In PVS

Temporal Operators

\[
\begin{align*}
(f \land g)'(d) & \iff (f' \land g')(d) \\
(f \lor g)'(d) & \iff (f' \lor g')(d) \\
\neg f'(d) & \iff \neg (f'(d)) \\
\forall t \in \mathbb{N}: \text{State} & \iff \text{State}'(d) \\
\exists t \in \mathbb{N}: \text{State} & \iff \text{State}'(d)
\end{align*}
\]

---

Example

Consider states sequence SS defined by the first 2 columns:

<table>
<thead>
<tr>
<th>STATE</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLUE</td>
<td>0</td>
</tr>
<tr>
<td>RED</td>
<td>1</td>
</tr>
<tr>
<td>BLUE</td>
<td>0</td>
</tr>
<tr>
<td>RED</td>
<td>1</td>
</tr>
<tr>
<td>RED</td>
<td>1</td>
</tr>
<tr>
<td>BLUE</td>
<td>0</td>
</tr>
<tr>
<td>RED</td>
<td>0</td>
</tr>
</tbody>
</table>

Temporal properties depend only on the state. Let-state (SS,i) = BLUE.

---

Disjunction, conjunction, negation and implication over assertions are defined.

\[
\begin{align*}
\text{ASSERTION} & \iff \text{ASSERTION}'(d) \\
\text{ASSERTION} & \iff \text{ASSERTION}'(d)
\end{align*}
\]

For the TLT operators:

\[
\begin{align*}
(f \land g)'(d) & \iff (f' \land g')(d) \\
(f \lor g)'(d) & \iff (f' \lor g')(d) \\
\neg f'(d) & \iff \neg (f'(d)) \\
\forall t \in \mathbb{N}: \text{State} & \iff \text{State}'(d) \\
\exists t \in \mathbb{N}: \text{State} & \iff \text{State}'(d)
\end{align*}
\]

We denote the notion of a temporal property holding at position \(d\) by \(P_d\).

Let \(S\) be a state sequence.

In the natural manner:

\[
(f \land g)'(d) \iff (f' \land g')(d)
\]

Temporal properties are defined on individual states, without reference to their position in the state sequence. Assertions are properties defined on individual states, without reference to their position in the state sequence.

\[
\begin{align*}
\text{ASSERTION} & \iff \text{ASSERTION}'(d) \\
\text{ASSERTION} & \iff \text{ASSERTION}'(d)
\end{align*}
\]
Compassion: For every \( i \in \text{TRANSITION-DOMAIN} \), there are infinitely many \( i \)-states in \( \text{seg} \), such that for all \( j \), there are infinitely many \( j \)-states in \( \text{seg} \).

Justice: For every \( i \in \text{TRANSITION-DOMAIN} \), there are infinitely many \( i \)-states in \( \text{seg} \), such that for all \( j \), there are infinitely many \( j \)-states in \( \text{seg} \).

Runs and Computations

A run \( \text{seq} \) of \( \text{pfs} \) is an initialized run if it satisfies

- Initiality: \( \text{seq}(0) \) is initial, i.e., \( \text{seg}(0) \) is a successor of \( \text{seg}(0) \).
- Consecution: For every \( t = 0, 1, 2 \), state \( \text{seg}(t + 1) \) is a successor of \( \text{seg}(t) \).
- A run \( \text{seq} \) of \( \text{pfs} \) is initialized with the fairness requirements

where

- \( \text{ Со } \text{) : The compassion requirements, each of the form \( d \) \) \.
- \( \text{ Ю } \text{) : The justice requirements, each of the form \( d \) \)
- \( \text{ С } \text{) : The transition relation, a predicate \( (d, \text{seg}, \text{seg}) \) \) returning to the both updated (current and previous) versions of state variables.
- \( \text{ Е } \text{) : The initial condition, an assertion characterizing the initial states.

END temporary operators

There is no state-variables (\( y \)) component

STATE and TRANSITION-DOMAIN parameters are given in defining the PFS
A temporal property

Validity

Sometimes having the stronger property is useful.

Rules like BINV actually prove the stronger \(\text{F-valid}\) property.

Generally, interested in \(\text{F-valid}\).

\(\text{F-valid} \Rightarrow \text{F-reachable valid}\)

and so

State sequence \(\Rightarrow\) initialized runs \(\Rightarrow\) computations.

Valiidy clc
In MUX-SEM, we could have defined

**Transition domains**

- Very often, as was the case in MUX-SEM, the transition domain is comprised of a *location* and *processor identifier* field.

**TRANS-DOMAIN** theory, which defines such a transition domain,

**IMPORTING** TRANS-DOMAIN[5, N] creates and imports the following definitions:

```
LOCATIONS: TYPE = unto[PROG_SIZE-1]
PROC_ID: TYPE = unto[N]
TRANS-DOMAIN: TYPE = [* # loc: LOCATION, pid: PROC_ID]
```

**END mu_xml**
Figure 3: Parameterized mutual exclusion algorithm

Bakery

Example: Bakery
Theorem: \[
\text{TLPVS} \equiv \text{PFS} = (\# \text{initial} = \text{fst} \lor \text{pfs} : \text{y}(\text{st})(\text{p}) = 0 \land \text{loc}(\text{st})(\text{p}) = 0, \text{rho} = \text{rho}, \text{justice} = \text{justice}, \text{compassion} = \text{empty compassion} \#)
\]

**Definition**

Deductive Verification of Reactive Systems, NYU, Fall, 2007

**Lecture 5:**

Proving properties of BAKERY

**Proving properties of BAKERY**

\[
\text{yZero}::= \begin{cases}
\text{xZero} & \text{IF} \quad \text{loc}(\text{current!1})(\text{p!1}) = 0 \land \text{y}(\text{next!1}) = \text{y}(\text{current!1}) \land \text{loc}(\text{next!1}) = \text{loc}(\text{current!1}) \\
& \text{WITH} \quad [(\text{p!1}) := 1]
\end{cases}
\]

\[
\text{yZero}::= \begin{cases}
\text{xZero} & \text{IF} \quad \text{loc}(\text{current!1})(\text{p!1}) = 1 \land (\exists \text{m:} \text{nat} : (\forall \text{q:PROC_ID} : \text{y}(\text{current!1})(\text{q}) < \text{m}) \land \text{y}(\text{next!1}) = \text{y}(\text{current!1}) \land \text{loc}(\text{next!1}) = \text{loc}(\text{current!1}) \land [(\text{p!1}) := \text{m}])
\end{cases}
\]

\[
\text{yZero}::= \begin{cases}
\text{xZero} & \text{IF} \quad \text{loc}(\text{current!1})(\text{p!1}) = 2 \land (\forall \text{q:} \text{q} \neq \text{p!1} \implies \text{y}(\text{current!1})(\text{q}) = 0 \lor \text{y}(\text{current!1})(\text{p!1}) \leq \text{y}(\text{current!1})(\text{q})) \land \text{y}(\text{next!1}) = \text{y}(\text{current!1}) \land \text{loc}(\text{next!1}) = \text{loc}(\text{current!1}) \land [(\text{p!1}) := 3])
\end{cases}
\]

\[
\text{yZero}::= \begin{cases}
\text{xZero} & \text{IF} \quad \text{loc}(\text{current!1})(\text{p!1}) = 3 \land \text{y}(\text{next!1}) = \text{y}(\text{current!1}) \land \text{loc}(\text{next!1}) = \text{loc}(\text{current!1}) \land [(\text{p!1}) := 4])
\end{cases}
\]

\[
\text{yZero}::= \begin{cases}
\text{xZero} & \text{IF} \quad \text{loc}(\text{current!1})(\text{p!1}) = 4 \land \text{y}(\text{next!1}) = \text{y}(\text{current!1}) \land \text{loc}(\text{next!1}) = \text{loc}(\text{current!1}) \land [(\text{p!1}) := 0])
\end{cases}
\]
ENDIF

IF y((current!1)(i)) = 0
THEN
  (\forall (i:PROC_ID) \{ (current!1)(i) = 0 \iff (loc(current!1)(i) = 0 \lor loc(current!1)(i) = 1) \})

|-------

\{-1,(\rho)\} IF y((current!1)(i!1)) = 0 THEN (loc(current!1) WITH [(p!1) := 1](i!1) = 0 \lor loc(current!1) WITH [(p!1) := 1](i!1) = 1) ELSE NOT(loc(current!1) WITH [(p!1) := 1](i!1) = 0 \lor loc(current!1) WITH [(p!1) := 1](i!1) = 1) ENDIF

RULE? (split-rho)

Deductive Verification of Reactive Systems, NYU, Fall, 2007
This completes the proof of \( y = 0 \).

\[ \neg \text{(split-all-inst(\text{"i!1"}))} \]

This completes the proof of \( y = 0 \).

\[ \text{END} \]

Rule (split-all-inst("i!1"))

This completes the proof of \( y = 0 \).
FORALL (i:PROC_ID):
(y(current!1)(i) = 0IFF (loc(current!1)(i) = 0 OR loc(current!1)(i) = 1))

|------- |

{1,(rtp)}(y(next!1)(i!1) = 0IFF (loc(next!1)(i!1) = 0 OR loc(next!1)(i!1) = 1))

RULE?(split-rho-all("i!1"))

Q.E.D.

For Example

```
\exists i. \forall n, \exists j, \exists k, \exists m. \
\exists i. \forall n, \exists j, \exists k, \exists m. \
\exists i. \forall n, \exists j, \exists k, \exists m. \
\exists i. \forall n, \exists j, \exists k, \exists m. \
\exists i. \forall n, \exists j, \exists k, \exists m. 
```

Deductive Verification of Reactive Systems, NYU, Fall, 2007

The Right Way of Using PVS

where x-z, x-Z, ..., x-n are all the free variables currently appearing in the

sequential

relation. use the standard strategy

Then, whenever encountering a verification condition which includes a transition

rather than

multiply quantified formulas as

To maximize the utility of the TLPVS package, it is recommended to structure all

avoid getting into deal with interaction issues.

PVS. Moreover, the user who is mainly interested in verifying programs, should

The previous pages showed some of the options and capabilities provided by

Deductive Verification of Reactive Systems, NYU, Fall, 2007

The Right Way of Using PVS
A. Pnueli

\{1, (rtp)\}

(y_{\text{next!1}}(i!1) = 0 \iff \text{loc}_{\text{next!1}}(i!1) = 0 \lor \text{loc}_{\text{next!1}}(i!1) = 1)

Rule? (\text{split-rho-all}("i!1";"p!1"))... Tri to complete a proof using \text{split-rho} and then if this fails does \text{simple simplification}.

...