Verification by Finitary Abstraction

A. Pnueli
Weizmann Institute of Sciences and
Universite Joseph Fourier, Grenoble

Fourth International Spin Workshop (SPIN'98)
Paris 2.11.98

Joint work with:

Y. Kesten

Ben Gurion University

The Common Wisdom

To verify a reactive system S,

- If it is finite state, model check it.

- Otherwise, prove it by temporal deduction, using a temporal deductive system such as [MP] or TLA, supported by theorem provers, such as STeP, TLP, or PVS.

Often, both approaches to verification can be simplified by using abstraction.

AAV: Abstraction Aided Verification

An Obvious idea:

- Abstract system S into S_A – a simpler system, but admitting more behaviors.
- Verify property for the abstracted system S_A.
- Conclude that property holds for the concrete system.

Approach is particularly impressive when abstracting an infinite-state system into a finite-state one.

Can Abstraction Replace Deduction?

An intriguing question is whether abstraction can completely replace the need for Temporal Deduction.

That is, is it the case that, for every (possibly infinite-state) system $\mathcal{K}$ and a property ψ valid for $\mathcal{K}$, we can find an abstraction α such that $\mathcal{K}^\alpha$ is finite-state, ψ^α is propositional, and $\mathcal{K}^\alpha \models \psi^\alpha$?

This will relegate temporal reasoning to the regime of automatic model checking techniques.

Possible Advantages:

- First-order temporal reasoning is more difficult to master than first-order state reasoning.
- People (engineers) find it easier to program, than to write logical formulas. An abstraction is easier to develop (program) than an invariant assertion.

Technically

Define the methodology of Verification by Finitary Abstraction (VFA) as follows:

To prove $\mathcal{K} \models \psi$,

- Abstract $\mathcal{K}$ into a finite-state system $\mathcal{K}^\alpha$ and the specification ψ into a propositional LTL formula ψ^α.
- Model check $\mathcal{K}^\alpha \models \psi^\alpha$.

The question considered in this talk is whether we can find instantiations of this general methodology which are sound and (relatively) complete.

The Computational Model: Fair Kripke's Structures

An FKS $\mathcal{K} = \langle V, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$ consists of:

- V – A finite set of state variables. A state s is an interpretation of V. Σ – the set of all states.

- Θ – An initial condition. A satisfiable assertion that characterizes the initial states.

- ρ – A transition relation. An assertion $\rho(V, V')$, referring to both unprimed (current) and primed (next) versions of the state variables. For example, $x' = x + 1$ corresponds to the assignment $x := x + 1$.

- $\mathcal{J} = \{J_1, \ldots, J_k\}$ A set of justice (weak fairness) requirements. Ensure that a computation has infinitely many J_i-states for each J_i, $i = 1, \ldots, k$.

- $\mathcal{C} = \{\langle p_1, q_1 \rangle, \ldots \langle p_n, q_n \rangle\}$ A set of compassion (strong fairness) requirements. Infinitely many p_i-states imply infinitely many q_i-states.

Computations

Let $\mathcal{K}$ be an FKS for which the above components have been identified. The state s' is defined to be a $\mathcal{K}$-successor of state s if

$$\langle s, s' \rangle \models \rho_{\mathcal{K}}(V, V').$$

We define a computation of $\mathcal{K}$ to be an infinite sequence of states

$$\sigma : s_0, s_1, s_2, ...,$$

satisfying the following requirements:

- Initiality: s_0 is initial, i.e., $s_0 \models \Theta_{\mathcal{K}}$.
- Consecution: For each $j = 0, 1, ...$, the state s_{j+1} is a $\mathcal{K}$-successor of the state s_j.
- Justice: For each $J \in \mathcal{J}$, σ contains infinitely many J-states
- Compassion: For each $\langle p, q \rangle \in \mathcal{C}$, if σ contains infinitely many p-states, it must also contain infinitely many q-states.

Example: Program ANY-Y

Consider the program

$$x,\ y : \textbf{integer initially } x = y = 0$$

$$\left[\begin{array}{l} \ell_0 : \quad \textbf{while } x = 0 \textbf{ do} \\ \qquad [\ell_1 : \ y := y + 1] \\ \ell_2 : \end{array}\right] \quad \| \quad \left[\begin{array}{l} m_0 : \quad x := 1 \\ m_1 : \end{array}\right]$$

$$- \quad P_1 \quad - \qquad\qquad - \quad P_2 \quad -$$

The Corresponding FKS

- State Variables V: $\left(\begin{array}{lll} x, y & : & \text{integer} \\ \pi_1 & : & \{\ell_0, \ell_1, \ell_2\} \\ \pi_2 & : & \{m_0, m_1\} \end{array}\right)$.

- Initial condition:

$$\Theta: \ \pi_1 = \ell_0 \ \wedge \ \pi_2 = m_0 \ \wedge \ x = y = 0.$$

- Transition Relation: ρ: $\rho_I \vee \rho_{\ell_0} \vee \rho_{\ell_1} \vee \rho_{m_0}$, with appropriate disjunct for each statement. For example, the disjuncts ρ_I and ρ_{ℓ_0} are

$$\rho_I: \quad {\pi_1}' = \pi_1 \wedge {\pi_2}' = \pi_2 \wedge x' = x \wedge y' = y$$

$$\rho_{\ell_0}: \quad \pi_1 = \ell_0 \quad \wedge \quad \left(\begin{array}{c} x = 0 \ \wedge \ {\pi_1}' = \ell_1 \\ \vee \\ x \neq 0 \ \wedge \ {\pi_1}' = \ell_2 \end{array}\right)$$

$$\wedge \quad {\pi_2}' = \pi_2 \wedge x' = x \wedge y' = y$$

- Justice set: $\mathcal{J}$: $\{\neg at\_\ell_0, \neg at\_\ell_1, \neg at\_m_0\}$.

- Compassion set: $\mathcal{C}$: $\emptyset$.

Abstraction

Based on the notion of abstract interpretation [CC77].

Let Σ denote the set of states of an FKS $\mathcal{K}$ – the concrete states. Let $\alpha : \Sigma \mapsto \Sigma_A$ be a mapping of concrete states into abstract states.

We formulate the strategy of Verification by Abstraction:

- Define an abstraction mapping α to abstract the concrete FKS $\mathcal{K}$ into an abstract FKS $\mathcal{K}_A^\alpha$.
- Abstract the temporal property ψ into an abstract property ψ^α.
- Verify $\mathcal{K}_A^\alpha \models \psi^\alpha$.
- Conclude $\mathcal{K} \models \psi$.

The question is how to define the abstractions $\mathcal{K}_A^\alpha$ and ψ^α such that $\mathcal{K}_A^\alpha \models \psi^\alpha$ implies $\mathcal{K} \models \psi$?

That is, how to ensure that the abstraction is sound.

Example: Program ANY-Y

Consider the program

$$x,\ y: \textbf{ integer initially } x = y = 0$$

$$\left[\begin{array}{ll} \ell_0: & \textbf{while } x = 0 \textbf{ do} \\ & \quad [\ell_1:\ y := y + 1] \\ \ell_2: & \end{array}\right] \quad \| \quad \left[\begin{array}{ll} m_0: & x := 1 \\ m_1: & \end{array}\right]$$

$$- \quad P_1 \quad - \qquad\qquad\qquad - \quad P_2 \quad -$$

Assume we wish to verify the property $\square\ (y \geq 0)$ for system ANY-Y.

The abstract variable Y, ranges over the abstract domain $\{neg, zero, pos\}$. The abstraction function α maps the domain of y into the domain of Y as follows:

$$\alpha(y): \quad \left(\begin{array}{llll} \textbf{if} & y < 0 & \textbf{then} & neg \\ \textbf{else-if} & y = 0 & \textbf{then} & zero \\ \textbf{else} & & & pos \end{array}\right)$$

The Abstracted Version

With the mapping α, we can obtain the abstract version of ANY-Y, called ANY-Y$^\alpha$:

$$\begin{array}{lll} x: & \textbf{integer} & \textbf{initially } x = 0 \\ Y: & \{neg, zero, pos\} & \textbf{initially } Y = zero \end{array}$$

$$P_1 :: \left[\begin{array}{l} \ell_0: \quad \textbf{while } x = 0 \textbf{ do} \\ \qquad [\ell_1 : \ Y := \left(\begin{array}{ll} \textbf{if} & Y = neg \\ \textbf{then} & \{neg, zero\} \\ \textbf{else} & pos \end{array} \right)] \\ \ell_2 : \end{array} \right]$$

$$\|$$

$$P_2 :: \left[\begin{array}{ll} m_0 : & x := 1 \\ m_1 : & \end{array} \right]$$

The original invariance property ψ: $\square\,(y \geq 0)$, is abstracted into:

$$\psi^\alpha: \quad \square\,(Y \in \{zero, pos\}),$$

which can be model-checked over ANY-Y$^\alpha$.

Joint Abstraction

System Only:

$$\left[|P|\right] \quad \subseteq \quad Q \qquad \text{implied by}$$

$$\left[|P|\right] \subseteq \left(|P|\right) \subseteq Q$$

System + Specification:

$$\left[|P|\right] \quad \subseteq \quad \left[|Q|\right]$$

implied by

$$\left[|P|\right] \subseteq \left(|P|\right)^{+} \subseteq \left(|Q|\right)^{-} \subseteq \left[|Q|\right]$$

Can be justified by

$$\left[|P|\right] \subseteq \left[|Q|\right] \qquad \Longleftrightarrow \qquad \left[|P|\right] \cap \overline{\left[|Q|\right]} \subseteq \emptyset$$

Abstraction of Assertions

Assume that the abstraction mapping is given as $V_A = \mathcal{E}^\alpha(V)$. How to lift α from states to assertions?

There are two (dual) ways to abstract an assertion p:

$$\begin{array}{llcl}\alpha^-(p): & \forall V\ (V_A = \mathcal{E}^\alpha(V) & \rightarrow & p(V)) \quad \text{and} \\ \alpha^+(p): & \exists V\ (V_A = \mathcal{E}^\alpha(V) & \wedge & p(V))\end{array}$$

Observe:

- An abstract state $S \in \Sigma_A$ satisfies $\alpha^-(p)$ iff all concrete states $s \in \alpha^{-1}(S)$ satisfy p.
- An abstract state $S \in \Sigma_A$ satisfies $\alpha^+(p)$ iff some concrete state $s \in \alpha^{-1}(S)$ satisfies p.

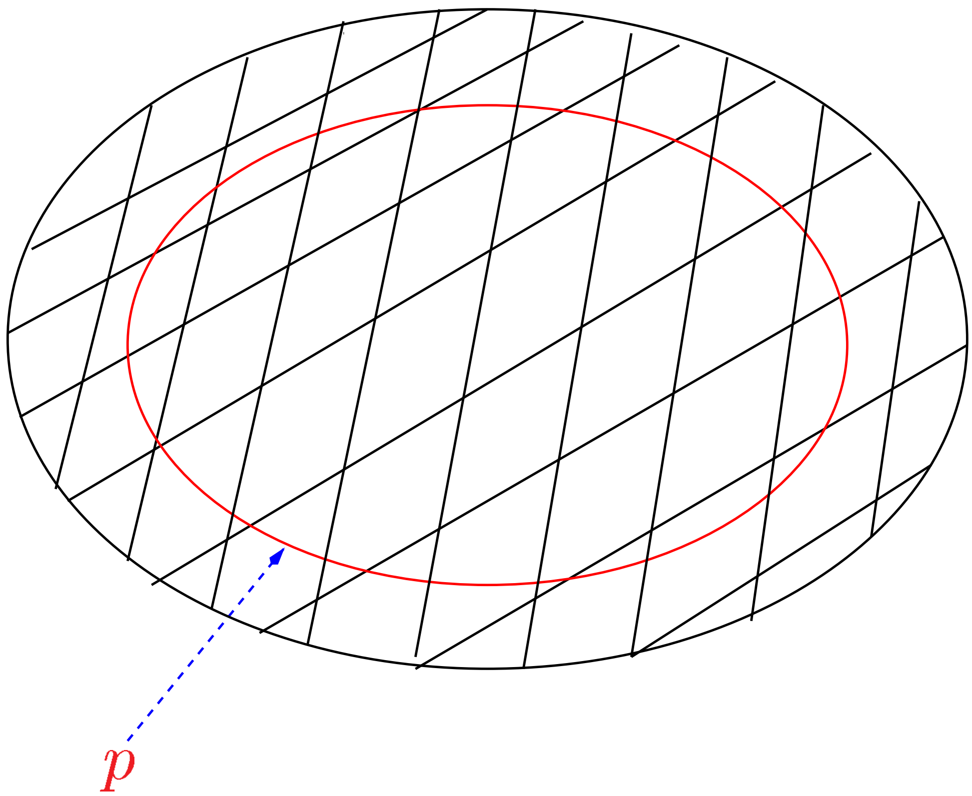
Equivalently:

$$\underbrace{\alpha^{-1}(\|\alpha^-(p)\|)}_{\text{contracting}} \quad \subseteq \quad \|p\| \quad \subseteq \quad \underbrace{\alpha^{-1}(\|\alpha^+(p)\|)}_{\text{expanding}}$$

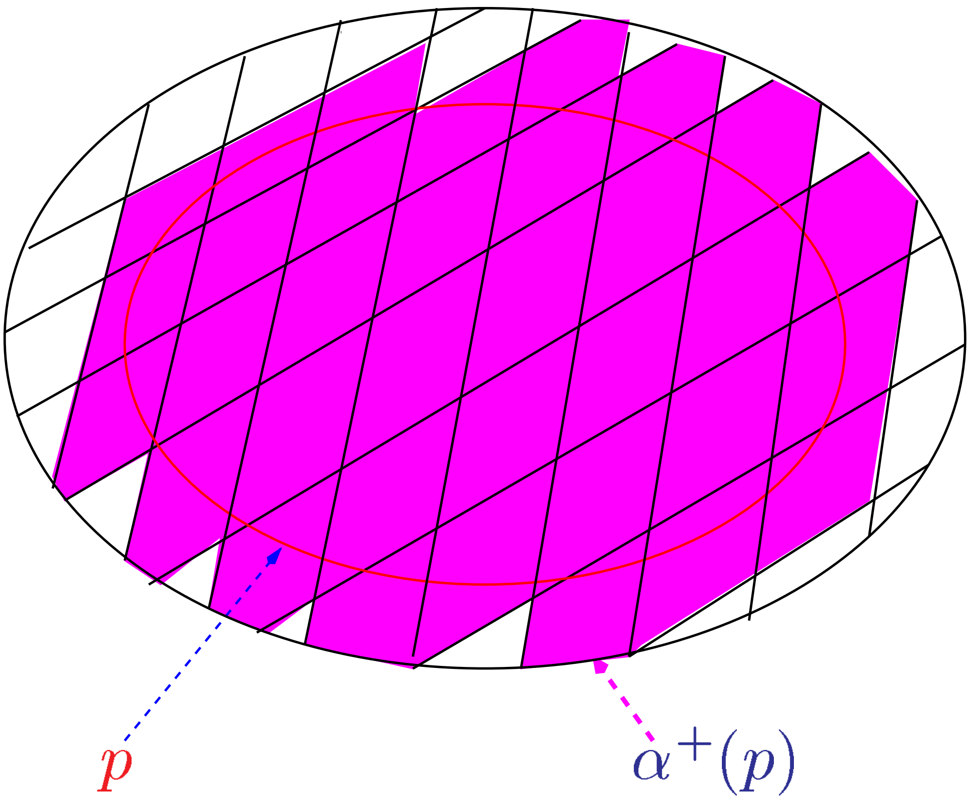
For example, for α : **integer** $\mapsto \{neg, zero, pos\}$,

$$\begin{array}{lcl}\alpha^-(0 \leq y \leq 5) & : & Y = zero \\ \alpha^+(0 \leq y \leq 5) & : & Y \in \{zero, pos\}\end{array}$$

The Two Abstraction Transformers

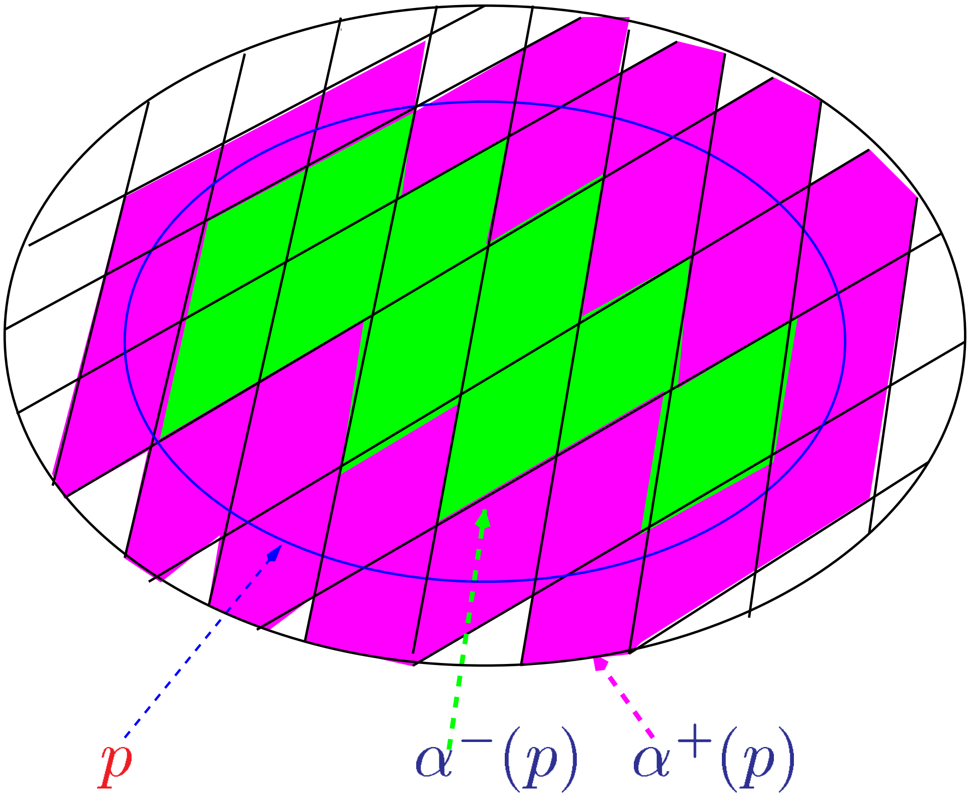


The Existential (expanding) Abstraction



An abstract state s belongs to $\alpha^+(p)$ if some concrete state α-mapped into s satisfies p.

The Universal (contracting) Abstraction



An abstract state s belongs to $\alpha^-(p)$ if all concrete states α-mapped into s satisfy p.

Sound Temporal Abstraction

For a temporal formula ψ, let ψ^α be obtained from ψ by replacing

- Every maximal p, a sub-assertion of ψ of positive polarity (enclosed within an even number of negations), by $\alpha^-(p)$, and
- Every maximal p, a sub-assertion of ψ of negative polarity (enclosed within an odd number of negations), by $\alpha^+(p)$.

Then

$$\models \psi^\alpha \qquad \text{implies} \qquad \models \psi.$$

This claim is based on the observation (provable by induction on the structure of ψ) that, for every state sequence $\sigma : s_0, s_1, \ldots$, and every position $j \geq 0$,

$$(\alpha(\sigma), j) \models \psi^\alpha \qquad \text{implies} \qquad (\sigma, j) \models \psi.$$

And Now to Systems

Given an FKS $\mathcal{K} = \langle V, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$, there exists a temporal formula $Sem(\mathcal{K})$, called the temporal semantics of $\mathcal{K}$, such that, for every infinite state sequence σ,

$$\sigma \models Sem(\mathcal{K}) \qquad \text{iff} \qquad \sigma \text{ is a computation of } \mathcal{K}.$$

$Sem(\mathcal{K})$ Is given by:

$$\Theta \;\wedge\; \Box\, \rho(V, \bigcirc V) \;\wedge\; \bigwedge_{J \in \mathcal{J}} \Box \Diamond J \;\wedge\; \bigwedge_{(p,q) \in \mathcal{C}} (\Box \Diamond p \rightarrow \Box \Diamond q)$$

Given a verification problem $\mathcal{K} \stackrel{?}{\models} \psi$, we construct the temporal formula

$$Ver(\mathcal{K}, \psi): \quad Sem(\mathcal{K}) \;\rightarrow\; \psi.$$

It is not difficult to establish that $\mathcal{K} \models \psi$ iff $Ver(\mathcal{K}, \psi)$ is valid.

Sound Joint Abstraction

For an FKS $\mathcal{K} = \langle V, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$, we define the α-abstracted version of $\mathcal{K}$ to be the FKS $\mathcal{K}^\alpha = \langle V_A, \Theta^\alpha, \rho^\alpha, \mathcal{J}^\alpha, \mathcal{C}^\alpha \rangle$, where

$$
\begin{array}{lcl}
\Theta^\alpha & = & \alpha^+(\Theta) \\
\rho^\alpha & = & \alpha^{++}(\rho) \\
\mathcal{J}^\alpha & = & \{\alpha^+(J) \mid J \in \mathcal{J}\} \\
\mathcal{C}^\alpha & = & \{(\alpha^-(p), \alpha^+(q)) \mid (p, q) \in \mathcal{C}\}
\end{array}
$$

Where,

$$
\alpha^{++}(\rho) \;=\; \exists V, V' \left(\begin{array}{cc} & V_A = \mathcal{E}^\alpha(V) \;\wedge\; V'_A = \mathcal{E}^\alpha(V') \\ \wedge & \rho(V, V') \end{array} \right)
$$

Soundness:

If $\mathcal{K}$ and ψ are abstracted according to the recipes presented above, then

$$
\mathcal{K}^\alpha \models \psi^\alpha \qquad \text{implies} \qquad \mathcal{K} \models \psi.
$$

Systematic Construction of Abstractions

Let $p_1, p_2, \ldots, p_k$ be the set of all atomic formulas referring to the data (non-control) variables appearing within conditions in the program P and within the temporal formula ψ.

Following [BBM95], define abstract boolean variables $B_{p_1}, B_{p_2}, \ldots, B_{p_k}$, one for each atomic data formula. The abstraction mapping α is defined by

$$\alpha: \quad \{B_{p_1} = p_1, B_{p_2} = p_2, \ldots, B_{p_k} = p_k\}$$

Example: Program BAKERY-2

local y_1, y_2 : **natural initially** $y_1 = y_2 = 0$

$$P_1 :: \left[\begin{array}{l} \ell_0 : \textbf{loop forever do} \\ \left[\begin{array}{ll} \ell_1 : & \textbf{Non-Critical} \\ \ell_2 : & y_1 := y_2 + 1 \\ \ell_3 : & \textbf{await } y_2 = 0 \ \lor \ y_1 < y_2 \\ \ell_4 : & \textbf{Critical} \\ \ell_5 : & y_1 := 0 \end{array} \right] \end{array} \right]$$

$\|$

$$P_2 :: \left[\begin{array}{l} m_0 : \textbf{loop forever do} \\ \left[\begin{array}{ll} m_1 : & \textbf{Non-Critical} \\ m_2 : & y_2 := y_1 + 1 \\ m_3 : & \textbf{await } y_1 = 0 \ \lor \ y_2 \leq y_1 \\ m_4 : & \textbf{Critical} \\ m_5 : & y_2 := 0 \end{array} \right] \end{array} \right]$$

The temporal properties for program BAKERY-2 are

$$\begin{array}{lcl} \psi_{exc} & : & \Box \neg(at\_\ell_4 \ \land \ at\_m_4) \\ \psi_{acc} & : & \Box \ (at\_\ell_2 \ \rightarrow \ \Diamond \ at\_\ell_4), \end{array}$$

Abstracting Program BAKERY-2

Define abstract variables $B_{y_1=0}$, $B_{y_2=0}$, and $B_{y_1<y_2}$.

local $B_{y_1=0}, B_{y_2=0}, B_{y_1<y_2}$: **boolean**
initially $B_{y_1=0} = B_{y_2=0} = 1, B_{y_1<y_2} = 0$

$$P_1 :: \left[\begin{array}{l} \ell_0 : \textbf{loop forever do} \\ \left[\begin{array}{ll} \ell_1 : & \textbf{Non-Critical} \\ \ell_2 : & (B_{y_1=0}, B_{y_1<y_2}) := (0,0) \\ \ell_3 : & \textbf{await } B_{y_2=0} \ \vee \ B_{y_1<y_2} \\ \ell_4 : & \textbf{Critical} \\ \ell_5 : & (B_{y_1=0}, B_{y_1<y_2}) := (1, \neg B_{y_2=0}) \end{array}\right] \end{array}\right]$$

$\|$

$$P_2 :: \left[\begin{array}{l} m_0 : \textbf{loop forever do} \\ \left[\begin{array}{ll} m_1 : & \textbf{Non-Critical} \\ m_2 : & (B_{y_2=0}, B_{y_1<y_2}) := (0,1) \\ m_3 : & \textbf{await } B_{y_1=0} \ \vee \ \neg B_{y_1<y_2} \\ m_4 : & \textbf{Critical} \\ m_5 : & (B_{y_2=0}, B_{y_1<y_2}) := (1,0) \end{array}\right] \end{array}\right]$$

The abstracted properties can now be model-checked.

II. Completeness

Warning:

Like every other completeness proof of an undecidable theory, our completeness proof is also non-constructive and does not provide recipes for constructing the proof.

A Complete System for Temporal Deduction

Proceeds through the following steps:

1. Rule INV for state invariances $\Box\, p$.
2. Rule CHAIN for response properties $p \Longrightarrow \Diamond\, q$ (state version) , achievable by a bounded number of helpful steps.
3. Rule WELL for response, using well-founded convergence measures.
4. Adding compassion to CHAIN and WELL .

To deal with a general temporal formula ψ, we first construct a temporal tester $\mathcal{K}_{\neg\psi}$ such that a sequence $\sigma : s_0, s_1, \ldots$ satisfies $\neg\psi$ iff it is an observation of $\mathcal{K}$. Then we use the fact that

$$\mathcal{K} \models \psi \qquad \text{iff} \qquad \mathcal{K} \,|||\, \mathcal{K}_{\neg\psi} \models \Diamond\, \mathit{false},$$

where $\mathcal{K} \,|||\, \mathcal{K}_{\neg\psi}$ is the synchronous parallel composition of $\mathcal{K}$ and $\mathcal{K}_{\neg\psi}$.

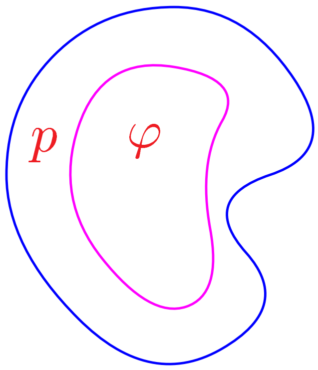
The VFA Method is Complete for Invariance

$$\mathcal{K} \models \square\, p$$

Temporal Deduction:

Rule INV

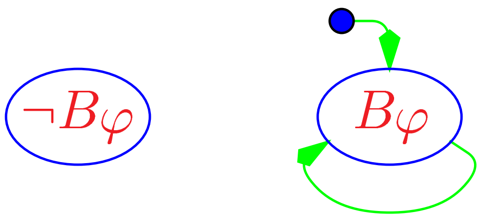
I1.	$\Theta \rightarrow \varphi$
I2.	$\rho \wedge \varphi \rightarrow \varphi'$
I3.	$\varphi \rightarrow p$
	$\square\, p$



VFA

$$\alpha : \quad B_\varphi = \varphi$$

$$\begin{array}{ccc} \mathcal{K} & \models & \square\, p \\ \big\downarrow \alpha^+ & & \big\downarrow \alpha^- \end{array}$$



$$\models \quad \square\, B_\varphi$$

Completeness for Response Properties Provable by Rule CHAIN

$$\mathcal{K} \models p \Longrightarrow \Diamond q$$

Rule CHAIN

For assertions $q = \varphi_0, \varphi_1, \ldots, \varphi_m$,
and justice requirements $J_1, \ldots, J_m \in \mathcal{J}$

$$\text{C1.} \quad p \quad \rightarrow \quad \bigvee_{j=0}^{m} \varphi_j$$

For $i = 1, \ldots, m$,

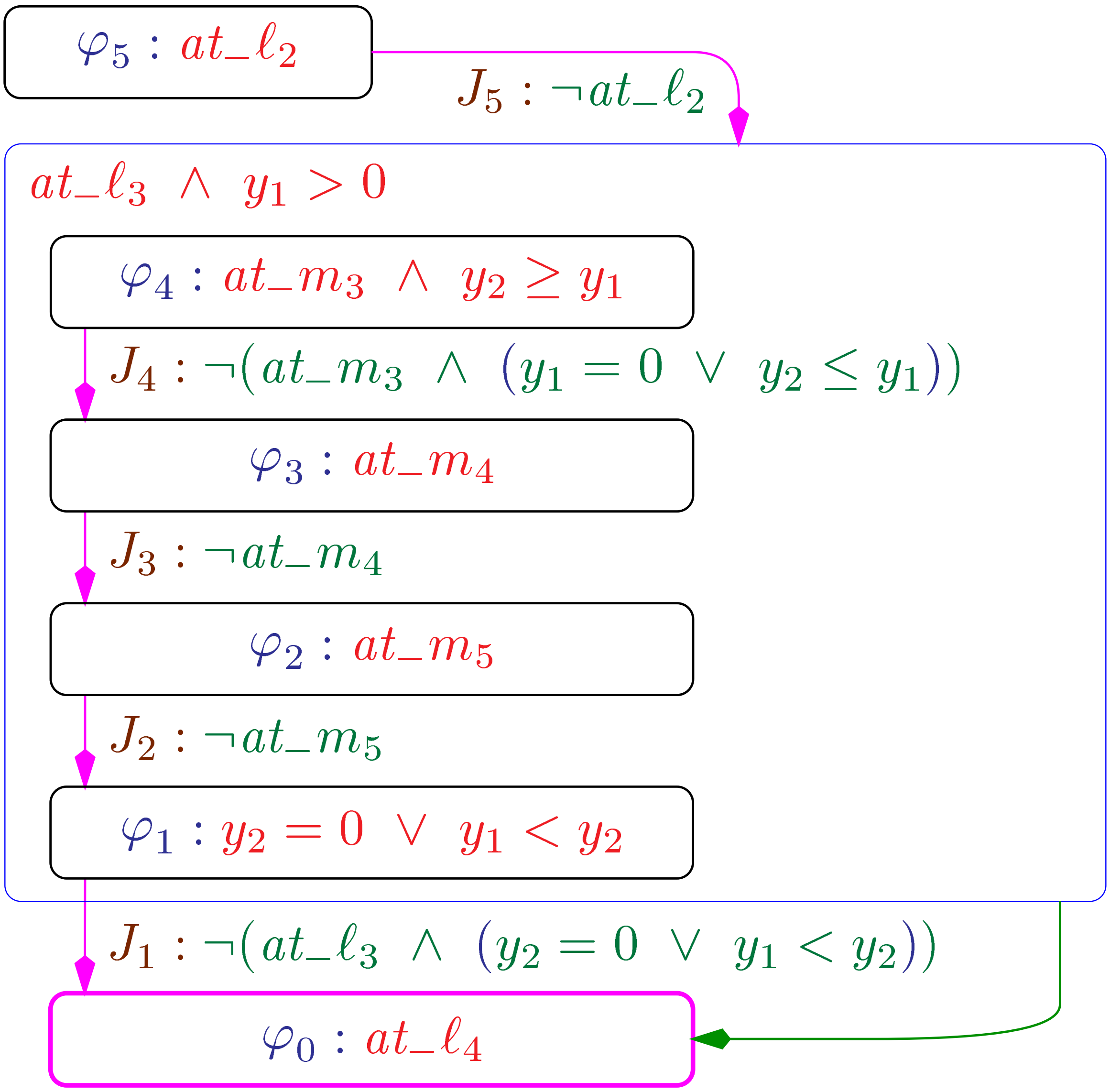
$$\left\{ \begin{array}{llcl} \text{C2.} & \rho \wedge \varphi_i & \rightarrow & \bigvee_{j \leq i} \varphi_j{}' \\ \text{C3.} & \rho \wedge \varphi_i \wedge J_i{}' & \rightarrow & \bigvee_{j < i} \varphi_j{}' \end{array} \right.$$

$$p \Longrightarrow \Diamond q$$

Can often be represented by a verification diagram.

Example: Verifying Accessibility for BAKERY-2

The following diagram verifies (using rule CHAIN) the property of accessibility for program BAKERY-2 specifiable by the formula $at\_\ell_2 \implies \diamond\ at\_\ell_4$.



The proof relies on the auxiliary invariant

$$I : \quad at\_m_{0..2} \ \rightarrow \ y_2 = 0$$

CHAIN-Relative Completeness of VFA

A finitary abstraction mapping α is said to be precise with respect to the assertion p if $\alpha^-(p) \sim \alpha^+(p)$. A sufficient condition for α to be precise w.r.t. p, is that α contains the definition $B_p = p$. For an α-precise p, we denote $\alpha(p) = \alpha^-(p) \sim \alpha^+(p)$.

The finitary mapping α is said to be adequate for the assertion p, assumed to be an invariant of system $\mathcal{K}$, if $\alpha^-(p)$ is an invariant of the abstracted system $\mathcal{K}^\alpha$. A sufficient condition for p-adequacy is that α is precise for φ, the auxiliary assertion used for verifying p by rule INV.

Claim 1. *Let us assume that the response property $\psi : p \Longrightarrow \diamondsuit\, q$ is verifiable by rule* CHAIN, *and that the abstraction α is precise w.r.t the assertions p, q, I, and $\varphi_1, \ldots, \varphi_m$, $J_1, \ldots, J_m$, appearing in the application of rule* CHAIN. *Also assume that α is adequate for the auxiliary invariant I used in the rule,*

Then ψ^α will model check successfully over $\mathcal{K}^\alpha$.

Corollary 1. *If $p \Longrightarrow \diamondsuit\, q$ is verifiable by rule* CHAIN, *then it is also verifiable by the* VFA *method.*

A Sketch of a Proof

Provability by CHAIN implies that premises C1, C2, and C3 are valid for

p, q, I, $\varphi_1, \ldots, \varphi_m$, $J_1, \ldots, J_m$, and ρ.

We will show that these premises are also valid for the choice of $\alpha(p)$, $\alpha(q)$, $\alpha(I)$, $\alpha(\varphi_1), \ldots, \alpha(\varphi_m)$, $\alpha(J_1), \ldots, \alpha(J_m)$, and $\alpha^{++}(\rho)$.

This will establish the validity of the abstracted response property $\psi^\alpha : \alpha(p) \Longrightarrow \diamondsuit\, \alpha(q)$ over $\mathcal{K}^\alpha$.

We illustrate the porting of validity on premise C2:

$$I \wedge \rho \wedge \varphi_i \quad \rightarrow \quad \bigvee_{j \leq i} \varphi_j{}'$$

Applying the $\exists\exists$-introduction rule, we obtain

$$\alpha^+(I \wedge \rho \wedge \varphi_i) \quad \rightarrow \quad \alpha^+(\bigvee_{j \leq i} \varphi_j{}'),$$

which implies

$$\alpha^-(I) \wedge \alpha^{++}(\rho) \wedge \alpha^-(\varphi_i) \quad \rightarrow \quad \bigvee_{j \leq i} \alpha^+(\varphi_j)'.$$

Using the precision of I and the φ_j's, we obtain

$$\alpha(I) \wedge \alpha^{++}(\rho) \wedge \alpha(\varphi_i) \quad \rightarrow \quad \bigvee_{j \leq i} \alpha(\varphi_j)',$$

which is the abstracted version of C2.

We conclude that ψ^α is valid over $\mathcal{K}^\alpha$ and, therefore, will model check successfully.

Response Proofs Requiring Unbounded Convergence Measures

Not every valid response property can be proven by rule CHAIN. Termination of program UP-DOWN below requires a convergence measure ranging over the ordinal $\omega + 1$.

$$\begin{array}{lll} x\text{:} & \textbf{boolean} & \textbf{initially } x = 0 \\ y\text{:} & \textbf{natural} & \textbf{initially } y = 0 \end{array}$$

$$\left[\begin{array}{ll} \ell_0: & \textbf{while } x = 0 \textbf{ do} \\ & [\ell_1: \ y := y + 1] \\ \ell_2: & \textbf{while } y > 0 \textbf{ do} \\ & [\ell_3: \ y := y - 1] \\ \ell_4: & \end{array}\right] \quad \| \quad \left[\begin{array}{ll} m_0: & x := 1 \\ m_1: & \end{array}\right]$$

$$- \ P_1 \ - \qquad\qquad - \ P_2 \ -$$

The Abstraction that Works

$$\begin{array}{lll} x\colon & \textbf{boolean} & \textbf{initially } x = 0 \\ Y\colon & \{zero, pos\} & \textbf{initially } Y = zero \end{array}$$

$$\left[\begin{array}{l} \ell_0: \quad \textbf{while } x = 0 \textbf{ do} \\ \quad [\ell_1: \ Y := add1(Y)] \\ \ell_2: \quad \textbf{while } Y = pos \textbf{ do} \\ \quad [\ell_3: \ Y := sub1(Y)] \\ \ell_4: \end{array}\right] \quad \| \quad \left[\begin{array}{ll} m_0: & x := 1 \\ m_1: & \end{array}\right]$$

$$- \quad P_1 \quad - \qquad\qquad\qquad - \quad P_2 \quad -$$

Abstract variable Y ranges over a fair data structure consisting of the domain $\{zero, pos\}$ and the operations $add1$, $sub1$, which satisfy the constraints

$$\begin{array}{lll} u = add1(v) & \rightarrow & u = pos \\ u = sub1(v) & \rightarrow & u = zero \ \vee \ v = pos, \end{array}$$

and the compassion requirement

$$(Y' = sub1(Y) \quad , \quad Y' = add1(Y) \ \vee \ Y = zero).$$

Termination of the abstracted UP-DOWN program, can now be model checked.

Discussion and Conclusions

- Work is still in progress. However, there are very good indications that the VFA method is complete relative to temporal deduction.

- This identifies the VFA approach as a viable and promising alternative/complement to temporal deduction.

- Note that first-order (state-)deduction has to stay, because this is the main tool for computing the abstractions $\alpha^+(p)$, $\alpha^-(q)$, $\alpha^{++}(\rho)$.

- We urgently need an extensive research into methods and heuristics for the automatic construction of useful abstractions.