Verifying Liveness Properties of Parameterized Systems

Amir Pnueli

The John von Neumann Minerva Center for the Verification of Reactive Systems
at the Weizmann Institute of Sciences

Saarbrücken, March, 2002
Symposium in Honor of Moseh Vardi

Joint work with:

Yonit Kesten, Elad Shahar, Lenore Zuck

Parameterized Systems

In its simplest form, a parameterized system is a parallel composition, such as

$S(N) = P[1] \parallel \cdots \parallel P[N]$ or $S(N) = C \parallel P[1] \parallel \cdots \parallel P[N]$

where $N > 1$ is a parameter and $P[1], \ldots, P[N]$ are finite-state processes.

Uniform Verification: Establish in one verification effort that all instances of the system satisfy a property ψ. That is, $S(N) \models \psi$, for every $N > 1$.

Many Approaches

Many approaches have been proposed and developed for verifying liveness properties of parameterized systems, including:

- Regular model checking.
- The method of network invariants.
- Counter Abstraction.
- The method of invisible invariants.

Today, we will discuss the method of network invariants.

Parameterized Verification by Networks Invariants

Consider a parameterized FDS (fair discrete system), $S(n)$: $P_1 \parallel \cdots \parallel P_n$, for which we wish to verify property p.

Assume that p only refers to the observable variables of P_1 and that processes $P_2, \ldots, P_n$ are identical (up to renaming).

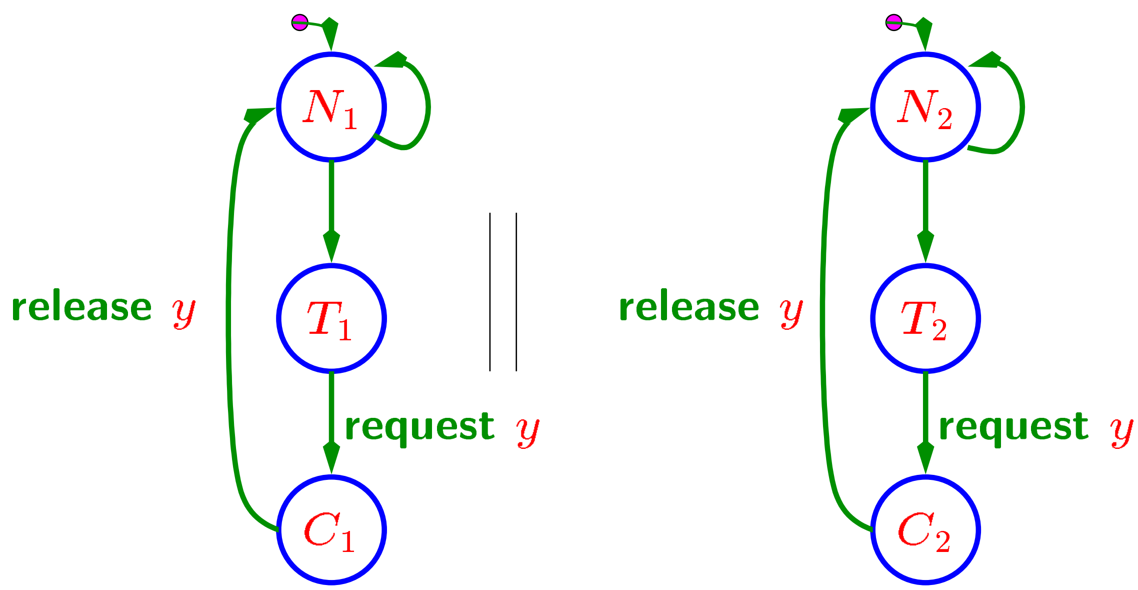
We propose the following strategy ([BCG86], [WL89], [SG89], [CGJ95], [KM95]):

1. Devise a network invariant $\mathcal{I}$, which is a fixed FDS intended to provide a modular abstraction for the parallel composition $P_2 \parallel \cdots, \parallel P_n$ for any n.
2. Confirm that $\mathcal{I}$ is indeed a network invariant, by verifying that $P_2 \sqsubseteq_M \mathcal{I}$ and that $(\mathcal{I} \parallel P_2) \sqsubseteq_M \mathcal{I}$.
3. Model check $(P_1 \parallel \mathcal{I}) \models p$.
4. Conclude that $S(n) \models p$ for every $n > 1$.

Example: Program MUX-SEM

Consider program MUX-SEM(2), implementing mutual exclusion by semaphores.

y: **natural initially** $y = 1$

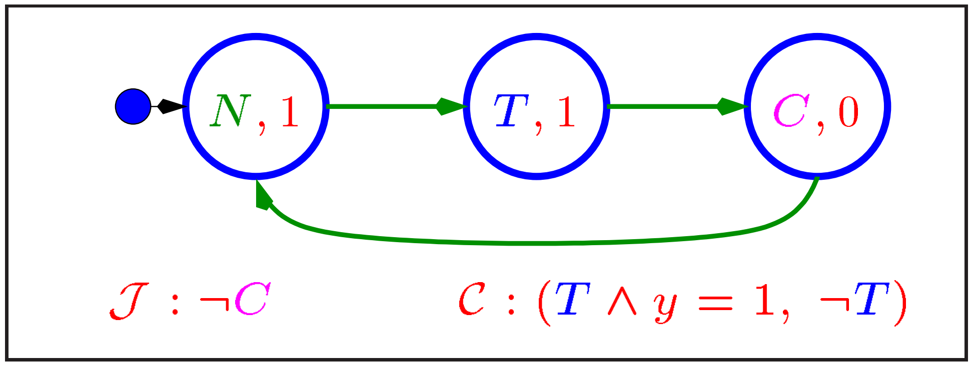


The semaphore instructions **request** y and **release** y respectively stand for

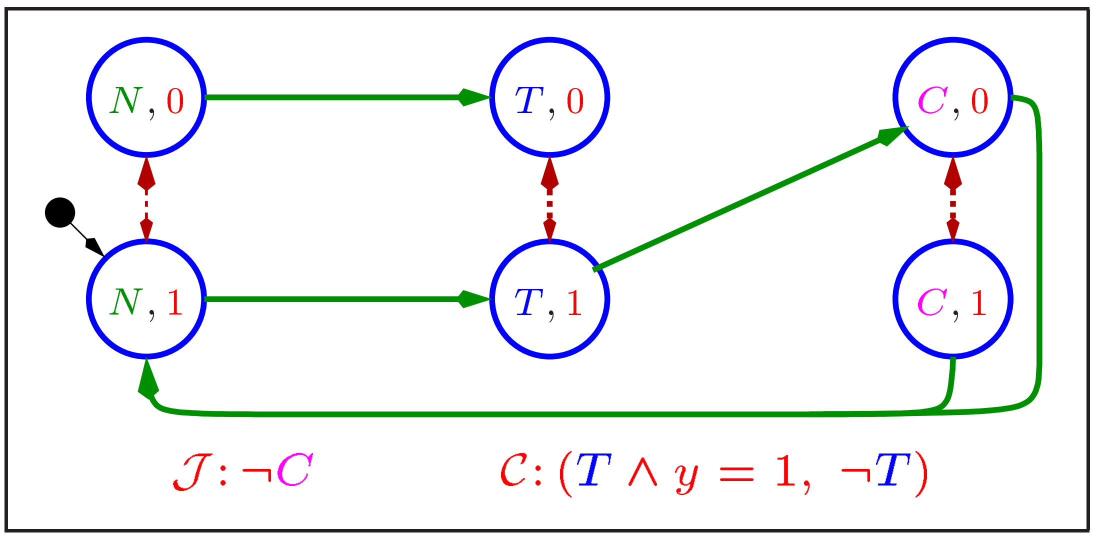
$$\langle \textbf{when } y = 1 \textbf{ do } y := 0 \rangle \quad \text{and} \quad y := 1.$$

Open Computations

A closed-system view of P_1 is given by:



An open-system view:



System transitions are represented by $\longrightarrow$, while environment transitions are represented by $\longrightarrow$. An example of an open computation of P_1 is:

$$\langle \pi : N,\ y : 1\rangle \xrightarrow{S} \langle \pi : T,\ y : 1\rangle \xrightarrow{S} \langle \pi : C,\ y : 0\rangle \xrightarrow{E}$$
$$\langle \pi : C,\ y : 1\rangle \xrightarrow{S} \langle \pi : N,\ y : 1\rangle \cdots$$

Modular Abstraction

For $\sigma : s_0, s_1, \ldots$ an open computation of the FDS $\mathcal{D}$, we say that the sequence $s_0 \Downarrow_{\mathcal{O}},\ s_1 \Downarrow_{\mathcal{O}},\ \ldots,$ is an observation of $\mathcal{D}$, where $s_i \Downarrow_{\mathcal{O}}$ is the projection of state s_i on $\mathcal{O}$ the observables of $\mathcal{D}$. Let $\mathit{Obs}(\mathcal{D})$ denote the set of all observations of system $\mathcal{D}$.

The FDS $\mathcal{D}_C$ is defined to be comparable with the FDS $\mathcal{D}_A$ if $\mathcal{O}_C = \mathcal{O}_A$.

The FDS $\mathcal{D}_A$ is a (modular) abstraction of the comparable $\mathcal{D}_C$, denoted by

$$\mathcal{D}_C \sqsubseteq_M \mathcal{D}_A,$$

if

$$\mathit{Obs}(\mathcal{D}_C) \subseteq \mathit{Obs}(\mathcal{D}_A).$$

We refer to $\mathcal{D}_C$ and $\mathcal{D}_A$ as the concrete and abstract systems, respectively.

Properties of the Modular Abstraction Relation

The following can be shown:

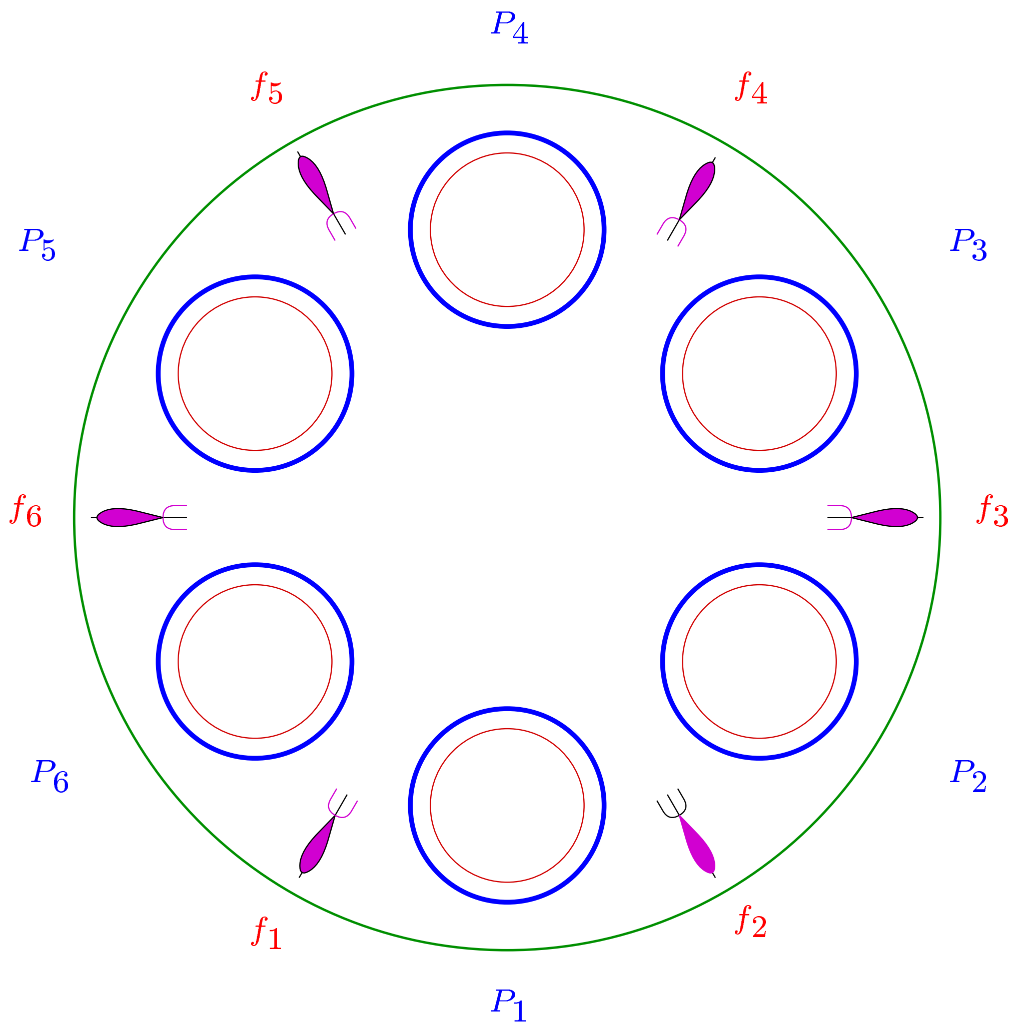
- The modular abstraction relation is reflexive and transitive.
- Modular abstraction is compositional. That is, whenever $\mathcal{D}_C \sqsubseteq_M \mathcal{D}_A$ then

$$(\mathcal{D}_C \parallel Q) \sqsubseteq_M (\mathcal{D}_A \parallel Q)$$

- Modular abstraction is property restricting. That is, if $\mathcal{D}_C \sqsubseteq_M \mathcal{D}_A$ then
  $\mathcal{D}_A \models p$ implies $\mathcal{D}_C \models p$.

The Dining Philosophers Metaphor

Consider n philosophers arranged around a table.



The life of a philosopher alternates between a thinking phase (a non-critical activity) and an eating phase. In order to eat, a philosopher needs both forks.

Program Dine

A first attempt yields the following program Dine:

$$
\begin{array}{lll}
\textbf{in} & n & : \textbf{integer initially } n \geq 2 \\
\textbf{local} & f & : \textbf{array } [1..n] \textbf{ of integer initially } f = 1
\end{array}
$$

$$
\mathop{\Big\|}_{j=1}^{n} P[j] :: \left[\begin{array}{l}
\ell_0 : \textbf{loop forever do} \\
\quad \left[\begin{array}{lll}
\ell_1 : & \textbf{Non-Critical} & \\
\ell_2 : & \textbf{request} & f[j] \\
\ell_3 : & \textbf{request} & f[j \oplus_n 1] \\
\ell_4 : & \textbf{Critical} & \\
\ell_5 : & \textbf{release} & f[j] \\
\ell_6 : & \textbf{release} & f[j \oplus_n 1]
\end{array}\right]
\end{array}\right]
$$

It is not difficult to verify the following safety property

$$\square \neg(at\_\ell_4[1] \ \wedge \ at\_\ell_4[2]),$$

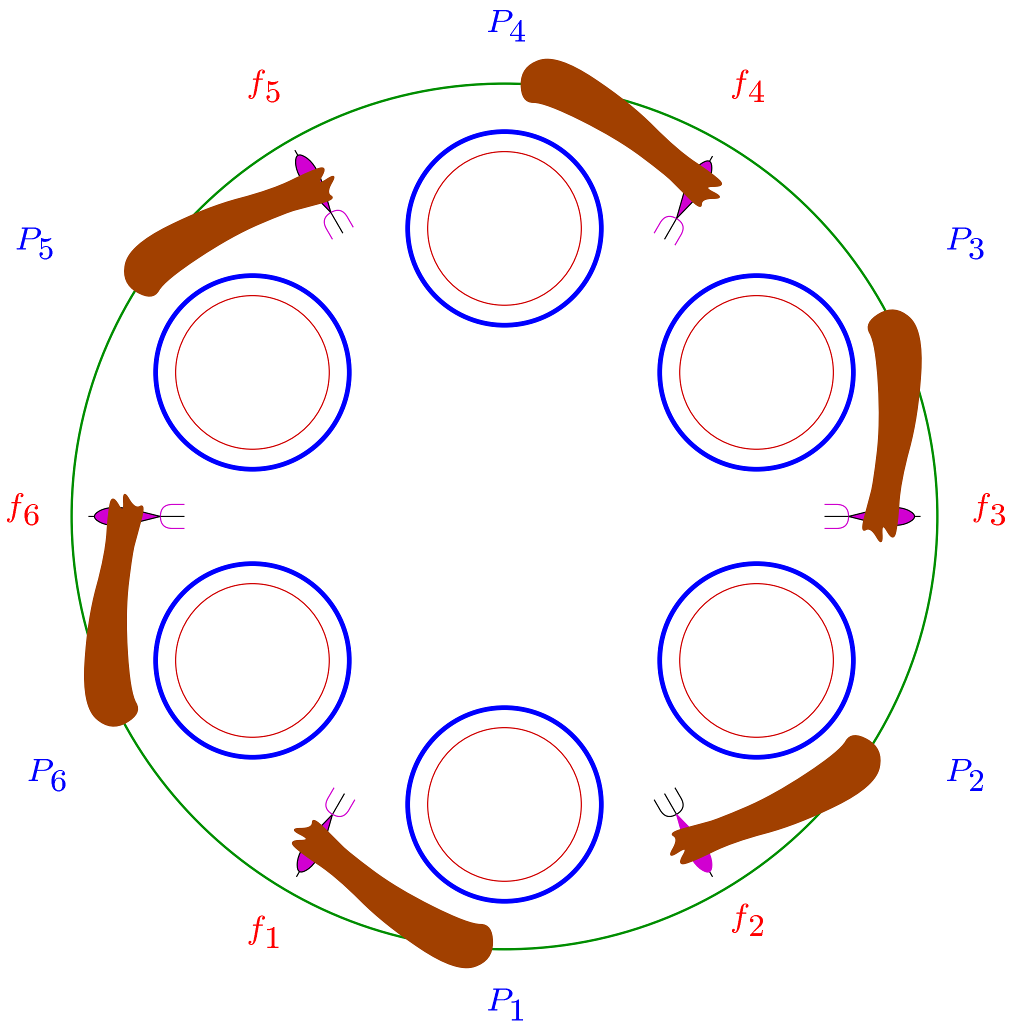
stating that philosophers $P[1]$ and $P[2]$ can never eat at the same time.

Accessibility not Guaranteed

Unfortunately, Dine cannot ensure accessibility for $P[1]$, specifiable by

$$\square\ (at\_\ell_2[1] \quad \rightarrow \quad \diamondsuit\ at\_\ell_4[1])$$

Because all philosophers may deadlock together.



Solution: One Contrary Philosopher

$$
\begin{array}{l}
\textbf{local} \quad f \quad : \textbf{array } [1..n] \textbf{ of natural initially } f = 1 \\
\overset{n-1}{\underset{j=1}{\|}} P[j] :: \left[\begin{array}{l}
\ell_0 : \textbf{loop forever do} \\
\quad \left[\begin{array}{lll}
\ell_1 : & \textbf{Non-Critical} \\
\ell_2 : & \textbf{request } f[j] \\
\ell_3 : & \textbf{request } f[j \oplus_n 1] \\
\ell_4 : & \textbf{Critical} \\
\ell_5 : & \textbf{release } f[j] \\
\ell_6 : & \textbf{release } f[j \oplus_n 1]
\end{array}\right]
\end{array}\right] \\
\qquad \| \\
P[n] :: \left[\begin{array}{l}
\ell_0 : \textbf{loop forever do} \\
\quad \left[\begin{array}{lll}
\ell_1 : & \textbf{Non-Critical} \\
\ell_2 : & \textbf{request } f[1] \\
\ell_3 : & \textbf{request } f[n] \\
\ell_4 : & \textbf{Critical} \\
\ell_5 : & \textbf{release } f[1] \\
\ell_6 : & \textbf{release } f[n]
\end{array}\right]
\end{array}\right]
\end{array}
$$

Wish to establish accessibility, expressible by

$$\psi_{acc}: \quad \square\ (at\_\ell_2[j] \ \rightarrow\ \diamondsuit\ (at\_\ell_4[j]))$$

Designing a Network Invariant

A regular philosopher can be viewed as a system $P(\mathit{left}, \mathit{right})$, where the semaphores left and right are the only observables.

We are searching for an invariant $\mathcal{I}(\mathit{left}, \mathit{right})$ which is a modular abstraction of the philosopher's string

$$S[2..k] :: \left[\begin{array}{lll} \textbf{in-out} & \mathit{left}, \mathit{right} : & \textbf{boolean where } \mathit{left} = \mathit{right} = 1 \\ \textbf{local} & f : & \textbf{array}[2..k] \textbf{ of boolean where } f = 1 \\ \multicolumn{3}{c}{P(\mathit{left}, f[2]) \parallel P(f[2], f[3]) \parallel \cdots \parallel P(f[k], \mathit{right})} \end{array}\right]$$

for every $k \geq 2$. This means that any $(\mathit{left}, \mathit{right})$-observation of $S[2..k]$ is matched by a corresponding $(\mathit{left}, \mathit{right})$-observation of $\mathcal{I}(\mathit{left}, \mathit{right})$.

We will consider two approaches to the construction of the desired invariant. There is a tradeoff:

A hand-crafted abstraction with simple proof vs.
A natural abstraction with hand-crafted proof.

An Ad-Hoc Hand-Crafted Construction

Observing how the two border members of $S[2..k]$ manipulate the obesrvables $left$ and $right$, led, after experimentation to the following two-halves abstraction:

$$\left[\begin{array}{l} \ell_0 : \textbf{loop forever do} \\ \quad \left[\begin{array}{lll} \ell_1 : & \textbf{Non-Critical} \\ \ell_2 : & \textbf{request } left \\ \ell_3 : & \textbf{idle} \\ \ell_4 : & \textbf{Critical} \\ \ell_5 : & \textbf{release } left \\ \ell_6 : & \textbf{skip} \end{array}\right] \end{array}\right] \quad \| \quad \left[\begin{array}{l} m_0 : \textbf{loop forever do} \\ \quad \left[\begin{array}{lll} m_1 : & \textbf{Non-Critical} \\ m_2 : & \textbf{idle} \\ m_3 : & \textbf{request } right \\ m_4 : & \textbf{Critical} \\ m_5 : & \textbf{skip} \\ m_6 : & \textbf{release } right \end{array}\right] \end{array}\right]$$

additional compassion: $(at\_\ell_3 \ \wedge \ right, \ at\_\ell_4)$

In this case the abstraction conditions were $S[2..3] \sqsubseteq_M \mathcal{I}$ and $(P \,\|\, \mathcal{I}) \sqsubseteq_M \mathcal{I}$.

The additional compassion requirement reflects the possibility that the leftmost process in $S[2..k]$ may deadlock while requesting its right fork, unless the rightmost process can acquire its right fork infinitely often.

Using the Network Invariant

To show that an arbitrary regular philsopoher never starves, it is sufficient to verify

$$(\mathcal{I} \parallel P \parallel \mathcal{I} \parallel R) \quad \models \quad \square\,(at\_\ell_2 \rightarrow \diamondsuit\, at\_\ell_4)$$

where R is a contrary phislopher and the locations in the property refer to P.

Fair Discrete System

A fair discrete system (FDS) $\mathcal{D} = \langle V, \mathcal{O}, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$ consists of:

- V – A finite set of typed state variables. A V-state s is an interpretation of V. Σ_V – the set of all V-states.
- $\mathcal{O} \subseteq V$ – A set of observable variables.
- 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.

Verifying $\mathcal{D}_C \sqsubseteq \mathcal{D}_A$

Consider a concrete and abstract systems:

$\mathcal{D}_C = \langle V_C, \mathcal{O}_C, \Theta_C, \rho_C, \mathcal{J}_C, \mathcal{C}_C \rangle$ and
$\mathcal{D}_A = \langle V_A, \mathcal{O}_A, \Theta_A, \rho_A, \mathcal{J}_A, \mathcal{C}_A \rangle$

With no loss of generality, assume that $V_C \cap V_A = \emptyset$ and that there exists a $1-1$ correspondence between $\mathcal{O}_C = \{x_1, \ldots, x_n\}$ and $\mathcal{O}_A = \{X_1, \ldots, X_n\}$.

The FDS $\mathcal{D}_S = \langle V_S, \mathcal{O}_S, \Theta_S, \rho_S, \mathcal{J}_S, \mathcal{C}_S \rangle$ is called a superposition of $\mathcal{D}_C$ and $\mathcal{D}_A$ if it has the form:

$$
\begin{array}{lcl}
V_S & = & V_C \cup V_A \\
\mathcal{O}_S & = & \mathcal{O}_C \cup \mathcal{O}_A \\
\Theta_S & = & \Theta_C \wedge \Theta_A \wedge \Theta_g \\
\rho_S & = & \rho_C \wedge \rho_A \wedge \rho_g \\
\mathcal{J}_S & = & \mathcal{J}_C \\
\mathcal{C}_S & = & \mathcal{C}_C
\end{array}
$$

for some Θ_g and ρ_g, guiding the abstract system how to choose values for its non-observable vairables.

A Proof Rule

The following rule can be used to prove $\mathcal{D}_C \sqsubseteq \mathcal{D}_A$, using symbolic model checking.

$$\begin{array}{lll}
A1. & \Theta_C & \rightarrow \quad \exists V_A : \Theta_S \\
A2. & \rho_C & \rightarrow \quad \exists V_A' : \rho_S \\
A3. & \mathcal{D}_S \models & \left[\begin{array}{ll} & \Box\,(\mathcal{O}_C = \mathcal{O}_A) \\ \wedge & \bigwedge_{J \in \mathcal{J}_A} \Box \Diamond J \\ \wedge & \bigwedge_{(p,q) \in \mathcal{C}_A} (\Box \Diamond p \rightarrow \Box \Diamond q) \end{array}\right] \\
\hline
 & & \mathcal{D}_C \sqsubseteq \mathcal{D}_A
\end{array}$$

The user has to provide the advisory Θ_g and ρ_g.

Note that since an FDS is essentially a non-deterministic Streett Auomtaton, the problem $\mathcal{D}_C \sqsubseteq \mathcal{D}_A$ can be algorithmically solved. However, the complexity of complementing a general Streett automaton is hopelessly high. This is why we have to resort to a user-guided proof.

In General

We can identify a desired relation between the concrete and abstract variables. This is the well-known abstraction mapping which has the general form

$$\delta : V_A = \alpha(V_C)$$

Among other things, we expect δ to imply $\mathcal{O}_A = \mathcal{O}_C$.

If we succeed to identify the desired relation δ, we usually include in Θ_g the conjunct

$$(\exists V_A : \Theta_A \ \land \ \delta) \rightarrow \delta$$

and in ρ_g, the conjunct

$$(\exists {V_A}' : \rho_C \ \land \ \rho_A \ \land \ \delta') \rightarrow \delta'$$

For Example

To prove $S[1..2] \sqsubseteq_M \mathcal{I}$, we have to establish

$$(P[1] \parallel P[2]) \sqsubseteq_M (LH[3] \parallel RH[4])$$

The desired abstraction mapping is:

$$\delta : \quad \mathit{Left} = \mathit{left} \ \wedge \ \mathit{Right} = \mathit{right} \ \wedge \ \Pi[3] = \pi[1] \ \wedge \ \Pi[4] = \pi[2]$$

This natural mapping is sufficient in order to establish all the abstraction conditions.

The Alternative: A Natural Invariant with Complex Proof

An alternate simpler invariant can be obtained by taking $\mathcal{I} = S[1..3]$, i.e. a string of 3 (unmodified) philosophers.

To prove that this is an invariant, it is sufficient to establish

$$(p[1] \parallel p[2] \parallel p[3] \parallel p[4]) \sqsubseteq_M (P[5] \parallel P[6] \parallel P[7])$$

i.e., that 3 philosophers can faithfully emulate 4 philosophers. What should the abstraction mapping be?

Obviously, $P[5]$ should mimic $p[1]$ and $P[7]$ should mimic $p[4]$. What should $P[6]$ do?

The Alternative: A Natural Invariant with Complex Proof

An alternate simpler invariant can be obtained by taking $\mathcal{I} = S[1..3]$, i.e. a string of 3 (unmodified) philosophers.

To prove that this is an invariant, it is sufficient to establish

$$(p[1] \parallel p[2] \parallel p[3] \parallel p[4]) \sqsubseteq_M (P[5] \parallel P[6] \parallel P[7])$$

i.e., that 3 philosophers can faithfully emulate 4 philosophers. What should the abstraction mapping be?

Obviously, $P[5]$ should mimic $p[1]$ and $P[7]$ should mimic $p[4]$. What should $P[6]$ do?

Answer:
$P[6]$ should remain idle until it finds out that $S[1..4]$ is at a guaranteed deadlock, at which point it should join $P[5]$ and $P[7]$ in order to form a similar deadlock.

This requires the capability of clairvoyance, which has been implemented within TLV.

Clairvoyance in Action

In file p4refsp3.smv, we place:

```
MODULE main
VAR  Cs : system Concrete; -- Concrete
     As : system Abstract; -- Abstract
DEFINE
 deadlock:= Cs.p1.loc=2 & Cs.p2.loc=2 & Cs.p3.loc=2 & Cs.p4.loc=2
         & !Cs.mid1 & !Cs.mid2 & !Cs.mid3 & !Cs.right;
  guaranteed : []deadlock;
MODULE Concrete
VAR   left,right : boolean;
      mid1,mid2,mid3: boolean own;
      p1: Phil(left,mid1);
      p2: Phil(mid1,mid2);
      p3: Phil(mid2,mid3);
      p4: Phil(mid3,right);
MODULE Abstract
VAR   Left,Right: Boolean;
      mid1,mid2 : boolean own;
      P5 : process Phil(Left,mid1);
      P6:  process Phil(mid1,mid2);
      P7:  process Phil(mid2,Right);
    ...
```

Guidance

In file p4refsp3.pf, we place:

```
Let theg := 1; -- \The_d
Let desire := (next(P5.loc) = next(p1.loc)) &
              (next(P7.loc) = next(p4.loc)) &
 (next(P6.loc) = case
    (next(p1.loc) != p1.loc) | (next(p4.loc) != p4.loc) : P6.loc;
    P6.loc = 0 & !next(guaranteed)                       : 0;
    P6.loc = 0 & next(guaranteed)                        : 1;
    P6.loc = 1 & As.mid1                                 : 2;
    P6.loc = 2 & As.mid2                                 : 3;
    P6.loc = 3                                           : 4;
    P6.loc = 4                                           : 5;
    P6.loc = 5                                           : 0;
    1                                                    : P6.loc;
                 esac);
Let rhog := ((rho_a & desire) forsome prime(Va)) -> desire;
Let corr := (Left = left) & (Right = right)  ; -- correspondence
fabs;
```

Application to Probabilistic Parameterized Systems

Consider the Courteous Philosophers' Protocol by Lehman and Rabin [LR81]:

$$
\begin{array}{llll}
 & \textbf{in} & N: & \textbf{integer where } N \geq 2 \\
 & \textbf{local} & signl, y,: & \textbf{array } [1..N] \textbf{ of boolean init } 0 \\
 & \textbf{local} & lastL: & \textbf{array } [1..N] \textbf{ of } \{-1,0,1\} \textbf{ init } -1 \\
\overset{N}{\underset{i=1}{\|}} P[i] :: &
\multicolumn{3}{l}{\left[\begin{array}{l}
\textbf{loop forever do} \\
\left[\begin{array}{l}
\ell_0 : \textbf{Non-Critical} \\
\ell_1 : signl[i] := 1; \quad \textbf{go to } \{0.5 : \ell_2;\ 0.5 : \ell_5\} \\
\ell_2 : \textbf{when } \neg y[i] \ \wedge\ (\neg signl[i \ominus 1] \ \vee\ lastL[i] = 1) \textbf{ do } y[i] := 1 \\
\ell_3 : \textbf{If } y[i \oplus 1] = 0 \textbf{ then } [y[i \oplus 1] := 1;\ \textbf{go to } \ell_8] \\
\ell_4 : y[i] := 0; \textbf{go to } \ell_1 \\
\ell_5 : \textbf{when } \neg y[i \oplus 1] \ \wedge\ (\neg signl[i \oplus 1] \ \vee\ lastL[i \oplus 1] = 0) \\
\qquad \textbf{do } y[i \oplus 1] := 1 \\
\ell_6 : \textbf{If } y[i] = 0 \textbf{ then } [y[i] := 1;\ \textbf{go to } \ell_8] \\
\ell_7 : y[i \oplus 1] := 0;\ \textbf{go to } \ell_1 \\
\ell_8 : \textbf{Critical} \\
\qquad signl[i] := 0; \quad lastL[i] := 0; \quad lastL[i \oplus 1] := 1 \\
\qquad y[i] := 0; \quad y[i \oplus 1] := 0
\end{array}\right]
\end{array}\right]}
\end{array}
$$

Step 1: Eliminate Variables Depending on Locations

$$
\begin{array}{l}
\quad\quad\quad\quad \textbf{in} \quad N : \quad \textbf{integer where } N \geq 2 \\
\quad\quad\quad\quad \textbf{local} \quad lastL : \quad \textbf{array } [1..N] \textbf{ of } \{-1, 0, 1\} \textbf{ init } -1 \\
\overset{N}{\underset{i=1}{\|}} P[i] :: \left[\begin{array}{l}
\textbf{loop forever do} \\
\left[\begin{array}{ll}
\ell_0 : & \textbf{Non-Critical} \\
\ell_1 : & \textbf{go to } \{0.5 : \ell_2;\ 0.5 : \ell_5\} \\
\ell_2 : & \textbf{await } at\_\ell_0[i \ominus 1] \ \vee \ at\_\ell_{0..5}[i \ominus 1] \wedge (lastL[i] \neq 0) \\
\ell_3 : & \textbf{If } at\_\ell_{1,2,5..7}[i \oplus 1] \textbf{ then go to } \ell_8 \\
\ell_4 : & \textbf{go to } \ell_1 \\
\ell_5 : & \textbf{await } at\_\ell_0[i \oplus 1] \ \vee \ at\_\ell_{1,2,5..7}[i \oplus 1] \wedge (lastL[i] \neq 1) \\
\ell_6 : & \textbf{If } at\_\ell_{0..5}[i \ominus 1] \textbf{ then go to } \ell_8 \\
\ell_7 : & \textbf{go to } \ell_1 \\
\ell_8 : & \textbf{Critical};\ lastL[i] := 0;\ lastL[i \oplus 1] := 1
\end{array}\right]
\end{array}\right]
\end{array}
$$

Wish to prove accessibility to location ℓ_8 with probability 1.

Replace Probability by Compassion

Transform problem to

$$
\begin{array}{l}
\begin{array}{lll}
\textbf{in} & N: & \textbf{integer where } N \geq 2 \\
\textbf{local} & lastL: & \textbf{array } [1..N] \textbf{ of } \{-1, 0, 1\} \textbf{ init } -1
\end{array} \\
\overset{N}{\underset{i=1}{\|}} P[i] :: \left[\begin{array}{l}
\textbf{loop forever do} \\
\left[\begin{array}{ll}
\ell_0: & \textbf{Non-Critical} \\
\ell_1: & \textbf{go to } \{\ell_2;\ \ell_5\} \\
\ell_2: & \textbf{await } at\_\ell_0[i \ominus 1] \ \vee\ at\_\ell_{0..5}[i \ominus 1] \wedge (lastL[i] \neq 0) \\
\ell_3: & \textbf{If } at\_\ell_{1,2,5..7}[i \oplus 1] \textbf{ then go to } \ell_8 \\
\ell_4: & \textbf{go to } \ell_1 \\
\ell_5: & \textbf{await } at\_\ell_0[i \oplus 1] \ \vee\ at\_\ell_{1,2,5..7}[i \oplus 1] \wedge (lastL[i] \neq 1) \\
\ell_6: & \textbf{If } at\_\ell_{0..5}[i \ominus 1] \textbf{ then go to } \ell_8 \\
\ell_7: & \textbf{go to } \ell_1 \\
\ell_8: & \textbf{Critical};\ lastL[i] := 0;\ lastL[i \oplus 1] := 1
\end{array}\right]
\end{array}\right]
\end{array}
$$

with additional compassion requirements:

$$\langle at\_\ell_1[i] \ \wedge\ cond,\ at\_\ell_2[i] \ \wedge\ cond\rangle, \quad \langle at\_\ell_1[i] \ \wedge\ cond,\ at\_\ell_5[i] \ \wedge\ cond\rangle$$

for every $cond$ in the set

$$\left\{\begin{array}{l}
at_\ell_{0,1,8}[i \ominus 1], at_\ell_{2,3}[i \ominus 1], at_\ell_4[i \ominus 1], at_\ell_{5,6}[i \ominus 1], at_\ell_7[i \ominus 1] \\
at_\ell_{0,1,8}[i \oplus 1], at_\ell_{2,3}[i \oplus 1], at_\ell_4[i \oplus 1], at_\ell_{5,6}[i \oplus 1], at_\ell_7[i \oplus 1]
\end{array}\right\}$$

The Two-Halves Abstraction

$$
\begin{array}{lll}
\textbf{in} & elloc, erloc: & [0..8] \\
\textbf{in-out} & llst, erlst: & [-1..1] \ \textbf{where}\ llst, erlst = -1 \\
\textbf{out} & lloc, rloc: & [0..8] \ \textbf{where}\ lloc, rloc = 0
\end{array}
$$

$$
\left[\begin{array}{l}
\textbf{loop forever do} \\
\quad \left[\begin{array}{l}
\ell_0 : \textbf{non-critical} \\
\ell_1 : \textbf{goto}\ \{\ell_2,\ \ell_5\} \\
\ell_2 : \textbf{await}\ elloc = 0 \lor \\
\quad elloc \in \{0..5\} \land llst \neq 0 \\
\ell_3 : \textbf{go to}\ \{\ell_4,\ \ell_8\} \\
\ell_4 : \textbf{go to}\ \ell_1 \\
\ell_5 : \textbf{skip} \\
\ell_6 : \textbf{if}\ elloc \in \{0..5\}\ \textbf{then} \\
\qquad\qquad \textbf{go to}\ \ell_8 \\
\ell_7 : \textbf{go to}\ \ell_1 \\
\ell_8 : \textbf{Critical};\ llst := 0
\end{array}\right]
\end{array}\right]
\ \Big\|\
\left[\begin{array}{l}
\textbf{loop forever do} \\
\quad \left[\begin{array}{l}
m_0 : \textbf{non-critical} \\
m_1 : \textbf{goto}\ \{m_2,\ m_5\} \\
m_2 : \textbf{skip} \\
m_3 : \textbf{if}\ erloc \in \{1, 2, 5..7\}\ \textbf{then} \\
\qquad\qquad \textbf{go to}\ m_8 \\
m_4 : \textbf{go to}\ m_1 \\
m_5 : \textbf{await}\ erloc = 0 \lor \\
\quad erloc \in \{1, 2, 5..7\} \land erlst \neq 1 \\
m_6 : \textbf{go to}\ \{m_7,\ m_8\} \\
m_7 : \textbf{go to}\ m_1 \\
m_8 : \textbf{Critical};\ erlst := 1
\end{array}\right]
\end{array}\right]
$$

Extra Compassion: $(at\_\ell_{2..4}\ \land\ at\_m_{5..7},\quad at\_\ell_8\ \lor\ at\_m_8)$