

# 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  $\psi$  valid for  $\mathcal{K}$ , we can find an abstraction  $\alpha$  such that  $\mathcal{K}^\alpha$  is finite-state,  $\psi^\alpha$  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  $\psi$  into a propositional LTL formula  $\psi^\alpha$ .
- 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  $V$ .  $\Sigma$  – the set of all states.
- $\Theta$  – An initial condition. A satisfiable assertion that characterizes the initial states.
- $\rho$  – 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  $x := x + 1$ .
- $\mathcal{J} = \{J_1, \dots, J_k\}$  A set of justice (weak fairness) requirements. Ensure that a computation has infinitely many  $J_i$ -states for each  $J_i$ ,  $i = 1, \dots, k$ .
- $\mathcal{C} = \{\langle p_1, q_1 \rangle, \dots, \langle p_n, q_n \rangle\}$  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

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

satisfying the following requirements:

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

## Example: Program ANY-Y

Consider the program

$x, y$ : integer initially  $x = y = 0$

$$\begin{array}{c}
 \left[ \begin{array}{l}
 \ell_0 : \text{while } x = 0 \text{ do} \\
 \quad \left[ \ell_1 : y := y + 1 \right] \\
 \ell_2 :
 \end{array} \right] \quad \parallel \quad \left[ \begin{array}{l}
 m_0 : x := 1 \\
 m_1 :
 \end{array} \right] \\
 - \quad P_1 \quad - \qquad \qquad \qquad - \quad P_2 \quad -
 \end{array}$$

## The Corresponding FKS

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

- Initial condition:

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

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

$$\rho_I : \pi_1' = \pi_1 \wedge \pi_2' = \pi_2 \wedge x' = x \wedge y' = y$$

$$\rho_{\ell_0} : \pi_1 = \ell_0 \wedge \left( \begin{array}{l} x = 0 \wedge \pi_1' = \ell_1 \\ \vee \\ x \neq 0 \wedge \pi_1' = \ell_2 \end{array} \right) \\ \wedge \pi_2' = \pi_2 \wedge x' = x \wedge y' = y$$

- 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  $\Sigma$  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**  $\alpha$  to abstract the concrete FKS  $\mathcal{K}$  into an **abstract FKS**  $\mathcal{K}_A^\alpha$ .
- **Abstract** the temporal property  $\psi$  into an **abstract property**  $\psi^\alpha$ .
- **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  $\psi^\alpha$  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

$x, y$ : integer initially  $x = y = 0$

$$\begin{array}{c}
 \left[ \begin{array}{l}
 \ell_0 : \text{while } x = 0 \text{ do} \\
 \quad [\ell_1 : y := y + 1] \\
 \ell_2 :
 \end{array} \right] \quad \parallel \quad \left[ \begin{array}{l}
 m_0 : x := 1 \\
 m_1 :
 \end{array} \right] \\
 \text{--- } P_1 \text{ ---} \qquad \qquad \qquad \text{--- } P_2 \text{ ---}
 \end{array}$$

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  $\alpha$  maps the domain of  $y$  into the domain of  $Y$  as follows:

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

## The Abstracted Version

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

$x$ : **integer**                    **initially**  $x = 0$   
 $Y$ :  $\{neg, zero, pos\}$     **initially**  $Y = zero$

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

||

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

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

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

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

# Joint Abstraction

## System Only:

$$\llbracket P \rrbracket \subseteq Q \quad \text{implied by}$$

$$\llbracket P \rrbracket \subseteq \left( \llbracket P \rrbracket \right)^+ \subseteq Q$$

## System + Specification:

$$\llbracket P \rrbracket \subseteq \llbracket Q \rrbracket \quad \text{implied by}$$

$$\llbracket P \rrbracket \subseteq \left( \llbracket P \rrbracket \right)^+ \subseteq \left( \llbracket Q \rrbracket \right)^- \subseteq \llbracket Q \rrbracket$$

Can be justified by

$$\llbracket P \rrbracket \subseteq \llbracket Q \rrbracket \quad \iff \quad \llbracket P \rrbracket \cap \overline{\llbracket Q \rrbracket} \subseteq \emptyset$$

## Abstraction of Assertions

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

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

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

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  $p$ .

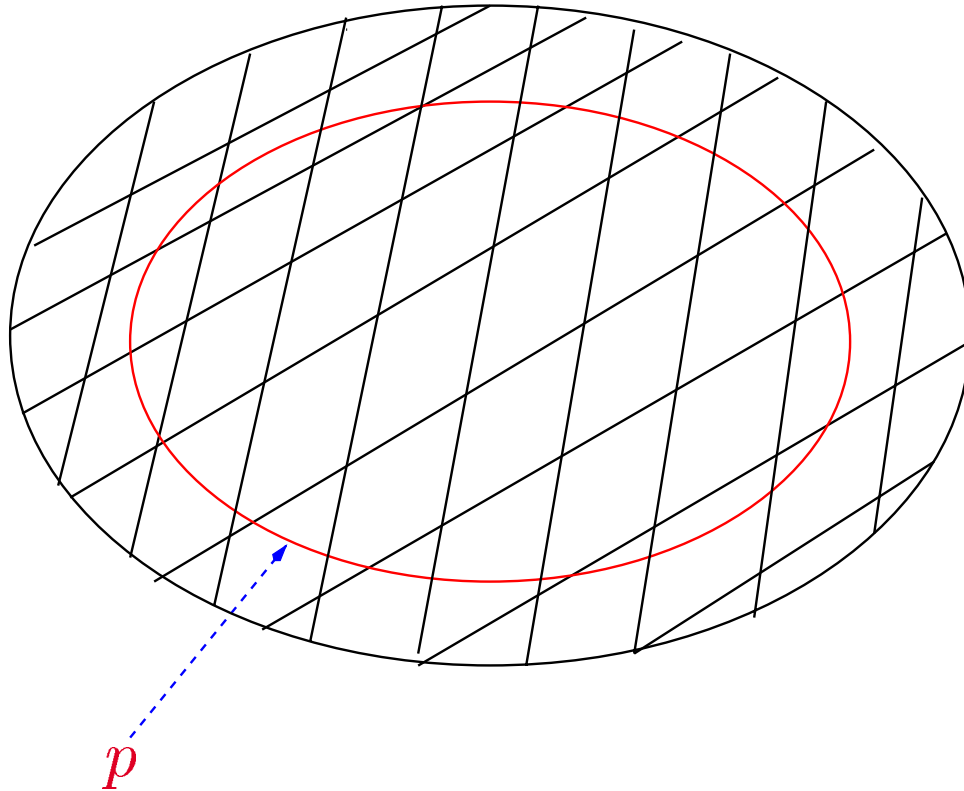
Equivalently:

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

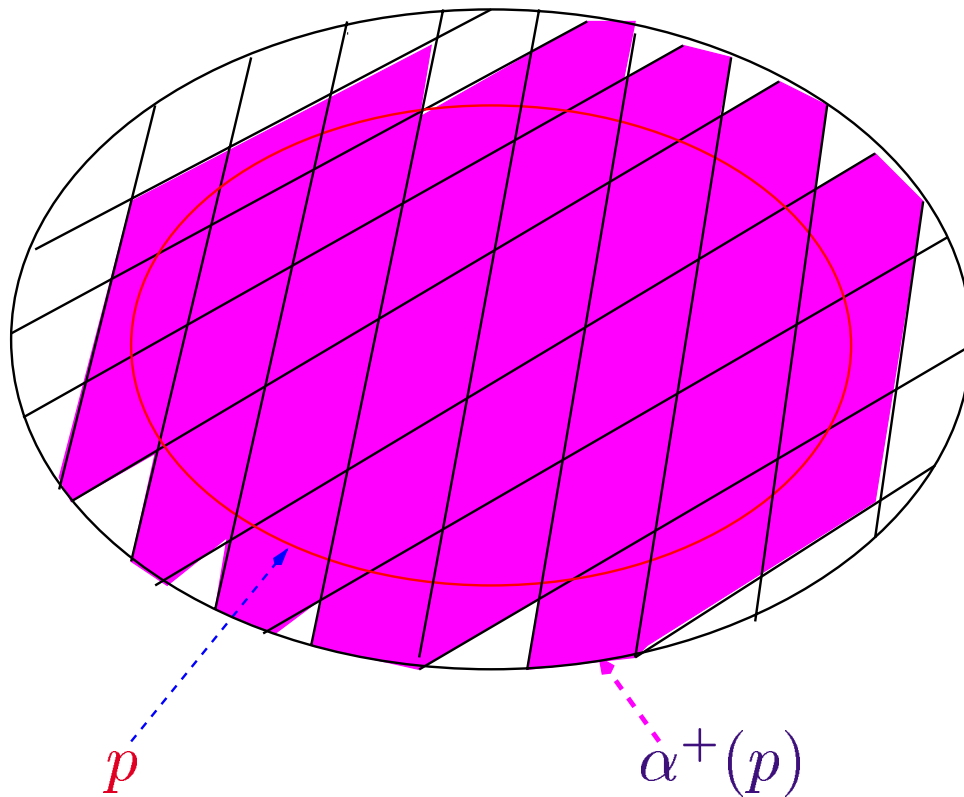
For example, for  $\alpha : \text{integer} \mapsto \{\text{neg}, \text{zero}, \text{pos}\}$ ,

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

# The Two Abstraction Transformers

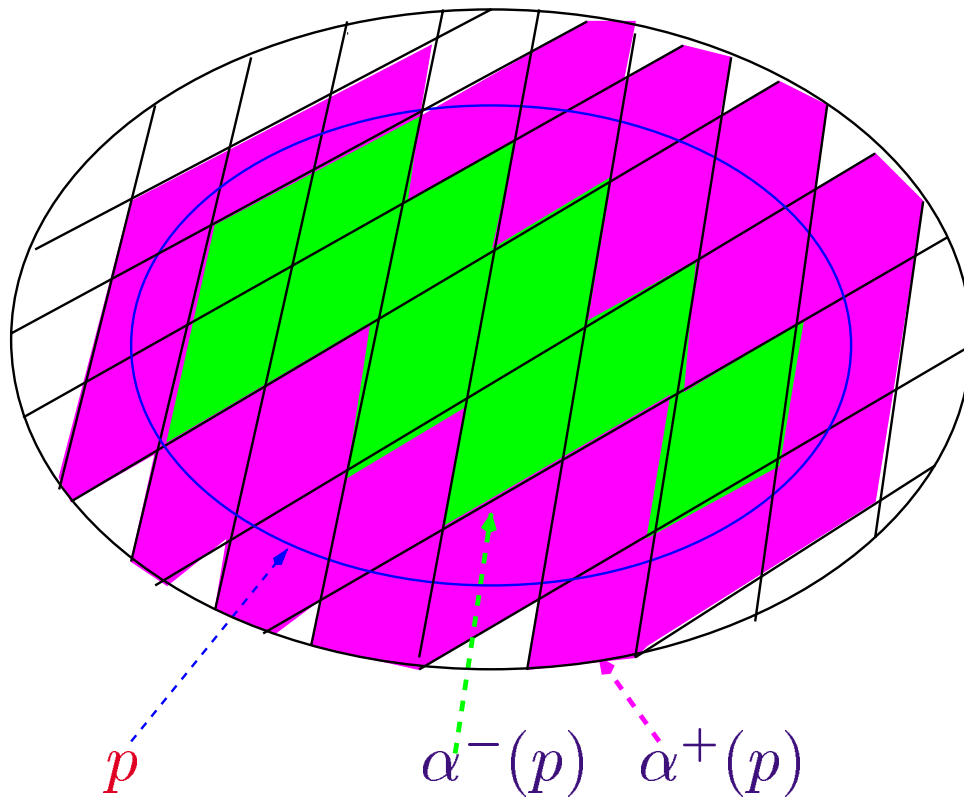


# The Existential (expanding) Abstraction



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

# The Universal (contracting) Abstraction



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

## Sound Temporal Abstraction

For a temporal formula  $\psi$ , let  $\psi^\alpha$  be obtained from  $\psi$  by replacing

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

Then

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

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

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

## 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  $\sigma$ ,

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

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

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

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

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

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  $\alpha$ -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

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

Where,

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

### Soundness:

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

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

## Systematic Construction of Abstractions

Let  $p_1, p_2, \dots, p_k$  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  $\psi$ .

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

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

## Example: Program BAKERY-2

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

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

||

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

The temporal properties for program BAKERY-2 are

$$\begin{aligned} \psi_{exc} & : \square \neg (at_{\ell_4} \wedge at_{m_4}) \\ \psi_{acc} & : \square (at_{\ell_2} \rightarrow \lozenge at_{\ell_4}), \end{aligned}$$

## 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**  $B_{y_1=0} = B_{y_2=0} = 1, B_{y_1 < y_2} = 0$

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

||

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

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  $\square p$ .
2. Rule **CHAIN** for response properties  $p \implies \diamond q$  (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  $\psi$ , we first construct a temporal tester  $\mathcal{K}_{\neg\psi}$  such that a sequence  $\sigma : s_0, s_1, \dots$  satisfies  $\neg\psi$  iff it is an **observation** of  $\mathcal{K}$ .

Then we use the fact that

$$\mathcal{K} \models \psi \quad \text{iff} \quad \mathcal{K} \parallel \mathcal{K}_{\neg\psi} \models \diamond \text{false},$$

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

# The VFA Method is Complete for Invariance

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

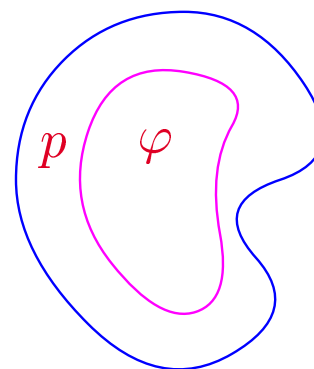
## Temporal Deduction:

Rule INV

1.  $\Theta \rightarrow \varphi$
2.  $\rho \wedge \varphi \rightarrow \varphi'$
3.  $\varphi \rightarrow p$

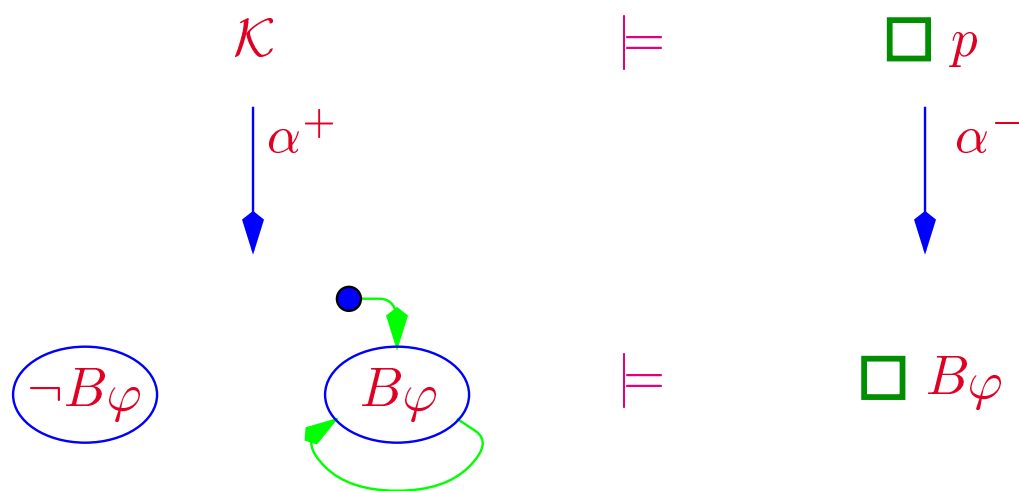
---

$\square p$



## VFA

$$\alpha : B\varphi = \varphi$$



# Completeness for Response Properties

## Provable by Rule CHAIN

$$\mathcal{K} \models p \implies \diamond q$$

### Rule CHAIN

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

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

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

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

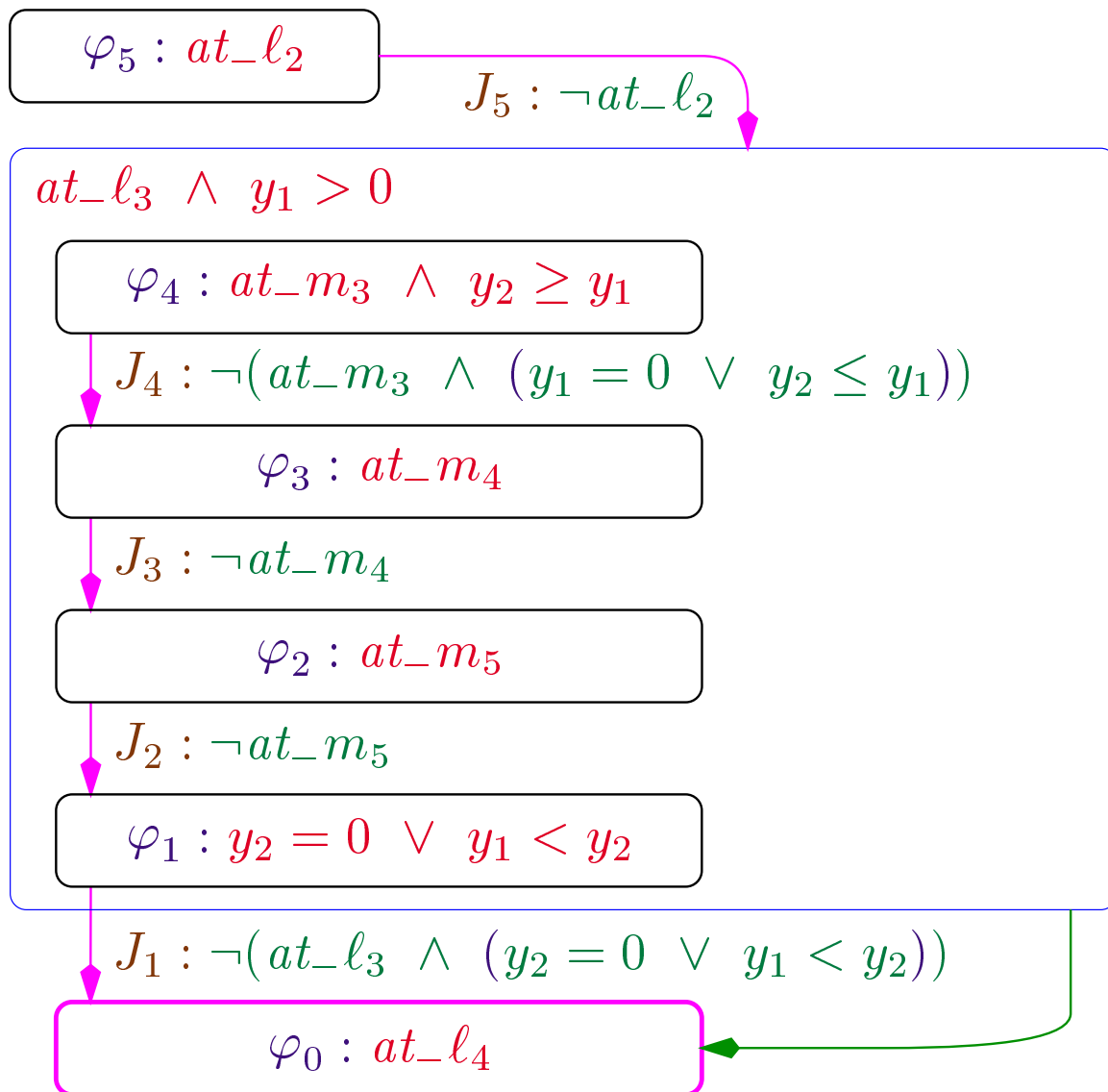
---


$$p \implies \diamond q$$

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  $at\_l_2 \implies \diamond at\_l_4$ .



The proof relies on the auxiliary invariant

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

## CHAIN-Relative Completeness of VFA

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

The finitary mapping  $\alpha$  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  $\alpha$  is precise for  $\varphi$ , the auxiliary assertion used for verifying  $p$  by rule **INV**.

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

*Then  $\psi^\alpha$  will model check successfully over  $\mathcal{K}^\alpha$ .*

**Corollary 1.** *If  $p \implies \diamond q$  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

$p, q, I, \varphi_1, \dots, \varphi_m, J_1, \dots, J_m,$  and  $\rho$ .

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

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

We illustrate the porting of validity on premise C2:

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

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

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

which implies

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

Using the precision of  $I$  and the  $\varphi_j$ 's, we obtain

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

which is the abstracted version of C2.

We conclude that  $\psi^\alpha$  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  $\omega + 1$ .

$x$ : **boolean** initially  $x = 0$   
 $y$ : **natural** initially  $y = 0$

$$\begin{array}{c}
 \left[ \begin{array}{l}
 l_0 : \text{while } x = 0 \text{ do} \\
 \quad [l_1 : y := y + 1] \\
 l_2 : \text{while } y > 0 \text{ do} \\
 \quad [l_3 : y := y - 1] \\
 l_4 :
 \end{array} \right] \quad \parallel \quad \left[ \begin{array}{l}
 m_0 : x := 1 \\
 m_1 :
 \end{array} \right] \\
 \text{--- } P_1 \text{ ---} \qquad \qquad \text{--- } P_2 \text{ ---}
 \end{array}$$

## The Abstraction that Works

$x$ : **boolean**      **initially**  $x = 0$   
 $Y$ :  $\{zero, pos\}$     **initially**  $Y = zero$

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

—  $P_1$  —

—  $P_2$  —

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

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

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.