (general safety)

The verification rule SAFE shown in Figure 6 reduces the verification of the general safety formula p to first-order premises $S1{-}S3$. Suppose we have a transition system $\mathcal{S}$, a general safety formula p and an assertion graph G for p. To prove that p is $\mathcal{S}$-valid, we have to find *intermediate assertions* $\{\varphi_n\}_{n \in N}$ such that premise $S1$ is $\mathcal{S}$-state-valid and for each $n \in N$, $S2$ and $S3$ are $(\mathcal{S}, n)$-valid.

Lemma 2. *Rule* SAFE *is sound.*

Proof Assuming the premises of $S1{-}S3$ of SAFE are satisfied, we must show that p is $\mathcal{S}$-valid. Consider an arbitrary computation $\sigma : s_0, s_1, \ldots$ of $\mathcal{S}$, i.e.,

$$s_0 \models \Theta \quad \text{and} \quad \forall i \geq 0 \; \exists \tau \in \mathcal{T} \cdot (s_i, s_{i+1}) \models \rho_\tau(\overline{x}, \overline{x}') \ .$$

We show that $\sigma \models p$. By premise $S3$ we have

$$\forall i \geq 0 \; \forall n \in N \cdot [s_i \models \varphi_n \quad \rightarrow \quad \exists m \cdot (n, m) \in E \ \wedge \ s_{i+1} \models \varphi_m] \ .$$

By premise $S1$, $\exists n \in N_0 \cdot s_0 \models \varphi_n$. Thus, by induction on i, it follows that $\forall i \geq 0 \; \exists n \in N \cdot s_i \models \varphi_n$. Hence, by $S2$, σ induces a path $n_0 (\in N_0), n_1, \ldots$ on G such that $s_i \models \lambda(n_i)$ for every $i \geq 0$. Therefore σ is a model of p. ∎

Lemma 3. *Rule* SAFE *is complete.*

Proof (sketch) Assuming p is $\mathcal{S}$-valid we must establish that, given an assertion graph G for p, there are assertions $\{\varphi_n\}_{n \in N}$ satisfying premises $S1{-}S3$. By encoding finite sequences and elementary operations on these in the assertion

language $\mathcal{A}$ (assumed to be sufficiently expressive), it is possible to construct a formula $acc(\mathcal{S}, n)$ for each $n \in N$, such that:

$$s \models acc(\mathcal{S}, n) \quad \text{iff} \quad s \text{ is } (\mathcal{S}, n)\text{-accessible} \ .$$

The assertions $\{acc(\mathcal{S}, n)\}_{n \in N}$ trivially satisfy the premises of SAFE. ∎

Finding useful strengthened assertions $\{\varphi_n\}_{n \in N}$ for rule SAFE is the main obstacle in its use; assertions $\{acc(\mathcal{S}, n)\}_{n \in N}$ are seldom very useful. Similar to the analysis of INV we now give fixedpoint characterizations of the strongest and weakest candidates for SAFE.

Notation: For an ordered set N we will use $(a_n)_{n \in N}$ or a_N for a tuple indexed by the elements in N.

4.1 Forward Propagation

To generate a unique least set of intermediate assertions for SAFE we need to work with deterministic assertion graphs. We say that an assertion graph G is *deterministic* if any model has at most one path in G, that is, for any n_1, n_2, if $n_1, n_2 \in N_0$ or for some $m \in N$, $(m, n_1), (m, n_2) \in E$ then $\lambda(n_1) \wedge \lambda(n_2)$ is unsatisfiable. We also define G to be *edge-complete* if any model whose first state satisfies $\bigvee_{n \in N_0}$ has at least one path in G, that is, for any $n \in N$, $\bigvee_{(n,m) \in E} \lambda(m)$ holds.

If G is not deterministic then there may be incomparable minimal (with respect to the order on $\mathcal{A}^{|N|}$ defined pointwise by the implication relation) tuples of assertions satisfying $S1{-}S3$.

Example (two minimal solutions for $S1{-}S3$) Consider a graph with two nodes, $N = N_0 = \{n_1, n_2\}$, labeled $\lambda(n_1) : x = 0$ and $\lambda(n_2) : y = 0$, such that any two nodes are connected. This graph is an assertion graph for the property $\Box(x = 0 \ \vee \ y = 0)$ and is not deterministic because $n_1, n_2 \in N_0$ and $\lambda(n_1) \wedge \lambda(n_2) \ \leftrightarrow \ x = 0 \wedge y = 0$ is satisfiable. Consider the transition system with $\Theta : x = 0 \wedge y = 0$ and only one transition, τ_I, the idling transition, defined by the transition relation $\rho_\tau : \ x' = x \ \wedge \ y' = y$. In this system there are two minimal solutions to $S1{-}S3$, namely

$$\begin{cases} \varphi_{n_1} : \ x = 0 \ \wedge \ y = 0 \\ \varphi_{n_2} : \ \text{F} \end{cases} \qquad \begin{cases} \varphi_{n_1} : \ \text{F} \\ \varphi_{n_2} : \ x = 0 \ \wedge \ y = 0 \end{cases}$$ ∎

For deterministic assertion graphs G, we will show that we can define a forward propagation operator $\mathcal{F}_N : \mathcal{A}^{|N|} \mapsto \mathcal{A}^{|N|}$ such that if $S1{-}S3$ can be established and the propagation sequence associated with $\mathcal{F}$ converges after finitely many iterations, then its limit is the unique minimal solution to $S1{-}S3$. Note that a nondeterministic assertion graph can be transformed into a deterministic one by a standard automata-theoretic construction, *so in the following we assume a deterministic G.*

First we make the *observation* that a tuple φ_N satisfies premises $S1{-}S3$ of rule SAFE iff it satisfies

$(C1a)\ \ \Theta \wedge \lambda(n) \rightarrow \varphi_n,\ \ n \in N_0$ $\qquad$ $(C1b)\ \ \Theta \rightarrow \bigvee_{n \in N_0} \lambda(n)$

$(C2)\ \ \varphi_n \rightarrow \lambda(n),\ \ n \in N$

$(C3a)\ \ \mathrm{SP}(\mathcal{T}, \varphi_m) \wedge \lambda(n) \rightarrow \varphi_n,\ \ (m,n) \in E$ $\qquad$ $(C3b)\ \ \{\varphi_m\}\mathcal{T}\{\bigvee_{(m,n)\in E} \lambda(n)\}$

This is because:

- $S2$ is identical to $(C2)$.
- $(C1a)$ and $(C1b)$ imply $S1$.
- $S1$ and $S2$ imply $(C1b)$, and $S2$ implies $(C1a)$ since

$$\begin{aligned}
\Theta \wedge \lambda(n) &\rightarrow \lambda(n) \wedge (\textstyle\bigvee_{n \in N_0} \varphi_n) \\
&\rightarrow \bigwedge_{m \neq n \wedge m \in N_0} \neg\lambda(m) \wedge \lambda(n) \wedge (\bigvee_{n \in N_0} \varphi_n) \quad \text{since } G \text{ is deterministic} \\
&\rightarrow \bigwedge_{m \neq n \wedge m \in N_0} \neg\varphi_m \wedge \lambda(n) \wedge (\bigvee_{n \in N_0} \varphi_n) \quad \text{by } S2 \\
&\rightarrow \varphi_n
\end{aligned}$$

- $(C3a)$ and $(C3b)$ imply $S3$ since for $m \in N$:

$$\begin{aligned}
\mathrm{SP}(\mathcal{T}, \varphi_m) &\rightarrow \bigvee_{(m,n)\in E} \lambda(n) && \text{by } (C3b) \\
&\rightarrow \bigvee_{(m,n)\in E} (\mathrm{SP}(\mathcal{T}, \varphi_m) \wedge \lambda(n)) \\
&\rightarrow \bigvee_{(m,n)\in E} \varphi(n) && \text{by } (C3a)
\end{aligned}$$

- $S3$ implies $(C3a)$ since for $(m,n) \in E$:

$$\begin{aligned}
\mathrm{SP}(\mathcal{T}, \varphi_m) \wedge \lambda(n) &\rightarrow \lambda(n) \wedge \bigvee_{(m,n')\in E} \varphi'_n && \text{by } S3 \\
&\rightarrow \lambda(n) \wedge \bigwedge_{(m,n')\in E} \neg\lambda(n') \wedge \bigvee_{(m,n')\in E} \varphi'_n \quad \text{since } G \text{ is deterministic} \\
&\rightarrow \lambda(n) \wedge \bigwedge_{(m,n')\in E} \neg\varphi'_n \wedge \bigvee_{(m,n')\in E} \varphi'_n \quad \text{by } S2 \\
&\rightarrow \varphi(n)
\end{aligned}$$

  $S2$ and $S3$ imply $(C3b)$, since

$$\mathrm{SP}(\mathcal{T}, \varphi_m) \underbrace{\rightarrow}_{\text{by } S3} \bigvee_{(m,n)\in E} \varphi_n \underbrace{\rightarrow}_{\text{by } S2} \bigvee_{(m,n)\in E} \lambda(n)$$

Consider the monotone operator $\mathcal{F}_N : \mathcal{A}^{|N|} \mapsto \mathcal{A}^{|N|}$ defined pointwise, at each $n \in N$, by

$$\mathcal{F}_n(\varphi_N) \ \triangleq\ \left[(\Theta \wedge n \in N_0) \vee \textstyle\bigvee_{(m,n)\in E} \mathrm{SP}(\mathcal{T}, \varphi_m)\right] \wedge \lambda(n)\ .$$

A tuple of formulas φ_N satisfies $(C1a)$ and $(C3a)$ iff φ_N satisfies $\mathcal{F}_n(\varphi_N) \to \varphi_n$ for every $n \in N$.

If the sequence

$$(\mathrm{F})_{n\in N}, \quad \mathcal{F}_N((\mathrm{F})_{n\in N}), \quad \mathcal{F}_N^{(2)}((\mathrm{F})_{n\in N}), \quad \ldots$$

reaches a limit $\varphi_{\mathcal{F},N}$ in finitely many steps, then $\varphi_{\mathcal{F},N}$ is the least fixedpoint of $\mathcal{F}_N$.

Notice that $(C2)$ is always satisfied by $\varphi_{\mathcal{F},N}$, since $\varphi_{\mathcal{F},n} \leftrightarrow \mathcal{F}_n(\varphi_{\mathcal{F},N})$ and $\mathcal{F}_n(\varphi_{\mathcal{F},N}) \to \lambda(n)$ for $n \in N$. We can make the connection between $S1{-}S3$ and more precise by

Lemma 4. *$S1{-}S3$ have a solution iff $(C1b)$ holds and $\varphi_{\mathcal{F},N}$ satisfies $(C3b)$.*

Proof Let $\varphi_{\mathcal{F},N}$ be the least fixedpoint of $\mathcal{F}_N$, then it is the least tuple $\varphi_{\mathcal{F},N}$ satisfying

$$\mathcal{F}_n(\varphi_{\mathcal{F},N}) \;\to\; \varphi_{\mathcal{F},n} \qquad \forall n \in N$$

$\Leftarrow$ As we observed,

$$\begin{array}{ll} (C1a) & : \Theta \;\wedge\; \lambda(n) \;\to\; \varphi_{\mathcal{F},n} \\ (C3a) & : \mathrm{SP}(\mathcal{T}, \varphi_{\mathcal{F},m}) \;\wedge\; \lambda(n) \;\to\; \varphi_{\mathcal{F},n} \qquad (m,n) \in E \\ (C2) & : \varphi_{\mathcal{F},n} \;\to\; \lambda(n) \end{array}$$

If furthermore $(C1b)$ and $(C3b)$ hold, then by our *observation* $S1{-}S3$ are satisfied by $\varphi_{\mathcal{F},N}$.

$\Rightarrow$ Assume there is a tuple φ_N satisfying $S1{-}S3$. We have to show, that $(C1b)$ holds and $\varphi_{\mathcal{F},N}$ satisfies $(C3b)$.

$(C1b)$ holds since $S1$ and $S2$ are satisfiable.

By the *observation* φ_N satisfies $(C1a)$ and $(C3a)$, so

$$\mathcal{F}_n(\varphi_N) \;\to\; \varphi_n \qquad \forall n \in N.$$

Since $\varphi_{\mathcal{F},N}$ is the least fixedpoint of $\mathcal{F}_N$, also

$$\varphi_{\mathcal{F},N} \;\to\; \varphi_n \qquad \forall n \in N.$$

Therefore

$$\begin{array}{rl} \mathrm{SP}(\mathcal{T}, \varphi_{\mathcal{F},m}) \to & \mathrm{SP}(\mathcal{T}, \varphi_n) \;\text{ by montonicity of SP} \\ \to & \bigvee_{(m,n)\in E} \varphi_n \text{ by } S3 \\ \to & \bigvee_{(m,n)\in E} \lambda_n \text{ by } S2 \end{array}$$

hence $(C3b)$ holds.

Notice that $\varphi_{\mathcal{F},N}$ precisely characterizes the $(\mathcal{S},n)$-accessible states.

For a graphical interpretation of the propagation method associated with $\mathcal{F}_N$, we define the *extension* G^e of G, which is a deterministic, edge-complete supergraph of G. Suppose that n_f and $\{n^e\}_{n\in N}$ are new distinct symbols not in N. Let $G^e = \langle N^e, N_0^e, E^e, \lambda^e\rangle$, where

$$\begin{array}{ll} N^e = N \cup \{n_f\} \cup \{n^e \mid n \in N\} & \lambda^e(n) \;\; = \lambda(n) \quad \text{if } n \in N \\ N_0^e = N_0 & \lambda^e(n^e) = \neg(\bigvee_{(n,m)\in E} \lambda(m)) \\ E^e \;= E \cup \{(n,n^e),(n^e,n_f) \mid n \in N\} & \lambda^e(n_f) = \mathrm{T} \end{array}$$

The n^e nodes can be considered as escape nodes for those computations that after reaching n have nowhere to go in G. All the computations that fail to stay in G end looping in the node n_f. Using G^e, we can characterize the models of both p and $\neg p$. A model σ satisfies p iff its (unique) path in G^e stays in N and conversely, it satisfies $\neg p$ iff its first state does not satisfy $\bigvee_{n\in N_0} \lambda(n)$ or its path in G^e reaches a node in $N^e \setminus N$.

Example (G **and** G^e) For program BAKERY in Figure 3, the 1-bounded overtaking property for process P_1 is expressed by the general safety formula

$$p_{bound}: \quad \Box[\ \ell_2 \ \rightarrow\ (\ell_2 \wedge \neg m_3)\ W\ ((\ell_2 \wedge m_3)\ W\ ((\ell_2 \wedge \neg m_3)\ W\ \ell_3))]\ .$$

That is, whenever process P_1 reaches ℓ_2, process P_2 can access its critical region at most once before P_1 reaches its critical region at ℓ_3.

An assertion graph G for p_{bound} and its extension G^e are shown in Figure 7. The nodes in $N^e \setminus N$ are drawn with two concentric ovals. Statechart conventions [Har84] are used for a more compact graphical representation. For example, the edge departing the compound node containing n_1, n_2 and n_3 represents the edges $(n_1,n_4),(n_2,n_4),(n_3,n_4)$.

We extend the definition of the forward operator $\mathcal{F}_N$ to all the nodes of G^e, so for all $n \in N^e$,

$$\mathcal{F}_n(\varphi_{N^e}) \quad \triangleq \quad \left[(\Theta \wedge n \in N_0^e) \vee \bigvee_{(m,n)\in E^e} \mathrm{SP}(\mathcal{T},\varphi_m)\right] \ \wedge\ \lambda^e(n)\ .$$

With this definition, condition $(C3b)$ can be replaced by the condition that $\varphi_{\mathcal{F},n}$ is unsatisfiable for any $n \in N^e \setminus N$. If the sequence $(\mathrm{F})_{n\in N^e}$, $\mathcal{F}_{N^e}((\mathrm{F})_{n\in N^e})$, $\mathcal{F}^{(2)}_{N^e}((\mathrm{F})_{n\in N^e})$, $\ldots$ converges in finitely many steps to $\varphi_{\mathcal{F},N^e}$, then

- $C_{G^e}(\varphi_{N^e})$ is $\mathcal{S}$-valid.
- p is $\mathcal{S}$-valid iff $(C1b)$ holds and for any $n \in N^e \setminus N$, $\varphi_{\mathcal{F},n}$ is unsatisfiable.

Thus the bottom-up approach to verifying p consists of the following steps: check that $(C1b)$ holds, generate the least fixedpoint, $\varphi_{\mathcal{F},N^e}$, of $\mathcal{F}_{N^e}$ and finally check that $\forall n \in N^e \setminus N$, $\varphi_{\mathcal{F},n}$ is unsatisfiable. Notice that, unlike the invariance case, forward propagation for general safety formulas depends on the given property.

Example (generating intermediate assertions for rule SAFE) Consider

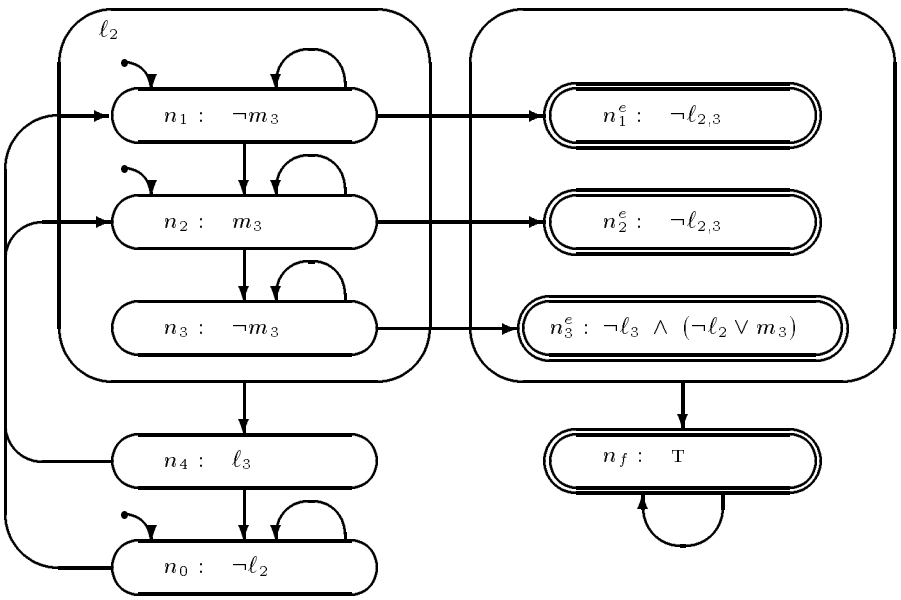


Fig. 7. Graph G and its extension G^e for property p_{bound}

$$\begin{array}{l} \textbf{local} \quad e, f \quad : \textbf{integer where } e = 1, f = 0 \\ \qquad\quad x, y, z : \textbf{integer where } x = y = z = 0 \end{array}$$

$$Prod :: \left[\begin{array}{l} \textbf{loop forever do} \\ \quad \left[\begin{array}{l} \ell_0\text{: } \textbf{produce } x \\ \ell_1\text{: } \textbf{request } e \\ \ell_2\text{: } y := x \\ \ell_3\text{: } \textbf{release } f \end{array}\right] \end{array}\right] \;||\; Cons :: \left[\begin{array}{l} \textbf{loop forever do} \\ \quad \left[\begin{array}{l} m_0\text{: } \textbf{request } f \\ m_1\text{: } z := y \\ m_2\text{: } \textbf{release } e \\ m_3\text{: } \textbf{consume } z \end{array}\right] \end{array}\right]$$

Fig. 8. Program PROD-CONS

program PROD-CONS (Figure 8), a simple version of a producer-consumer protocol. The statement **produce** x assigns some nonzero value to x and **consume** x sets x to 0. They do not change other variables. Statements **request** and **release** stand for the standard semaphores P and V. We want to establish the causality property that a given value u is not consumed unless it was first produced:

$$p_{caus}: \quad \Box\,[z = u \;\rightarrow\; \diamondsuit\!\!\!-\, x = u] \quad .$$

A deterministic graph G and its edge-complete extension G^e are shown in Figure 9. Forward propagation with program PROD-CONS and formula p_{caus} generates disjuncts from φ_{n_1} as shown in Table 2. The first column is the disjunct

number and the second is the iteration when it was obtained. The top row represents the result of the first iteration $\mathcal{F}_{n_1}((\text{F})_{n\in N})$, the first two rows the result after the second iteration, and so on. As $\{n_1\}$ is a maximal strongly connected component that cannot be reached from other components, we can perform the forward propagation first on $\{n_1\}$. Each disjunct is obtained from the initial condition or is the result of applying strongest postcondition to a previously obtained disjunct. For instance, disjunct 5, obtained at the fifth iteration, generates disjuncts 6 and 7 in the table. The propagation is completed by the eleventh iteration. We can now propagate $\varphi_{\mathcal{F},n_1}$ to other components in the graph. We obtain $\varphi_{\mathcal{F},n_1^e} = \text{F}$ and therefore $\varphi_{\mathcal{F},n_f} = \text{F}$. We can also compute $\varphi_{\mathcal{F},n_2}$ and $\varphi_{\mathcal{F},n_3}$, but since $N^e \setminus N$ is not reachable from these nodes it is not important what the actual formulas are. (We could take $\varphi_{n_2} : x = u$ and $\varphi_{n_3} : \text{T}$ together with the other formulas to satisfy conditions $S1$–$S3$). As $\varphi_{n_1^e} : \text{F}$, we have proved that program PROD-CONS satisfies formula p_{caus}. ⌟

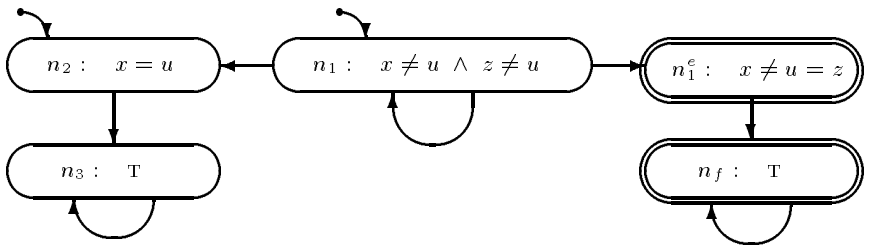


Fig. 9. Graphs G and G^e for the formula p_{caus}

Table 2. The first few disjuncts of φ_{n_1} where $c : \ u \neq 0 \ \wedge \ u \neq x \ \wedge \ u \neq y$

#	it.	source	disjunct
1	1	Θ	$\ell_0 \ \wedge \ m_0 \ \wedge \ x = 0 \ \wedge \ y = 0 \ \wedge \ z = 0 \ \wedge \ (e,f) = (1,0) \ \wedge \ c$
2	2	$n_1, 1$	$\ell_1 \ \wedge \ m_0 \ \wedge \ x \neq 0 \ \wedge \ y = 0 \ \wedge \ z = 0 \ \wedge \ (e,f) = (1,0) \ \wedge \ c$
3	3	$n_1, 2$	$\ell_2 \ \wedge \ m_0 \ \wedge \ x \neq 0 \ \wedge \ y = 0 \ \wedge \ z = 0 \ \wedge \ (e,f) = (0,0) \ \wedge \ c$
4	4	$n_1, 3$	$\ell_3 \ \wedge \ m_0 \ \wedge \ x \neq 0 \ \wedge \ y = x \ \wedge \ z = 0 \ \wedge \ (e,f) = (0,0) \ \wedge \ c$
5	5	$n_1, 4$	$\ell_0 \ \wedge \ m_0 \ \wedge \ x \neq 0 \ \wedge \ y = x \ \wedge \ z = 0 \ \wedge \ (e,f) = (0,1) \ \wedge \ c$
6	6	$n_1, 5$	$\ell_1 \ \wedge \ m_0 \ \wedge \ x \neq 0 \ \wedge \ y \neq 0 \ \wedge \ z = 0 \ \wedge \ (e,f) = (0,1) \ \wedge \ c$
7	6	$n_1, 5$	$\ell_0 \ \wedge \ m_1 \ \wedge \ x \neq 0 \ \wedge \ y = x \ \wedge \ z = 0 \ \wedge \ (e,f) = (0,0) \ \wedge \ c$
			$\ldots$

The bottom-up method for proving a safety formula p can also be used for debugging. With a deterministic assertion graph G and its extension G^e for p we generate assertions φ_{N^e}, expressing the $(\mathcal{S}, n)$-accessible states, in stages. Stage

$i+1$ is obtained from stage i by applying $\mathcal{F}_{N^e}$. The propagation stops when we obtain a satisfiable $\varphi_{n,i}$ for $n \in N^e \setminus N$ and $i \geq 0$. By recording the history of previous iterations we can reconstruct the symbolic computation that ends outside G.

Example (**counterexamples using forward propagation**) Consider again program PROD-CONS (Figure 8). Suppose we want the protocol to have a lazy production property, namely that once the producer writes a value into x it will wait until that value is read by the consumer before assigning a new value to x. This property is expressed by the formula

$$p_{ord}: \quad \Box[x = u \rightarrow ((x = u \wedge z \neq u)\ W\ (x = u \wedge z = u))\ \ W\ x \neq u]$$

where u is an auxiliary integer variable used to record the value of x. Figure 10 shows the deterministic graph G of all models of p_{ord}, and its extension G^e.

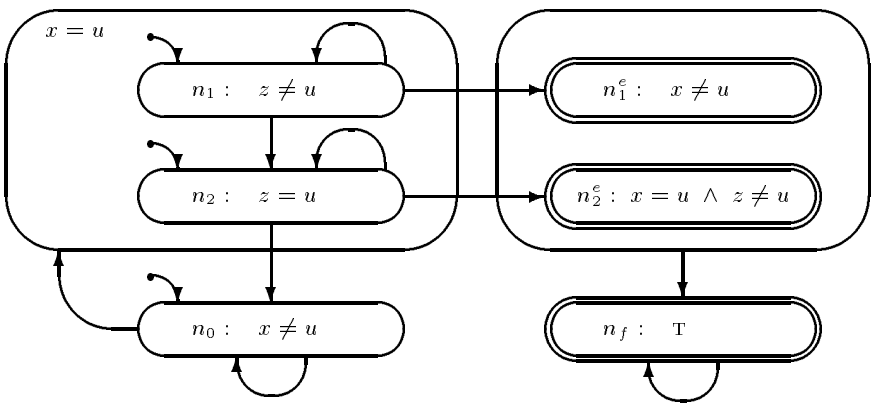


Fig. 10. Graphs G and G^e for the formula p_{ord}

Table 3 shows the disjuncts produced in the first six applications of the forward propagation operator $\mathcal{F}_N$ and maintains enough information to construct an abstract counterexample. We stopped when we obtained a satisfiable disjunct for a node in $N^e \setminus N$. A counterexample $s_1, s_2, \ldots, s_6, s_6, \ldots$ is generated by tracing back the origins of the first disjunct of $\varphi_{n_1^e}$:

$$\begin{array}{ll}
s_1 \models \ell_0 \wedge m_0 \wedge 0 = x \neq u \wedge y = 0 \wedge z = 0 \wedge e = 1 \wedge f = 0 & \text{in node } n_0 \\
s_2 \models \ell_1 \wedge m_0 \wedge 0 \neq x = u \wedge y = 0 \wedge z = 0 \wedge e = 1 \wedge f = 0 & \text{in node } n_1 \\
s_3 \models \ell_2 \wedge m_0 \wedge 0 \neq x = u \wedge y = 0 \wedge z = 0 \wedge e = 0 \wedge f = 0 & \text{in node } n_1 \\
s_4 \models \ell_3 \wedge m_0 \wedge 0 \neq x = u \wedge y = u \wedge z = 0 \wedge e = 0 \wedge f = 0 & \text{in node } n_1 \\
s_5 \models \ell_0 \wedge m_0 \wedge 0 \neq x = u \wedge y = u \wedge z = 0 \wedge e = 0 \wedge f = 1 & \text{in node } n_1 \\
s_6 \models \ell_1 \wedge m_0 \wedge 0 \neq x \neq u \wedge y = u \wedge z = 0 \wedge e = 0 \wedge f = 1 & \text{in node } n_1^e
\end{array}$$

Table 3. Forward propagation for PROD-CONS and p_{ord} (6 iterations) where $c : z = 0$

	#	it.	source	disjunct												
φ_{n_0}	1	1	Θ	ℓ_0	$\wedge$	m_0	$\wedge$	$0 = x \neq u$	$\wedge$	$y = 0$	$\wedge$	$(e,f) = (1,0)$	$\wedge$	c
	2	2	$n_0, 1$	ℓ_1	$\wedge$	m_0	$\wedge$	$0 \neq x \neq u$	$\wedge$	$y = 0$	$\wedge$	$(e,f) = (1,0)$	$\wedge$	c
	3	3	$n_0, 2$	ℓ_2	$\wedge$	m_0	$\wedge$	$0 \neq x \neq u$	$\wedge$	$y = 0$	$\wedge$	$(e,f) = (0,0)$	$\wedge$	c
	4	4	$n_0, 3$	ℓ_3	$\wedge$	m_0	$\wedge$	$0 \neq x \neq u$	$\wedge$	$y = x$	$\wedge$	$(e,f) = (0,0)$	$\wedge$	c
	5	5	$n_0, 4$	ℓ_0	$\wedge$	m_0	$\wedge$	$0 \neq x \neq u$	$\wedge$	$y = x$	$\wedge$	$(e,f) = (0,1)$	$\wedge$	c
	6	6	$n_0, 5$	ℓ_1	$\wedge$	m_0	$\wedge$	$0 \neq x \neq u$	$\wedge$	$y \neq u$	$\wedge$	$(e,f) = (0,1)$	$\wedge$	c
	7	6	$n_0, 5$	ℓ_0	$\wedge$	m_1	$\wedge$	$0 \neq x \neq u$	$\wedge$	$y = x$	$\wedge$	$(e,f) = (0,0)$	$\wedge$	c
φ_{n_1}	1	2	$n_0, 1$	ℓ_1	$\wedge$	m_0	$\wedge$	$0 \neq x = u$	$\wedge$	$y = 0$	$\wedge$	$(e,f) = (1,0)$	$\wedge$	c
	2	3	$n_1, 1$	ℓ_2	$\wedge$	m_0	$\wedge$	$0 \neq x = u$	$\wedge$	$y = 0$	$\wedge$	$(e,f) = (0,0)$	$\wedge$	c
	3	4	$n_1, 2$	ℓ_3	$\wedge$	m_0	$\wedge$	$0 \neq x = u$	$\wedge$	$y = u$	$\wedge$	$(e,f) = (0,0)$	$\wedge$	c
	4	5	$n_1, 3$	ℓ_0	$\wedge$	m_0	$\wedge$	$0 \neq x = u$	$\wedge$	$y = u$	$\wedge$	$(e,f) = (0,1)$	$\wedge$	c
	5	6	$n_1, 4$	ℓ_0	$\wedge$	m_1	$\wedge$	$0 \neq x = u$	$\wedge$	$y = u$	$\wedge$	$(e,f) = (0,0)$	$\wedge$	c
	6	6	$n_1, 4$	ℓ_1	$\wedge$	m_0	$\wedge$	$0 \neq x = u$	$\wedge$	$y = u$	$\wedge$	$(e,f) = (0,1)$	$\wedge$	c
	7	6	$n_0, 5$	ℓ_1	$\wedge$	m_0	$\wedge$	$0 \neq x = u$	$\wedge$	$y \neq u$	$\wedge$	$(e,f) = (0,1)$	$\wedge$	c
φ_{n_2}	1	1	Θ	ℓ_0	$\wedge$	m_0	$\wedge$	$0 = x = u$	$\wedge$	$y = 0$	$\wedge$	$(e,f) = (1,0)$	$\wedge$	c
$\varphi_{n_1^e}$	1	6	$n_1, 4$	ℓ_1	$\wedge$	m_0	$\wedge$	$0 \neq x \neq u$	$\wedge$	$y = u$	$\wedge$	$(e,f) = (0,1)$	$\wedge$	c
$\varphi_{n_2^e}$														

The counterexample shows that it is possible that *Prod* produces a new value before *Cons* has a chance to read the previous one. A fix to this problem is to switch statements ℓ_0 and ℓ_1. ∎

4.2 Backward Propagation

The backward propagation operator $\mathcal{B}_N : \mathcal{A}^{|N|} \mapsto \mathcal{A}^{|N|}$, on an assertion graph G, is defined pointwise as

$$\mathcal{B}_n(\varphi_N) \quad \stackrel{\triangle}{=} \quad \lambda(n) \ \wedge \ \mathrm{WP}(\mathcal{T}, \bigvee_{(n,m)\in E} \varphi_m) \ .$$

A tuple of formulas φ_N satisfies $S2$ and $S3$ iff it satisfies $\mathcal{B}_n(\varphi_N) \leftarrow \varphi_n$ for any $n \in N$. If the sequence $(\mathrm{T})_{n\in N}, \mathcal{B}_N((\mathrm{T})_{n\in N}), \mathcal{B}_N^{(2)}((\mathrm{T})_{n\in N}), \ldots$ converges in finitely many steps to a tuple of formulas $\varphi_{\mathcal{B},N}$ then $\varphi_{\mathcal{B},N}$ is the maximal tuple satisfying $S2$ and $S3$. The formula $\varphi_{\mathcal{B},n}$ characterizes those states s with the property that for any sequence of states $\sigma : s_0{=}s, s_1, \ldots$ originating in s such that $\forall i \geq 0 \cdot s_{i+1} \in \tau(s_i)$ there is a path $n_0{=}n, n_1, \ldots$ in G starting in n such that for any $i \geq 0$, $s_i \models \lambda(n_i)$. Furthermore,

$$p \text{ is } \mathcal{S}\text{-valid iff } \ \Theta \rightarrow \bigvee_{n\in N_0} \varphi_{\mathcal{B},n} \ .$$

Thus a top-down approach to verifying p consists of first generating the greatest fixedpoint, $\varphi_{\mathcal{B},N}$, of $\mathcal{B}_N$ and then checking that $\Theta \rightarrow \bigvee_{n\in N_0} \varphi_{\mathcal{B},n}$ holds.

Example (generating intermediate assertions for rule SAFE) Consider program BAKERY (Figure 3) and the 1-bounded overtaking property

$$p_{bound}: \quad \Box[\ell_2 \ \rightarrow \ (\ell_2 \wedge \neg m_3) \ W \ ((\ell_2 \wedge m_3) \ W \ ((\ell_2 \wedge \neg m_3) \ W \ \ell_3))] \ .$$

An assertion graph G for p_{bound} and its extended graph G^e are shown in Figure 7. Forward propagation does not terminate, but the property can be proved using backward propagation. To compute the limit of the sequence $(\mathrm{T})_{n\in N}$, $\mathcal{B}_N((\mathrm{T})_{n\in N})$, $\mathcal{B}_N^{(2)}((\mathrm{T})_{n\in N}), \ldots$ we assume the invariant $\Box(\ell_2 \rightarrow y_1 \geq 1)$. This invariant can be generated by the methods presented in Sections 7 and 8.

One iteration of $\mathcal{B}$ generates $\varphi_n = \lambda(n)$ for each $n \in N$. In the second iteration we get

$$\begin{aligned}\varphi_{n_3} &= \lambda(n_3) \ \wedge \ \mathrm{WP}(\mathcal{T}, \lambda(n_3) \vee \lambda(n_4)) \ = \ (\neg m_3 \wedge \ell_2) \wedge \mathrm{WP}(\mathcal{T}, \ell_3 \vee (\neg m_3 \wedge \ell_2)) \\ &= (\neg m_3 \wedge \ell_2) \wedge \underbrace{(m_2 \rightarrow y_2 \geq y_1 \wedge y_1 \neq 0)}_{\mathrm{WP}(\tau_{m_2}, \neg m_3 \wedge \ell_2)} \ = \ (\neg m_3 \wedge \ell_2) \wedge \underbrace{(m_2 \rightarrow y_2 \geq y_1)}_{\text{by } \Box(\ell_2 \rightarrow y_1 \geq 1)}\end{aligned}$$

The other φ's remain at their previous values. A third application of $\mathcal{B}$ shows that a fixedpoint was reached. Since Θ implies φ_{n_0} and $n_0 \in N_0$, premise $S1$ holds. Backward propagation has thus provided the necessary intermediate assertions to establish the 1-bounded overtaking property.

5 Approximate Analysis

As we have seen, there is no guarantee for success in forward and backward analysis. For infinite or even large finite-state systems, the propagation may not terminate or, even if the sequences converge in a finite number of iterations, we may not be able to prove it (we might not be able to prove for instance that $\mathcal{F}^{n+1}(\mathrm{F}) \rightarrow \mathcal{F}^n(\mathrm{F})$). A solution to the difficulties in generating invariants is to limit the search for valid properties and intermediate assertions to a domain $\mathcal{D}$ where we can find approximations to the least (greatest) fixedpoint of the forward (backward) operator. At a minimum, the domain $\mathcal{D}$ is required to support a partial order $\leq$ corresponding to implication in $\mathcal{A}$. The connection between $\mathcal{D}$ and the assertion language $\mathcal{A}$ is given by a monotone function

$$\gamma : \mathcal{D} \mapsto \mathcal{A} \ .$$

We say that $\mathcal{D}$ *abstracts* $\mathcal{A}$ and that γ is the *concretization function.* Abstract interpretation, introduced in [CC77], offers a general framework for the study of abstractions that preserve certain properties. It has been used to lift the analysis of these properties to a domain where it is easier to carry out.

Invariance

In the approximate analysis, the forward propagation operator $\mathcal{F}$ is relaxed to a weaker operator

$$\overline{\mathcal{F}} : \mathcal{D} \mapsto \mathcal{D} \qquad \text{such that} \qquad \mathcal{F} \circ \gamma \ \rightarrow \ \gamma \circ \overline{\mathcal{F}} \ .$$

This condition guarantees the properties

1. if $\overline{\mathcal{F}}(a) \leq a$ then $\mathcal{F}(\gamma(a)) \rightarrow \gamma(a)$,
2. p is $\mathcal{S}$-valid if $\overline{\mathcal{F}}(a) \leq a$ and $\gamma(a) \rightarrow p$.

These provide a general, domain-independent, justification of the the bottom-up approximate analysis method. The first property shows that an invariant can be obtained by first generating a solution $a \in \mathcal{D}$ of $\overline{\mathcal{F}}(a) \leq a$ and then computing its concretization $\gamma(a)$. The second property shows that p is $\mathcal{S}$-valid if such an invariant satisfies premise $I2$ of rule INV.

Similarly, $\mathcal{B}$ has to be approximated by a stronger monotone operator

$$\underline{\mathcal{B}} : \mathcal{D} \mapsto \mathcal{D} \qquad \text{such that} \qquad \mathcal{B} \circ \gamma \ \leftarrow \ \gamma \circ \underline{\mathcal{B}} \ .$$

The following properties are the basis for top-down approximate analysis:

1. if $\underline{\mathcal{B}}(a) \geq a$ then $\mathcal{B}(\gamma(a)) \leftarrow \gamma(a)$,
2. p is $\mathcal{S}$-valid if $\underline{\mathcal{B}}(a) \geq a$ and $\Theta \rightarrow \gamma(a)$.

The first property shows that a strengthening of p can be obtained by first generating a solution $a \in \mathcal{D}$ of $\underline{\mathcal{B}}(a) \geq a$ and then computing its concretization $\gamma(a)$. The second property shows that p is $\mathcal{S}$-valid if such an invariant satisfies premise $I1$ of rule INV.

In the domains described in Section 8, the least (greatest) fixedpoint of $\overline{\mathcal{F}}$ ($\underline{\mathcal{B}}$) can be directly computed by solving a system of constraints constructed from $\mathcal{S}$ and the formula p. In other domains, solutions to $\mathcal{F}(\varphi) \rightarrow \varphi$ or $\mathcal{B}(\varphi) \leftarrow \varphi$ can be found by approximating the propagation methods. These domains must have a minimal element $\bot$ for forward propagation and a maximal element $\top$ for the backward propagation. The partial order $\leq$ should be decidable, so we can detect the convergence of the sequences. Furthermore, it should be possible to find a propagation sequence that converges in finitely many steps. The domain of linear systems of equations presented in Section 6 has this property because any monotone sequence of equations is guaranteed to converge in finitely many steps. In the polyhedra domain (Section 7) the propagation sequences do not necessarily converge after a finite number of iterations, but it is possible to achieve convergence by approximating the process further. Table 1 in Section 1 summarizes the properties of the approximate methods we review.

General Safety Formulas

Let p be a safety formula and $\mathcal{D}$ an abstraction of $\mathcal{A}$ with concretization function $\gamma : \mathcal{D} \mapsto \mathcal{A}$. It is possible to approximate the propagation methods for p in $\mathcal{D}$. Indeed, let G be an assertion graph for a safety formula p. We will assume that G is deterministic whenever we consider forward propagation. The exact forward and backward operators are operators on $\mathcal{A}^{|N|}$. The domain $\mathcal{D}^{|N|}$ is an abstraction of $\mathcal{A}^{|N|}$ with the concretization function given on any component of a tuple in $\mathcal{D}^{|N|}$ by $\gamma : \mathcal{D} \mapsto \mathcal{A}$.

In the approximate bottom-up analysis, if condition $(C1b)$ holds, we are interested in generating assertions φ_N that satisfy conditions $(C2)$, $(C1a)$ and $(C3a)$ and then check if they also satisfy condition $(C3b)$. Property $(C2)$ cannot be guaranteed if the labelings $\lambda(n)$ are not in the image of γ. Therefore we first have to strengthen the graph G to a graph $\underline{G}$, which has the same structure as G but a stronger labeling function $\underline{\lambda} = \gamma(\lambda^{\mathcal{D}})$ where $\lambda^{\mathcal{D}} : N \mapsto \mathcal{D}$ is a a labeling of the nodes in the abstraction domain such that

$$\gamma(\lambda^{\mathcal{D}}(n)) \;\rightarrow\; \lambda(n) \;\; .$$

Thus $\underline{G}$ is an assertion graph dependent on the abstraction domain. Approximating G by $\underline{G}$ is sound: the only premise of rule SAFE that depends on the labeling λ is $(C2)$, and if φ_N satisfies $(C2)$ for $\underline{G}$ then it satisfies $(C2)$ for G as well. In the exact forward analysis of safety formulas, we have used a complete deterministic extension of the assertion graph and replaced condition $(C3b)$ by the equivalent condition $\forall n \in N^e \setminus N . \neg\varphi_n$. In the abstraction domains we will make another conservative approximation and dispense with the requirement that the supergraph be deterministic. In this case, $(C3b)$ is only implied by $\forall n \in N^e \setminus N \cdot \neg\varphi_n$.

Let $\mathcal{F}_{N^e}$ be the forward propagation operator associated with $\underline{G}^e$ and let $\overline{\mathcal{F}}_{N^e}$ be an approximation of it in $\mathcal{D}^{|N|}$, that is, a monotone operator such that for each $n \in N^e$ and $a_{N^e} = (a_n)_{n \in N^e} \in \mathcal{D}^{|N^e|}$,

$$\mathcal{F}_n((\gamma(a_n))_{n \in N^e}) \;\;\rightarrow\;\; \gamma(\overline{\mathcal{F}}_n(a_{N^e})) \;\; .$$

The forward operator $\mathcal{F}_{N^e}$ has the following properties:

1. if $\forall n \in N^e \cdot \overline{\mathcal{F}}_n(a_{N^e}) \leq a_n$ then $\forall n \in N^e \cdot \mathcal{F}_n((\gamma(a_n))_{n \in N^e}) \rightarrow \gamma(a_n)$
2. p is $\mathcal{S}$-valid if there exists a tuple of assertions a_N such that
$$\Theta \;\rightarrow\; \bigvee_{n \in N_0} \lambda(n), \;\; \forall n \in N^e \cdot \overline{\mathcal{F}}_n(a_{N^e}) \leq a_n,$$
$$\forall n \in N^e \cdot a_n \leq \lambda^{\mathcal{D}}(n) \text{ and } \forall n \notin N \cdot \neg\gamma(a_n).$$

The first property implies that assertions satisfying conditions $(C1a)$ and $(C3a)$ can be obtained by first generating a tuple a_{N^e} that satisfies $\forall n \in N^e \cdot \overline{\mathcal{F}}_n(a_{N^e}) \leq a_n$ and then computing its concretization $\varphi_{N^e} = (\gamma(a_n))_{n \in N^e}$ in $\mathcal{A}^{|N^e|}$. Under these conditions, φ_N, the restriction of φ_{N^e} to the nodes in N, satisfies $(C1a)$ and $(C3a)$. Notice that if each a_n is bounded by $\lambda^{\mathcal{D}}(n)$ then the assertion tuple φ_N also satisfies condition $(C2)$. The second property shows that if in addition for any $n \notin N$, φ_n is unsatisfiable and condition $(C1b)$ holds, then p is $\mathcal{S}$-valid.

Let $\mathcal{B}$ be the backward operator for G and $\underline{\mathcal{B}}$ an approximation of it in $\mathcal{D}^{|N|}$, i.e., $\mathcal{B}$ is a monotone operator such that for each $n \in N$ and $a_N = (a_n)_{n\in N} \in \mathcal{D}^{|N|}$,

$$\mathcal{B}_n((\gamma(a_n))_{n\in N}) \quad \leftarrow \quad \gamma(\underline{\mathcal{B}}_n(a_N)) \quad .$$

Similar to the invariance case, we have the properties

1. if $\forall n \in N \cdot \underline{\mathcal{B}}_n(a_N) \geq a_n$ then $\mathcal{B}_n((\gamma(a_n))_{n\in N}) \leftarrow \gamma(a_n)$,
2. p is $\mathcal{S}$-valid if $\forall n \in N \cdot \underline{\mathcal{B}}_n(a_N) \geq a_n$ and $\Theta \rightarrow \bigvee_{n\in N_0} \gamma(a_n)$.

The first property shows that assertions satisfying premises $S2$ and $S3$ of rule SAFE can be obtained by first generating a tuple a_N that satisfies $\underline{\mathcal{B}}_n(a_N) \geq a_n$ for all $n \in N$, and then computing its concretization $\varphi_N = (\gamma(a_n))_{n\in N}$ in $\mathcal{A}^{|N|}$. φ_N defines a strengthening of the safety formula p, namely, $C_G(\gamma(a_n))_{n\in N}$. If φ_N also satisfies premise $S1$ of rule SAFE then, by the second property, p is $\mathcal{S}$-valid. These properties define the top-down approximate analysis method for the general safety case.

6 Systems of Linear Equations

We now present a method for discovering *linear relationships* between the variables of a system, that is, formulas expressible as conjuctions of linear equations:

$$(\overline{u}_1 \cdot \overline{x} + v_1 = 0) \wedge \ldots \wedge (\overline{u}_k \cdot \overline{x} + v_k = 0) \quad .$$

The main ideas reviewed here can be found in [Kar76]. More recently the linear equation analysis was generalized to linear congruence equations in [Gra91]. Invariants of this flavor have also been studied extensively in the Petri net literature [Lau73, Rei85], where they are called S-invariants. In [MP95] these ideas are applied to fair transition systems with locations.

Linear invariants are generated in the abstraction domain Λ of linear systems of equations over the variables of the system $\overline{x} = x_1, \ldots, x_n$:

$$\Lambda \quad \stackrel{\Delta}{=} \quad \{\mathcal{E} \mid \mathcal{E}(\overline{x}) : \; U\overline{x} + \overline{v} = 0\},$$

where U ranges over $m \times n$-matrices, and $\overline{v}$ is a column vector of length m. Two systems $\mathcal{E}_1(\overline{x})$ and $\mathcal{E}_2(\overline{x})$ are equal if they have the same set of solutions. Λ is partially ordered by $\sqsubseteq$, where

$\mathcal{E}_1 \sqsubseteq \mathcal{E}_2$ iff the set of solutions of $\mathcal{E}_1$ is a subset of the set of solutions of $\mathcal{E}_2$.

Λ has the following minimal and maximal elements with respect to the order $\sqsubseteq$:

$\bot$: the system with no solution, e.g., the system $0\overline{x} + 1 = 0$
$\top$: the system that admits any solution, $0\overline{x} = 0$.

The concretization function $\gamma : \Lambda \mapsto \mathcal{A}$ is defined by

$$\gamma(\mathcal{E}) \quad \stackrel{\Delta}{=} \quad \bigwedge_{i=1}^{m} (\sum_{j=1}^{n} u_{ij}x_j + v_i = 0) \quad .$$

Dualization Consider the system of equations $\mathcal{E}(\overline{x}) : U\overline{x} + \overline{v} = 0$. Let $\overline{x}_0$ be a solution of $\mathcal{E}$ and $\{\overline{x}_1, \ldots, \overline{x}_k\}$ a basis of solutions for $U\overline{x} = 0$. The homogeneous system of equations

$$\mathcal{E}^d(\overline{u}, v) : \quad \begin{cases} \overline{x}_0 \cdot \overline{u} + v = 0 \\ \overline{x}_1 \cdot \overline{u} \quad\quad = 0 \\ \ldots \\ \overline{x}_k \cdot \overline{u} \quad\quad = 0 \end{cases}$$

is called the *dual* of $\mathcal{E}$. Consider now a homogeneous system of equations $\mathcal{H}(\overline{u}, v) : X\overline{u} + \overline{y}v = 0$. Let $\{(\overline{u}_1, v_1), \ldots, (\overline{u}_n, v_n)\}$ be a basis for the set of solutions of $\mathcal{H}$. The system of equations

$$\mathcal{H}^d(\overline{x}) : \quad \begin{cases} \overline{u}_1 \cdot \overline{x} + v_1 = 0 \\ \ldots \\ \overline{u}_n \cdot \overline{x} + v_n = 0 \end{cases}$$

is called the *dual* of $\mathcal{H}$.

Let $\mathcal{F}^\Lambda : \Lambda \mapsto \Lambda$ be the operator

$$\mathcal{F}^\Lambda(\mathcal{E}) \stackrel{\triangle}{=} \mathcal{E}_\Theta \sqcup \bigsqcup_{\tau \in \mathcal{T}} \tau(\mathcal{E}), \text{ where :}$$

- $\sqcup : \Lambda \times \Lambda \mapsto \Lambda$ is defined by $(\mathcal{E}_1 \sqcup \mathcal{E}_2)^d(\overline{u}, v) \stackrel{\triangle}{=} \left\{ \begin{array}{l} \mathcal{E}_1^d(\overline{u}, v) \\ \mathcal{E}_2^d(\overline{u}, v) \end{array} \right\}$
- $\mathcal{E}_\Theta$: $U_\Theta \overline{x} + \overline{v}_\Theta = 0$, where $\Theta \to U_\Theta \overline{x} + \overline{v}_\Theta = 0$
- $\tau(\mathcal{E})$: Let A_τ and $\overline{b}_\tau$ be such that $\rho_\tau(\overline{x}, \overline{x}') \to \left[A_\tau \begin{pmatrix} \overline{x} \\ \overline{x}' \end{pmatrix} + \overline{b}_\tau = 0 \right]$. Suppose $\mathcal{E} : U\overline{x} + \overline{v} = 0$. Let $\begin{pmatrix} \overline{x}_0 \\ \overline{x}'_0 \end{pmatrix}$ be a solution of $\left\{ \begin{array}{l} A_\tau \begin{pmatrix} \overline{x} \\ \overline{x}' \end{pmatrix} + \overline{b}_\tau = 0 \\ U\overline{x} + \overline{v} = 0 \end{array} \right\}$, and $\left\{ \begin{pmatrix} \overline{x}_1 \\ \overline{x}'_1 \end{pmatrix}, \ldots, \begin{pmatrix} \overline{x}_k \\ \overline{x}'_k \end{pmatrix} \right\}$ be a basis of solutions of $\left\{ \begin{array}{l} A_\tau \begin{pmatrix} \overline{x} \\ \overline{x}' \end{pmatrix} = 0 \\ U\overline{x} = 0 \end{array} \right\}$. Let $X = (\overline{x}'_0 \ \overline{x}'_1 \ \ldots \ \overline{x}'_k)^{\mathrm{T}}$ and $y = (1 \ 0 \ \ldots \ 0)^{\mathrm{T}}$. Then, $\tau(\mathcal{E})$ is the system dual to $X\overline{u} + \overline{y}v = 0$.

This definition guarantees that for any system $\mathcal{E}$,

$$\mathcal{F}(\gamma(\mathcal{E})) \to \gamma(\mathcal{F}^\Lambda(\mathcal{E})) \ .$$

Thus $\mathcal{F}^\Lambda$ is a good approximation of the forward operator $\mathcal{F}$. When the transition system is *linear*, i.e., satisfies $\Theta \leftrightarrow \gamma(\mathcal{E}_\Theta)$ and $\rho_\tau(\overline{x}, \overline{x}') \leftrightarrow A_\tau \begin{pmatrix} \overline{x} \\ \overline{x}' \end{pmatrix} + \overline{b}_\tau = 0$ for every transition τ in $\mathcal{T}$, then $\mathcal{F}^\Lambda$ is the best possible approximation of $\mathcal{F}$ in Λ as summarized in the following proposition:

Proposition5. (Linear completeness) *Let $\mathcal{S}$ be a linear transition system, and $\mathcal{E}_\infty$ be the least fixedpoint of $\mathcal{F}^\Lambda$. If $\mathcal{E}$ is a system of equations such that $\gamma(\mathcal{E})$ is $\mathcal{S}$-valid, then $\gamma(\mathcal{E}_\infty) \to \gamma(\mathcal{E})$.*

The least fixedpoint of $\mathcal{F}^{\Lambda}$ is computable. Indeed, in Λ, any strictly monotone sequence of systems of equations cannot have more than $n+1$ distinct systems, where n is the number of variables. Therefore the sequence $\mathcal{E}_0 : \bot$, $\mathcal{E}_1 : \mathcal{F}^{\Lambda}(\mathcal{E}_0)$, $\mathcal{E}_2 : \mathcal{F}^{\Lambda}(\mathcal{E}_1)$, ... converges after at most $n+1$ steps.

Example (**generation of invariants**) We illustrate the technique on the transition system STABLE (Figure 11) by showing how to derive the linear invariant $\Box(y+z-2=0)$.

$$
\begin{array}{lll}
\mathcal{V} & : & \{y, z\} \\
\Theta & : & y = 0 \wedge z = 2 \\
\mathcal{T} & : & \{\tau_I, \tau\} \text{ where } \rho_\tau : \quad y' = y - 1 \wedge z' = y + 2z - 1
\end{array}
$$

Fig. 11. Transition system STABLE

The invariant $\Box(y+z-2=0)$ is found by computing the converging sequence $\bot \sqsubseteq \mathcal{E}_1 \sqsubseteq \mathcal{E}_2 \sqsubseteq \ldots$ We use the notation $\overline{x} = (y\ z)^{\mathrm{T}}$ and similarly $\overline{x}' = (y'\ z')^{\mathrm{T}}$. First, notice that we have the following implications:

$$
\begin{array}{lll}
\underbrace{y = 0 \ \wedge \ z = 2}_{\Theta} & \rightarrow & \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{U_\Theta} \begin{pmatrix} y \\ z \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ -2 \end{pmatrix}}_{\overline{v}_\Theta} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\
\underbrace{y' = y - 1 \ \wedge \ z' = y + 2z - 1}_{\rho_\tau} & \rightarrow & \underbrace{\begin{pmatrix} -1 & 0 & 1 & 0 \\ -1 & -2 & 0 & 1 \end{pmatrix}}_{A_\tau} \begin{pmatrix} y \\ z \\ y' \\ z' \end{pmatrix} + \underbrace{\begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{\overline{b}_\tau} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad .
\end{array}
$$

The system $\mathcal{E}_\Theta$ is the consequent of the first implication and $\mathcal{E}_\Theta^d$ (which is needed later) is determined by the system

$$(a) \quad X = (0\ 2) \quad \text{and} \quad \overline{y} = (1),$$

since a particular solution to $U_\Theta \overline{x} + v_\Theta = \overline{0}$ is $\overline{x} = (0\ 2)^{\mathrm{T}}$, and there are no nonzero solutions to the homogeneous system $U_\Theta \overline{x} = \overline{0}$.

The terms of the sequence $\mathcal{E}_0, \mathcal{E}_1, \ldots$ are computed in stages:[4]

- $\mathcal{E}_0 = \bot$
- $\mathcal{E}_1 = \mathcal{E}_\Theta \sqcup \tau(\bot) = \mathcal{E}_\Theta$.
- $\mathcal{E}_2 = \mathcal{E}_\Theta \sqcup \tau(\mathcal{E}_1)$. We have to solve the system

$$
\begin{cases} A_\tau(\overline{x}, \overline{x}') + \overline{b}_\tau = 0 \\ U_\Theta(\overline{x}) + \overline{v}_\Theta \quad = 0. \end{cases}
$$

[4] The idling transition τ_I can be ignored in the invariance case.

A solution of the system is $(\overline{x}_0\ \overline{x}'_0)^{\mathrm{T}} = (0\ 2\ -1\ 3)^{\mathrm{T}}$ and the homogeneous system of equations

$$\begin{cases} A_\tau(\overline{x}, \overline{x}') = 0 \\ U_\Theta(\overline{x}) = 0 \end{cases}$$

has only the trivial $(0\ 0\ 0\ 0)^{\mathrm{T}}$ solution. $\tau(\mathcal{E}_1)^d$ is therefore determined by

$$(b) \quad X = (-1\ 3) \quad \text{and} \quad \overline{y} = (1) \ .$$

Combining the systems defined by (a) and (b), which express $\mathcal{E}^d_\Theta$ and $\tau(\mathcal{E}_1)^d$, we get

$$\mathcal{E}^d_2 : \quad \begin{pmatrix} 0 & 2 \\ -1 & 3 \end{pmatrix} \overline{u} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} v = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \ .$$

The set of solutions of $\mathcal{E}^d_2$ is a 1-dimensional vector space with basis given by $\{(\underbrace{1\ 1}_{\overline{u}_1^{\mathrm{T}}}\ \underbrace{-2}_{v_1})^{\mathrm{T}}\}$ hence,

$$\mathcal{E}_2 : \quad \underbrace{(1\ 1)}_{U_2} \overline{x} - \underbrace{2}_{\overline{v}_2} = 0 \ .$$

– $\mathcal{E}_3 = \mathcal{E}_\Theta \sqcup \tau(\mathcal{E}_2)$. By performing calculations similar to the previous step, we arrive at a linear system identical to $\mathcal{E}_2$.

Hence, we have reached a fixedpoint, which is translated by the concretization function into the linear invariant $\Box(y + z - 2 = 0)$. ∎

7 Convex Polyhedra

The convex polyhedra abstraction was first studied as a tool in the analysis of programs in [CH78]. More recently, convex polyhedra have also been used in the analysis of linear hybrid systems [HRP94, HH95]. We first summarize the method (following [CH78]) and give an example of its use in the generation of invariants. Next we illustrate how the techniques can be extended to automatically generate polyhedral intermediate assertions for the verification of general safety formulas.

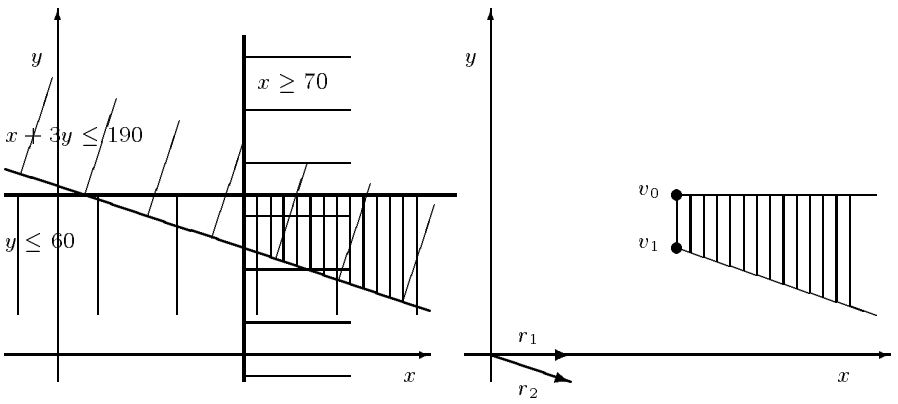
A closed convex polyhedron P is a formula $\bigwedge_i L_i$ where each L_i is a weak inequality, i.e.,

$$L_i = \sum_j a_{ij} x_j \geq b_i \ .$$

P can be viewed as the set of solutions of its system of linear constraints

$$P = \{\overline{x} \mid A\overline{x} \geq \overline{b}\} \ .$$

We say that $P = \{\overline{x} \mid A\overline{x} \leq \overline{b}\}$ *satisfies a linear constraint* $\overline{c} \cdot \overline{x} \leq d$ if $P = \{\overline{x} \mid A\overline{x} \leq \overline{b}, \overline{c} \cdot \overline{x} \leq d\}$, that is, $\overline{c} \cdot \overline{x} \leq d$ is linearly dependent on the constraints in $A\overline{x} \leq \overline{b}$.



(a) Constraint representation (b) Generator representation

Fig. 12. Representations of a polyhedron

Another representation of P is given by a system of generators: a finite set of *rays* R and a finite set of *vertices* V such that

$$P = \left\{ \sum_{v_i \in V} \lambda_i \cdot v_i + \sum_{r_j \in R} \mu_j \cdot r_j \;\middle|\; \lambda_i \geq 0,\ \mu_j \geq 0 \text{ and } \sum_i \lambda_i = 1 \right\}$$

where P is the convex closure of this system of generators. Both representations are used in the standard inclusion decision procedure.

Example (**polyhedra representations**) Consider the polyhedron shown in Figure 12. Its constraint representation is

$$\left\{ \begin{pmatrix} x \\ y \end{pmatrix} \;\middle|\; \begin{pmatrix} -1 & -3 \\ 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} -190 \\ 70 \\ -60 \end{pmatrix} \right\}.$$

The sets of rays and vertices R and V are

$$R = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 3 \\ -1 \end{pmatrix} \right\}, \qquad V = \left\{ \begin{pmatrix} 70 \\ 60 \end{pmatrix}, \begin{pmatrix} 70 \\ 40 \end{pmatrix} \right\},$$

which determine the generator representation

$$\left\{ \lambda_1 \begin{pmatrix} 70 \\ 60 \end{pmatrix} + \lambda_2 \begin{pmatrix} 70 \\ 40 \end{pmatrix} + \mu_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \mu_2 \begin{pmatrix} 3 \\ -1 \end{pmatrix} \;\middle|\; \begin{array}{l} \lambda_1, \lambda_2, \mu_1, \mu_2 \geq 0 \text{ and} \\ \lambda_1 + \lambda_2 = 1 \end{array} \right\}.$$

7.1 Invariants

The closed convex polyhedra Φ are an abstraction of the assertion language $\mathcal{A}$ with the concretization function given by the inclusion $\gamma : \Phi \mapsto \mathcal{A}$. The partial order, minimal and maximal elements are the subset relation, the empty set, and the total set which are mapped by γ into the implication relation, F and T respectively.

An approximation of $\mathcal{F}$ in Φ is

$$\mathcal{F}^{\Phi}(\varphi) \quad \stackrel{\triangle}{=} \quad \alpha(\Theta) \vee^{\Phi} \mathrm{SP}^{\Phi}(\alpha(\mathcal{T}), \varphi), \text{ where :}$$

- $\vee^{\Phi}$ is the convex hull operator: $P_1 \vee^{\Phi} P_2$ is the smallest closed convex polyhedron that includes P_1 and P_2. The convex hull can be computed using the generator representations: if (R_1, V_1) and (R_2, V_2) are the generator representations for P_1 and P_2 then $(R_1 \cup R_2, V_1 \cup V_2)$ is a generator representation for $P_1 \vee^{\Phi} P_2$.
- $\mathrm{SP}^{\Phi}(\alpha(\mathcal{T}), \varphi) \stackrel{\triangle}{=} \bigvee^{\Phi}_{\tau} \mathrm{SP}(\alpha(\tau), \varphi)$, where $\mathrm{SP}(\alpha(\tau), \varphi)(V) \stackrel{\triangle}{=} \exists V^p \cdot (\alpha(\rho_\tau)(V^p, V) \wedge \varphi(V^p))$. Notice that $\exists x$ is an operator on Φ. Indeed, it can be shown that $\exists x.P = P \vee^{\Phi} (\bigwedge_{y \neq x}(y = 0))$.
- $\alpha(\varphi) \in \Phi$ is such that $\varphi \rightarrow \alpha(\varphi)$. For a given φ we can compute such an $\alpha(\varphi)$ by first rewriting $\varphi = c_1 \wedge \ldots \wedge c_k$ and then taking $\alpha(\varphi) \stackrel{\triangle}{=} \bigwedge\{c_i \mid c_i \text{ is linear}\}$.

The sequence

$$P_0 : \mathrm{F}, \quad P_1 : \mathcal{F}^{\Phi}(P_0), \quad P_2 : \mathcal{F}^{\Phi}(P_1), \ \ldots$$

does not necessarily converge in finitely many steps (it actually might not even converge in Φ). However it is possible to achieve convergence using a *widening* operator [CH78]. This is an operator $\bigtriangledown : \Phi \times \Phi \mapsto \Phi$ such that

(W1) for any $P_1, P_2 \in \Phi$, $P_1 \vee^{\Phi} P_2 \rightarrow P_1 \bigtriangledown P_2$,
(W2) for any $P_0 \rightarrow P_1 \rightarrow \ldots$ in Φ, the sequence P'_0, P'_1, $\ldots$ defined by $P'_0 = P_0$ and $P'_{i+1} = P'_i \bigtriangledown P_{i+1}$ converges in a finite number of steps.

Several widening operators appear in the literature. The simplest is defined by

$$P_1 \bigtriangledown P_2 \quad \stackrel{\triangle}{=} \quad \begin{cases} P_2 & \text{if } P_1 = \emptyset, \\ \text{the polyhedron defined by the} & \\ \qquad \text{constraints of } P_1 \text{ satisfied by } P_2 & \text{otherwise} \end{cases}$$

The widening operator can be used to obtain the sequence

$$P'_0 : \mathrm{F}, \quad P'_1 : P'_0 \bigtriangledown \mathcal{F}^{\Phi}(P'_0), \quad P'_2 : P'_1 \bigtriangledown \mathcal{F}^{\Phi}(P'_1), \ \ldots.$$

which, by (W2), converges after finitely many iterations. The above sequence conservatively approximates $P_0, P_1, \ldots$ and therefore its limit is a solution of $\mathcal{F}^{\Phi}(P) \rightarrow P$.

Most widening operators have the property that the constraints of $P_1 \bigtriangledown P_2$ are a subset of the constraints of P_1 (if $P_1 \neq \emptyset$). Thus the constraints of the limit polyhedron are a subset of the constraints of the polyhedron to which the first

widening has been applied. For this reason, forward propagation with widening usually converges to a better solution if we start with a few iterations without widening. We will use this observation in our examples.

Example (generation of invariants) Consider program LOOP (Figure 13). Using polyhedra abstraction we can generate the invariant $0 \leq y \leq N$.

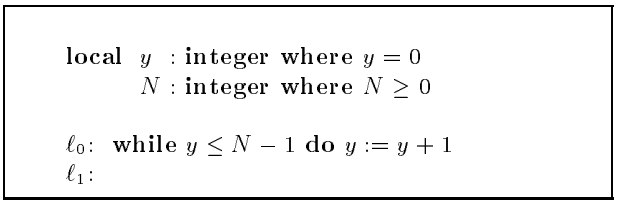


Fig. 13. Program LOOP

To illustrate the method and the need for a widening operator, we present in Figure 14 a few terms of the infinitely increasing sequence

$$P_1 : \mathcal{F}^{\Phi}(\bot), \quad P_2 : P_1 \vee^{\Phi} \mathcal{F}^{\Phi}(P_1), \quad P_3 : P_2 \vee^{\Phi} \mathcal{F}^{\Phi}(P_2), \ldots$$

as well as a few terms of the sequence

$$P'_1 : \mathcal{F}^{\Phi}(\bot), \quad P'_2 : P'_1 \vee^{\Phi} \mathcal{F}^{\Phi}(P'_1), \quad P'_3 : P'_2 \bigtriangledown \mathcal{F}^{\Phi}(P'_2), \ldots$$

that converges in finitely many steps. A polyhedral forward propagation operator $\mathcal{F}^{\Phi}$ for LOOP can be defined as

$$\begin{array}{rcl} \mathcal{F}^{\Phi} & \triangleq & \alpha(\Theta) \vee^{\Phi} \mathrm{SP}^{\Phi}(\alpha(\mathcal{T}), \varphi)) \\ \alpha(\Theta) & \triangleq & y = 0 \wedge N \geq 0 \\ \alpha(\tau_{\ell_0,\mathrm{T}}) & \triangleq & y \leq N - 1 \wedge y' = y + 1 \\ \alpha(\tau_{\ell_0,\mathrm{F}}) & \triangleq & y \geq N \end{array}$$

For simplicity we ignore the two label variables ℓ_0 and ℓ_1. (Taking them into account generates 4-dimensional polyhedra.)

Partitioning the state space into sets that have the same location we could generate a stronger invariant for the transition system LOOP:

$$\Box\,[(\ell_0 \rightarrow 0 \leq y \leq N) \ \wedge \ (\ell_1 \rightarrow 0 \leq y = N)] \ \blacksquare$$

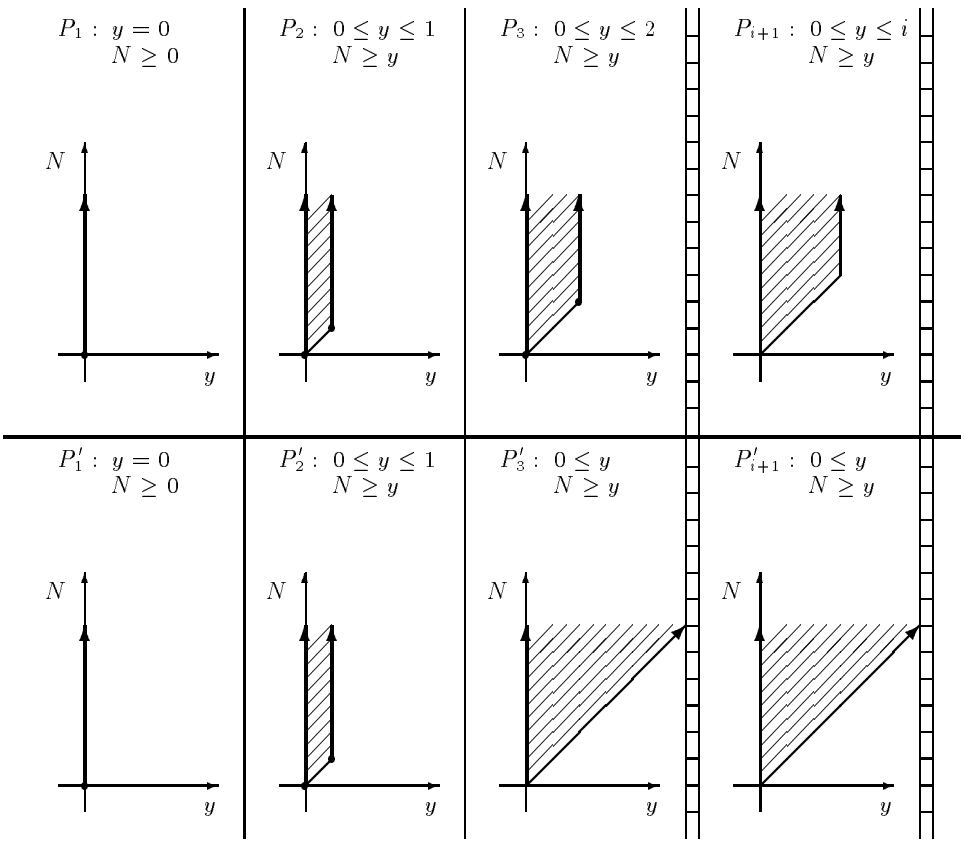


Fig. 14. The sequences $P_0, P_1, \ldots$ and $P'_0.P'_1, \ldots$

7.2 General Safety Formulas

Let G be an assertion graph and $\underline{G}$ a strengthened graph labeled by $\underline{\lambda} : N \mapsto \Phi \subseteq \mathcal{A}$ such that

$$\underline{\lambda}(n) \rightarrow \lambda(n) \ .$$

Let $G^e \supset \underline{G}$ be an extension graph of $\underline{G}$ with labels in Φ.

An approximation of $\mathcal{F}_{N^e}$ in Φ is given by

$$\mathcal{F}^{\Phi}_n(P_N) \quad \stackrel{\triangle}{=} \quad [\alpha(\Theta)_{n \in N_0} \vee^{\Phi} \bigvee^{\Phi}_{(m,n) \in E} \mathrm{SP}^{\Phi}(\mathcal{T}, P_m)] \ \wedge \ \underline{\lambda}(n)$$

where $\alpha(\Theta)_{n \in N_0}$ is defined to be $\alpha(\Theta)$ if $n \in N_0$ and $\bot$ otherwise. The other notations are the same as in Section 7.1. This operator has all the desired properties, except that the ascending sequence

$$\underbrace{(\emptyset)_{n \in N^e}}_{P_{N^e,0}}, \quad \underbrace{\mathcal{F}^{\Phi}_{N^e}(P_{N,0})}_{P_{N^e,1}}, \quad \underbrace{\mathcal{F}^{\Phi}_{N^e}(P_{N,1})}_{P_{N^e,2}}, \ \ldots$$

does not necessarily converge. We could use again a widening operator to speed up the convergence process. However, the widening operator of Section 7.1 does not guarantee that a solution φ_{N^e} has the property that $\forall n \in N^e \varphi_n \rightarrow \underline{\lambda}(n)$. This motivates the following definition:

A *bounded widening operator* is an operator $\triangledown : (\varphi_1, \varphi_2, \psi) \mapsto \varphi_1 \overset{\psi}{\triangledown} \varphi_2$ where

(W1′) for any $P_1, P_2, P \in \Phi$. $P_1 \vee^{\Phi} P_2 \rightarrow P_1 \overset{P}{\triangledown} P_2$
(W2′) for any $P_0 \rightarrow P_1 \rightarrow \ldots$ in Φ and $P \in \Phi$, the sequence $P'_0, P'_1, \ldots$ defined by $P'_0 = P_0$ and $P'_{i+1} = P'_i \overset{P}{\triangledown} P_{i+1}$ converges in finitely many steps.
(W3) for any $P_1, P_2, P \in \Phi$, if $P_1 \rightarrow P$ and $P_2 \rightarrow P$ then $P_1 \overset{P}{\triangledown} P_2 \rightarrow P$.

An example of such a bounded operator is

$$P_1 \overset{P}{\triangledown} P_2 = \begin{cases} P_2 & \text{if } P_1 = \emptyset \\ \text{the polyhedron defined by} & \\ \quad \text{the constraints of } P_1 \text{ satisfied by } P_2 \text{ and} & \\ \quad \text{the constraints of } P \text{ satisfied by } P_1, P_2 & \text{otherwise} \end{cases}$$

If the forward operator is bounded by P, then the bounded widening operator can be used to obtain converging sequences bounded by P. For any $n \in N^e$, the sequence

$$P'_{n,0} : \text{F}, \quad P'_{n,1} : P'_{n,0} \overset{\underline{\lambda}(n)}{\triangledown} \overline{\mathcal{F}}_n(P'_{N,0}), \quad \ldots$$

converges in finitely many steps to a limit that is bounded by $\underline{\lambda}(n)$.

Example (generation of intermediate assertions for rule SAFE) Consider again program LOOP (Figure 13). We want to prove that y takes on all nonnegative integer values in order, that is, if $0 \leq u_1 \leq u_2$ and y has value u_2 then previously y had value u_1. This property is expressed in temporal logic as

$$p_{caus} : \quad \Box(y = u_2 \rightarrow \diamondsuit\!\!\!-\, y = u_1) \ .$$

The assertion $0 \leq u_1 \leq u_2$ will be used as a new axiom. An assertion graph G for p_{caus} and its extension G^e are presented in Figure 15.

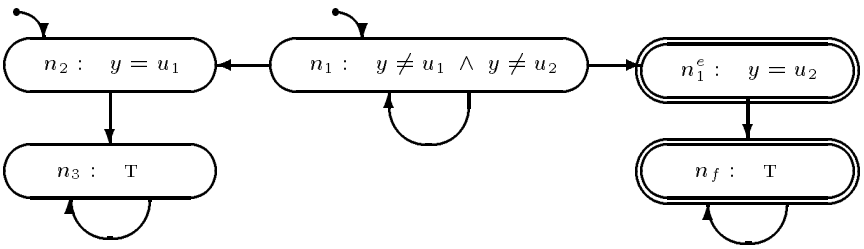


Fig. 15. Graphs $\underline{G}$ and $\underline{G^e}$ for the property p_{caus}

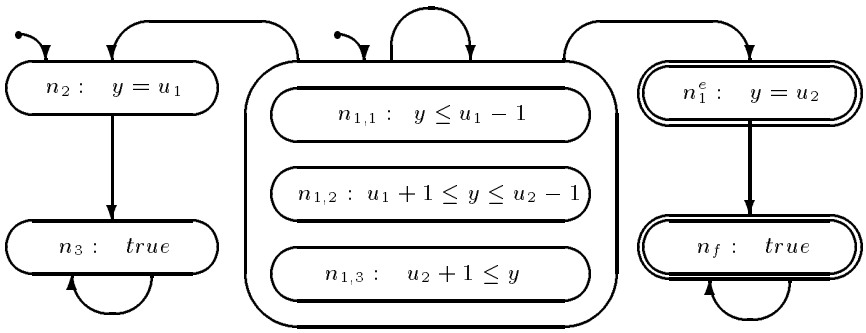


Fig. 16. Graphs $\underline{G}$ and $\underline{G}^e$ for the property p_{caus}

Table 4. Forward propagation for program LOOP and formula p_{caus} where $c : N \geq y$

	Θ iteration	exact iteration	$\triangledown$ iteration
$n_{1,1}$	$y = 0 \leq u_1 - 1 \; \wedge \; c$	$0 \leq y \leq 1 \leq u_1 - 1 \; \wedge \; c$	$0 \leq y \leq u_1 - 1 \; \wedge \; c$
$n_{1,2}$	F	F	F
$n_{1,3}$	F	F	F
n_2	$y = u_1 = 0 \; \wedge \; c$	$0 \leq y = u_1 \leq 1 \; \wedge \; c$	$0 \leq y = u_1 \; \wedge \; c$
n_3	F	$u_1 = 0 \leq y \leq 1 \; \wedge \; c$	$0 \leq u_1 \leq y \; \wedge \; c$
n_1^e	F	F	F
n^f	F	F	F

To conservatively approximate G we have to strengthen $\lambda(n_1)$ and $\lambda(n_2)$ and weaken $\lambda(n_1^e)$ in the polyhedra domain. In our example, all labels except for $\lambda(n_1)$ are closed convex polyhedra. The labeling of node n_1 is $y \neq u_1 \wedge y \neq u_2$. The negated equalities can be transformed into pairs of inequalities; furthermore, since the variables are integers they can be transformed into weak inequalities, e.g., $y \neq u_1 \;\; \leftrightarrow \;\; y \leq u_1 - 1 \;\; \wedge \;\; y \geq u_1 + 1$. Rewriting this expression into disjunctive normal form produces polyhedral disjuncts, so no further weakening is necessary. A node labeled by a disjunction can always be split into a cluster of nodes, one for each disjunct. Thus we get the graph in Figure 16. The assertion $0 \leq u_1 \leq u_2$ was used to simplify the labels of the generated nodes, and nodes with label F are not shown.

The forward propagation proceeds as shown in Table 4. The sequence converges after three iterations with empty polyhedra corresponding to the nodes n_1^e and n_f, which proves that p_{caus} is $\mathcal{S}$-valid. ⌟

8 Set-based Analysis

Set-based analysis approximates the state space by associating each system variable with an indexed collection of sets. Each set contains the possible values of

a variable in a class of states, e.g. at a given location. Dependencies between variables are ignored, unlike the linear and polyhedral constraints. However, more general constraint solvers apply to this setting allowing easy generation of constraints over arbitrary data types. A comprehensive treatment of set-based analysis can be found in [Hei92], where it is used, for instance, in compile-time (binding-time) analysis of imperative, functional and logic programming languages. STeP currently applies only the basic ideas to SPL, generating a subset of the properties expressible using set constraints. We now illustrate how set constraints may generate general safety formulas that cannot be obtained by linear or polyhedral methods.

8.1 Invariance

Given the system variables $\mathcal{V} = \{x_1, \ldots, x_n\}$, let $\mathbf{X}$ be a tuple of sets $(X_1, \ldots, X_n)$ corresponding to $\mathcal{V}$. We refer to a family of such tuples by $\{\mathbf{X}_i\}_{i \in I}$, where I is an index set.

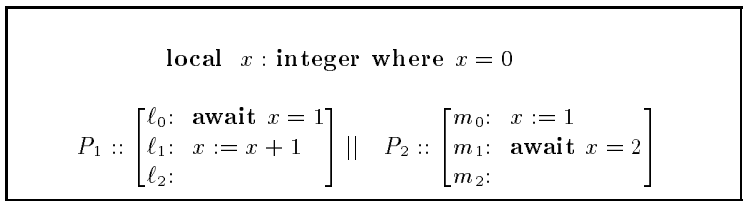


Fig. 17. Program MONADIC

Example (indexed sets) For the trivial program MONADIC, shown in Figure 17, $\mathcal{V} = \{x\}$, and we take $I = \{\ell_0, \ell_1, \ell_2, m_0, m_1, m_2\}$, which gives the family X_{ℓ_0}, X_{ℓ_1}, X_{ℓ_2}, X_{m_0}, X_{m_1}, X_{m_2} of six singleton tuples. ▟

A transition takes an indexed family and produces another according to the effect of the transition. Tuples that are not affected by the transition are set to be empty. Hence we can define

$$\mathcal{F}^{\mathcal{M}}(\{\mathbf{X}_i\}_{i \in I}) \triangleq \Theta_I \sqcup \bigsqcup_{\tau \in \mathcal{T}} \mathrm{SP}(\tau, \{\mathbf{X}_i\}_{i \in I})$$

where $\sqcup$ is defined as

$$\{(X_1, \ldots, X_n)_i\}_{i \in I} \sqcup \{(Y_1, \ldots, Y_n)_i\}_{i \in I} \triangleq \{(X_1 \cup Y_1, \ldots, X_n \cup Y_n)_i\}_{i \in I},$$

and Θ_I is an indexed family, where tuples for the initial locations contain the initial values and all other tuples have empty sets.

Example For MONADIC, Θ_I is $\{\{0\}_{\ell_0}, \emptyset_{\ell_1}, \emptyset_{\ell_2}, \{0\}_{m_0}, \emptyset_{m_1}, \emptyset_{m_2}\}$, and

$\mathrm{SP}(\tau_{\ell_0}, \{X_i\}_{i \in I}) : \{\emptyset_{\ell_0}, (\{1\} \cap X_{\ell_0})_{\ell_1}, \emptyset_{\ell_2}, \emptyset_{m_0}, \emptyset_{m_1}, \emptyset_{m_2}\}$
$\mathrm{SP}(\tau_{\ell_1}, \{X_i\}_{i \in I}) : \{\emptyset_{\ell_0}, \emptyset_{\ell_1}, (X_{\ell_1}{+}1)_{\ell_2}, (X_{\ell_1}{+}1)_{m_0}, (X_{\ell_1}{+}1)_{m_1}, (X_{\ell_1}{+}1)_{m_2}\}$
$\mathrm{SP}(\tau_{m_0}, \{X_i\}_{i \in I}) : \{\{1\}_{\ell_0}, \{1\}_{\ell_1}, \{1\}_{\ell_2}, \emptyset_{m_0}, \{1\}_{m_1}, \emptyset_{m_2}\}$
$\mathrm{SP}(\tau_{m_1}, \{X_i\}_{i \in I}) : \{\emptyset_{\ell_0}, \emptyset_{\ell_1}, \emptyset_{\ell_2}, \emptyset_{m_0}, \emptyset_{m_1}, (\{2\} \cap X_{m_1})_{m_2}\}$

where $X_{\ell_1}{+}1$ is shorthand for $\{x+1 \mid x \in X_{\ell_1}\}$; thus $+ : \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$ is lifted to its power-set version $+ : 2^{\mathcal{Z}} \times 2^{\mathcal{Z}} \to 2^{\mathcal{Z}}$. Combining the sets with $\sqcup$ we get

$$\mathcal{F}^{\mathcal{M}}(\{X_i\}_{i\in I}) = \left\{ \begin{array}{ll} (\{0,1\})_{\ell_0}, & (\{0\} \cup (X_{\ell_1}{+}1))_{m_0}, \\ (\{1\})_{\ell_1}, & (\{1\} \cup (X_{\ell_1}{+}1))_{m_1}, \\ (\{1\} \cup (X_{\ell_1}{+}1))_{\ell_2}, & ((\{2\} \cap X_{m_1}) \cup (X_{\ell_1}{+}1))_{m_2} \end{array} \right\} \ . \quad \blacksquare$$

A crucial observation in set-based analysis is that the least fixedpoint of $\mathcal{F}^{\mathcal{M}}$ can be calculated directly by collecting its effect as constraints on the indexed tuples. The smallest solution to the constraints is the least fixedpoint.

Example (constraints) For program MONADIC we generate the set constraints

$$\begin{array}{lll} X_{\ell_0} \supseteq \{0,1\}, & X_{\ell_1} \supseteq \{1\}, & X_{\ell_2} \supseteq \{1\} \cup (X_{\ell_1}{+}1), \\ X_{m_0} \supseteq \{0\} \cup (X_{\ell_1}{+}1), & X_{m_1} \supseteq \{1\} \cup (X_{\ell_1}{+}1), & X_{m_2} \supseteq (\{2\} \cap X_{m_1}) \cup (X_{\ell_1}{+}1) \ . \end{array}$$

The least solution of these constraints is:

$$\begin{array}{lll} X_{\ell_0} = \{0,1\}, & X_{\ell_1} = \{1\}, & X_{\ell_2} = \{1,2\}, \\ X_{m_0} = \{0,2\}, & X_{m_1} = \{1,2\}, & X_{m_2} = \{2\} \ . \end{array}$$

Constraint-solving methods based on tree automata or set constraints [HJ91] are guaranteed to generate this least solution algorithmically. $\blacksquare$

Each indexed tuple is associated with a first-order constraint by mapping $\mathbf{X}_i$ to $\gamma(\mathbf{X}_i) = \varphi_i \to M_1(x_1) \wedge \ldots \wedge M_n(x_n)$, where the M_k are monadic predicates. $M_k(x)$ is true iff x is a member of each $X_{k,i}$ that satisfies the constraints.

Example (concretization) The first-order constraints for MONADIC are

$$\begin{array}{lll} \gamma(X_{\ell_0})\colon\ \ell_0 \to 0 \le x \le 1, & \gamma(X_{\ell_1})\colon\ \ell_1 \to x = 1, & \gamma(X_{\ell_2})\colon\ \ell_2 \to 1 \le x \le 2, \\ \gamma(X_{m_0})\colon m_0 \to x{=}0 \vee x{=}2, & \gamma(X_{m_1})\colon m_1 \to 1 \le x \le 2, & \gamma(X_{m_2})\colon m_2 \to x = 2, \end{array}$$

and the invariant extracted is $\Box \bigwedge_{i\in I} \gamma(X_i)$. $\blacksquare$

8.2 General Safety

The ideas involved in generating general safety formulas are similar to those presented for invariance. The following example illustrates the potential of having powerful constraint solvers at our disposal.

Example (parity) Suppose we want to examine what happens between the point at which program BAKERY (Figure 3) sets y_1 to 1 and when both y_1 and y_2 are 0 again. A template of the events under examination is given in Figure 18.

With each node in G we associate a pair of sets, (Y_1, Y_2) corresponding to the variables y_1 and y_2, indexed by the node identifier, i.e., we have the tuples of sets $(Y_1, Y_2)_{n_1}, \ldots, (Y_1, Y_2)_{n_6}$. The propagation operator $\mathcal{F}^{\mathcal{M}}_N$ is defined for each node n following the general schema

$$\mathcal{F}^{\mathcal{M}}_n\left(\{(Y_1, Y_2)_n\}_{n\in N}\right) \quad \triangleq \quad \left[\Theta_{n\in N_0} \sqcup \bigsqcup_{\tau\in\mathcal{T}} \bigsqcup_{(m,n)\in E} \mathrm{SP}(\tau, (Y_1, Y_2)_m)\right] \ \sqcap \ \lambda(n)$$

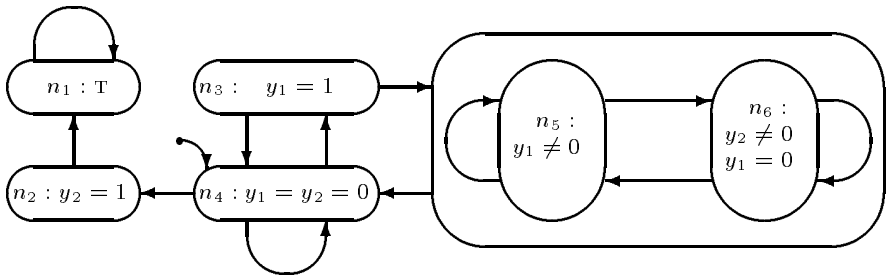


Fig. 18. Assertion graph G for BAKERY

where $\lambda(n)$ is a pair of sets constrained by the node labelings:

$$\begin{array}{lll} \lambda(n_1) = (\top, \top), & \lambda(n_2) = (\top, \{1\}), & \lambda(n_3) = (\{1\}, \top), \\ \lambda(n_4) = (\{0\}, \{0\}), & \lambda(n_5) = (\top \setminus \{0\}, \top), & \lambda(n_6) = (\{0\}, \top \setminus \{0\}) \end{array}$$

where $\top$ is the set of all possible values, and $\Theta_{n \in N_0} = (\{0\}, \{0\})$ for $n = n_4$ and $(\emptyset, \emptyset)$ otherwise. The operation $\sqcap$ is dual to $\sqcup$ and is characterized by $(X_1, Y_1) \sqcap (X_2, Y_2) = (X_1 \cap X_2, Y_1 \cap Y_2)$. In this case the operator SP takes a pair $(Y_1, Y_2)_n$ and produces a pair (Y_1', Y_2') corresponding to the effect of the transition. For instance,

$$\mathrm{SP}(\tau_{\ell_1}, (Y_1, Y_2)_{n_4}) = (Y_2 + 1, Y_2) \quad .$$

We have also applied the auxiliary invariant $\Box(y_2 \neq 0 \rightarrow m_2 \vee m_3 \vee m_4)$ to infer that transition τ_{m_1} is disabled at node n_6, hence $\mathrm{SP}(\tau_{m_1}, (Y_1, Y_2)_{n_6}) = (\emptyset, \emptyset)$.

From the fixedpoint equation $\mathcal{F}_N^{\mathcal{M}}(\{(Y_1, Y_2)_n\}_{n \in N}) = \{(Y_1, Y_2)_n\}_{n \in N}$ we extract, as in the invariance analysis, a set of constraints, whose minimal solution is the fixedpoint. For $n_3, \ldots, n_6$ the constraints are:

$$\begin{array}{llllll} \{1\} \subseteq Y_{1,n_3} & \{0\} \subseteq Y_{1,n_4} & Y_{1,n_3} \cup (Y_{2,n_5} \cup Y_{2,n_6}) + 1 \subseteq Y_{1,n_5} & & \{0\} \subseteq Y_{1,n_6} \\ \{0\} \subseteq Y_{2,n_3} & \{0\} \subseteq Y_{2,n_4} & Y_{2,n_3} \cup (Y_{1,n_3} \cup Y_{1,n_5}) + 1 \subseteq Y_{2,n_5} & Y_{2,n_5} \setminus \{0\} \subseteq Y_{2,n_6} & \end{array}$$

whose minimal solution is

$$\begin{array}{ll} (Y_1, Y_2)_{n_3} = (\{1\}, \{0\}) & (Y_1, Y_2)_{n_5} = (\{n \in \mathcal{N} \mid odd(n)\}, \{n \in \mathcal{N} \mid even(n)\}) \\ (Y_1, Y_2)_{n_4} = (\{0\}, \{0\}) & (Y_1, Y_2)_{n_6} = (\{0\}, \{n \in \mathcal{N} \mid even(n) \wedge n > 0\}) \end{array}$$

where $\mathcal{N}$ is the set of nonnegative integers. We have thus extracted the information that when y_1 is set to 1 (represented by n_3), y_2 remains even and y_1 either odd or 0 until both are simultaneously 0 again (represented by n_4). ▟

Acknowledgements We thank Henny Sipma and Tomás Uribe for their many helpful comments and suggestions.

References

[CC77] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In 4^{th} *ACM Symp. Princ. of Prog. Lang.*, pages 238–252. ACM Press, 1977.

[CGL92] E.M. Clarke, O. Grumberg, and D.E. Long. Model checking and abstraction. In 19^{th} *ACM Symp. Princ. of Prog. Lang.*, pages 343–354, 1992.

[CH78] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among the variables of a program. In 5^{th} *ACM Symp. Princ. of Prog. Lang.*, pages 84–97, Jan. 1978.

[DGG94] D.R. Dams, O. Grumberg, and R. Gerth. Abstract interpretation of reactive systems: Abstractions preserving $\forall$CTL*, $\exists$ECTL*, CTL*. In *IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET 94)*, pages 573–592, June 1994.

[Gra91] P. Granger. Static analysis of linear congruence equalities among variables of a program. In *TAPSOFT 91, Vol.1: Colloq. on Trees in Algebra and Programming (CAAP '91)*, LNCS, pages 169–192. Springer-Verlag, April 1991.

[GW75] S. M. German and B. Wegbreit. A Synthesizer of Inductive Assertions. *IEEE transactions on Software Engineering*, 1(1):68–75, March 1975.

[Har84] D. Harel. Statecharts: A visual approach to complex systems. Technical Report CS84-05, Dept. of Applied Mathematics, Weizmann Institute of Science, 1984.

[Hei92] N. Heintze. *Set Based Program Analysis.* PhD thesis, Carnegie Mellon University, 1992.

[HH95] T.A. Henzinger and P.-H. Ho. Algorithmic analysis of nonlinear hybrid systems. In *Proc. 7^{th} Intl. Conference on Computer Aided Verification*, LNCS, 1995.

[HJ91] N. Heintze and J. Jaffar. A decision procedure for a class of Herbrand set constraints. Technical Report CMU-CS-91-110, Carnegie Mellon University, Feb. 1991. Abstract appears in *Proceedings of the 5^{th} Annual IEEE Symposium on Logic in Computer Science*, 1990.

[HRP94] N. Halbwachs, P. Raymond, and Y.-E. Proy. Verification of linear hybrid systems by means of convex approximations. In 1^{st} *Intl. Static Analysis Symp.*, vol. 864 of *LNCS*, pages 223–237. Springer-Verlag, Sept. 1994.

[Kar76] M. Karr. Affine relationships among variables of a program. *Acta Informatica*, 6:133–151, 1976.

[KM76] S. Katz and Z. Manna. Logical analysis of programs. *Communications of the ACM*, 19(4):188–206, April 1976.

[Lau73] K. Lautenbach. Exacte Bedingungen der Lebendigkeit für eine Klasse von Petri-Netzen. *St. Augustin, GMD Bonn*, 82, 1973.

[MAB+94] Z. Manna, A. Anuchitanukul, N. Bjørner, A. Browne, E. Chang, M. Colón, L. de Alfaro, H. Devarajan, H.B. Sipma, and T.E. Uribe. STeP: The Stanford temporal prover. Technical Report STAN-CS-TR-94-1518, Computer Science Department, Stanford University, July 1994.

[MP95] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.

[Rei85] W. Reisig. *Petri Nets: An Introduction*, vol. 4 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, Berlin, 1985.

This article was processed using the LaTeX macro package with LLNCS style