



Specification and Verification of Parallel Programs



Augsburg, 28.11.2001, Michael Balsler

Contents



1. Specification

- (a) State
- (b) Interval
- (c) Parallel Programs
- (d) Temporal Logic

2. Verification

- (a) Predicate Logic
- (b) Symbolic Execution
- (c) Trace Induction
- (d) Modular Proofs
- (e) Automation

Specification – State



State σ assigns values to variables.

x, y, n, \dots value of flexible variables

x', y', n', \dots value of primed variables

x'', y'', n'', \dots value of doubly primed variables

X, N, M, \dots value of rigid variables independent of state

Predicate Logic

- Properties in full first order predicate logic.
- Functions and predicates are rigid

$$x = 0 \wedge \text{even}(x)$$

$$25 \leq a \wedge a \leq 48 \wedge l < 12$$

$$\rightarrow \text{get-bilirubin}(\text{set-age-level}(tsb, a, l)) = \text{observation}$$

Specification – Traces

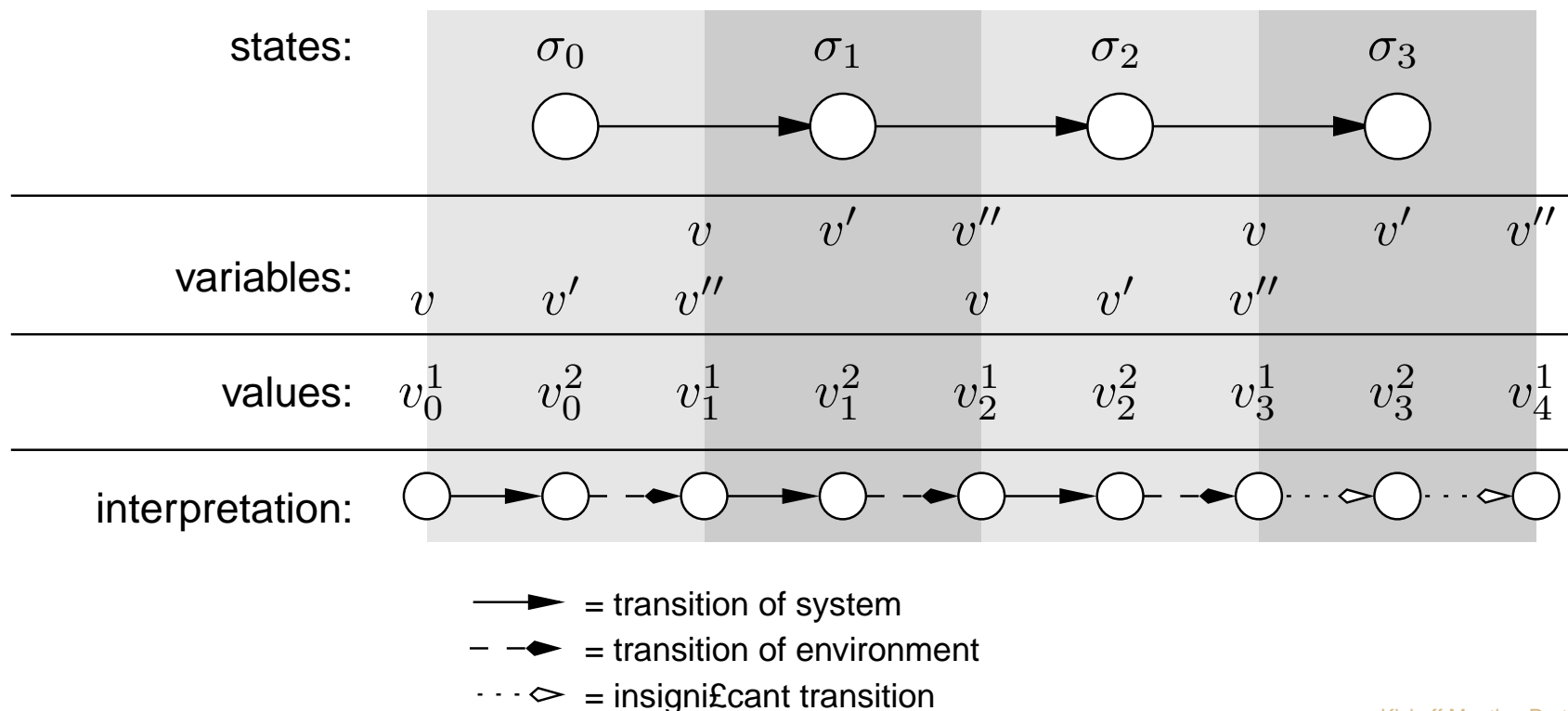


Traces are finite or infinite sequences of states

$$(\sigma_0, \sigma_1, \dots)$$

Doubly primed variables equal to unprimed variables of next state

Interpretation



Specification – Parallel Programs



Parallel programs describe transition systems

Example 1

```
while true do begin  
     $p\#(; s); l_1 := \text{true}; l_1 := \text{false}; v\#(; s)$   
end  
|| while true do begin  
     $p\#(; s); l_2 := \text{true}; l_2 := \text{false}; v\#(; s)$   
end
```

Example 2

```
while  $n < m$  do  $n := n + 1$ 
```

Specification – Parallel Programs



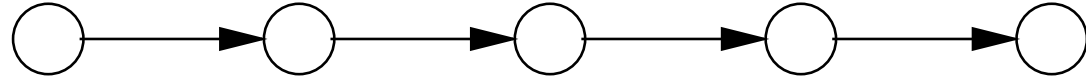
Standard program constructs

Assignment	$x := \tau$
Random assignment	$x := ?$
Skip	skip
Abort	abort
Composition	$\alpha; \beta$
Conditional	if ε then α else β
Loop	while ε do α
Interleaving	$\alpha \parallel \beta$
Synchronization	await ε
Blocking	$\{\alpha\}$

Specification – Temporal Logic



Important operators



- $\square \varphi$ φ always holds from now on
- $\diamond \varphi$ φ now or eventually later
- φ **until** ψ eventually ψ holds, until then φ holds
- φ **unless** ψ as long as ψ does not hold, φ holds
- $\circ \varphi$ there is a next state which satisfies φ
- $\bullet \varphi$ if there is a next state, it satisfies φ
- last** current state is last state
- $\varphi; \psi$ first part of interval satisfies φ , second part ψ

Example

get-bilirubin(*tsb*) = observation **until last**

Verification – Basics



Sequent

$$\begin{aligned} & \Gamma \vdash \Delta \\ \equiv & \varphi_1, \dots, \varphi_n \vdash \psi_1, \dots, \psi_n \\ \equiv & \varphi_1 \wedge \dots \wedge \varphi_n \rightarrow \psi_1 \vee \dots \vee \psi_n \end{aligned}$$

Γ antecedent
 Δ succedent

Sequent calculus

$$\frac{\Gamma_1 \vdash \Delta_1 \quad \dots \quad \Gamma_n \vdash \Delta_n}{\Gamma \vdash \Delta} \textit{rule}$$

$\Gamma_i \vdash \Delta_i$ premise
 $\Gamma \vdash \Delta$ conclusio

Verification – Predicate Logic



$$\frac{\varphi, \psi, \Gamma \vdash \Delta}{\varphi \wedge \psi, \Gamma \vdash \Delta} \text{ con left}$$

$$\frac{\Gamma \vdash \varphi, \Delta \quad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \wedge \psi, \Delta} \text{ con right}$$

$$\frac{\Gamma \vdash \varphi, \Delta \quad \psi, \Gamma \vdash \Delta}{\varphi \rightarrow \psi, \Gamma \vdash \Delta} \text{ imp left}$$

$$\frac{\varphi, \Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \rightarrow \psi, \Delta} \text{ imp right}$$

$$\frac{}{\varphi, \Gamma \vdash \varphi, \Delta} \text{ axiom}$$

Example

$$pc = 0 \rightarrow \text{even}(x), pc = 0 \vdash pc = 0 \wedge \text{even}(x)$$

Verification – Predicate Logic



$$\frac{\varphi, \psi, \Gamma \vdash \Delta}{\varphi \wedge \psi, \Gamma \vdash \Delta} \textit{ con left} \qquad \frac{\Gamma \vdash \varphi, \Delta \quad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \wedge \psi, \Delta} \textit{ con right}$$

$$\frac{\Gamma \vdash \varphi, \Delta \quad \psi, \Gamma \vdash \Delta}{\varphi \rightarrow \psi, \Gamma \vdash \Delta} \textit{ imp left} \qquad \frac{\varphi, \Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \rightarrow \psi, \Delta} \textit{ imp right}$$

$$\frac{}{\varphi, \Gamma \vdash \varphi, \Delta} \textit{ axiom}$$

Example

$$\frac{\dots, pc = 0 \vdash pc = 0 \quad pc = 0 \rightarrow \text{even}(x), pc = 0 \vdash \text{even}(x)}{pc = 0 \rightarrow \text{even}(x), pc = 0 \vdash pc = 0 \wedge \text{even}(x)}$$

Verification – Predicate Logic



$$\frac{\varphi, \psi, \Gamma \vdash \Delta}{\varphi \wedge \psi, \Gamma \vdash \Delta} \text{ con left} \qquad \frac{\Gamma \vdash \varphi, \Delta \quad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \wedge \psi, \Delta} \text{ con right}$$

$$\frac{\Gamma \vdash \varphi, \Delta \quad \psi, \Gamma \vdash \Delta}{\varphi \rightarrow \psi, \Gamma \vdash \Delta} \text{ imp left} \qquad \frac{\varphi, \Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \rightarrow \psi, \Delta} \text{ imp right}$$

$$\frac{}{\varphi, \Gamma \vdash \varphi, \Delta} \text{ axiom}$$

Example

$$\frac{\dots, pc = 0 \vdash pc = 0 \quad \frac{pc = 0 \vdash pc = 0, \dots \quad \dots}{pc = 0 \rightarrow \text{even}(x), pc = 0 \vdash \text{even}(x)}}{pc = 0 \rightarrow \text{even}(x), pc = 0 \vdash pc = 0 \wedge \text{even}(x)}$$

Verification – Principle



$n = 0, \mathbf{while} \ n < m \ \mathbf{do} \ n := n + 1,$

$\square \ n'' = n' \wedge m'' = m' \vdash \diamond \ n = m$

- Symbolic Execution

1. Unwind operators until sequent in normal form

$n = 0, n < m, n' = 1, n'' = 1, m'' = m, \bullet \ \mathbf{while} \ n < m \ \mathbf{do} \ n := n$

$\bullet \ \square \ n'' = n' \wedge m'' = m' \vdash \circ \ \diamond \ n = m$

2. Advance a step in the trace

$n_0 = 0, n = 1, n \leq m, \mathbf{while} \ n < m \ \mathbf{do} \ n := n + 1,$

$\square \ n'' = n' \wedge m'' = m' \vdash \diamond \ n = m$

- Trace Induction

- ★ Generalisation $n = 0 \Rightarrow n \leq m$

- ★ Induction Term $m - n$

Verification – Unwinding Programs (1)



Rule examples

$$\frac{\varepsilon, \alpha; \mathbf{while} \ \varepsilon \ \mathbf{do} \ \alpha, \Gamma \vdash \mathbf{last}, \Delta \quad \mathbf{last}, \Gamma \vdash \varepsilon, \Delta}{\mathbf{while} \ \varepsilon \ \mathbf{do} \ \alpha, \Gamma \vdash \Delta} \textit{while left}$$

$$\frac{x := \tau, \circ \alpha, \Gamma \vdash \Delta}{x := \tau; \alpha, \Gamma \vdash \Delta} \textit{chop left}^*$$

$$(1) \quad x_1 := \tau_1, \circ (\alpha \parallel (x_2 := \tau_2; \beta)), \Gamma \vdash \Delta$$

$$(2) \quad x_2 := \tau_2, \circ ((x_1 := \tau_1; \alpha) \parallel \beta), \Gamma \vdash \Delta$$

$$\frac{(1) \quad x_1 := \tau_1, \circ (\alpha \parallel (x_2 := \tau_2; \beta)), \Gamma \vdash \Delta \quad (2) \quad x_2 := \tau_2, \circ ((x_1 := \tau_1; \alpha) \parallel \beta), \Gamma \vdash \Delta}{(x_1 := \tau_1; \alpha) \parallel (x_2 := \tau_2; \beta), \Gamma \vdash \Delta} \textit{interleaving left}^*$$

* simplified version

Verification – Unwinding Programs (2)



Example

$$\frac{\frac{n < m, n = 0, n' = 1, m' = m, \circ \mathbf{while} \dots, \square \dots \vdash \diamond \dots}{n := n + 1, n < m, n = 0, \circ \mathbf{while} \dots, \square \dots \vdash \diamond \dots} (3)}{n < m, n = 0, n := n + 1; \mathbf{while} \dots, \square \dots \vdash \diamond \dots} (2) \dots \frac{\dots}{n = 0, \mathbf{while} \ n < m \ \mathbf{do} \ n := n + 1, \square \dots \vdash \diamond \dots} (1)$$

(1) *while left*

(2) *chop left **

(3) *assign left **

Verification – Unwinding TL (1)



Recursive equations

$$\begin{aligned}\Box \varphi &\leftrightarrow \varphi \wedge \bullet \Box \varphi \\ \varphi \text{ until } \psi &\leftrightarrow \psi \vee \varphi \wedge \circ \varphi \text{ until } \psi\end{aligned}$$

Rule examples

$$\frac{\varphi, \bullet \Box \varphi, \Gamma \vdash \Delta}{\Box \varphi, \Gamma \vdash \Delta} \text{ always left}$$

$$\frac{\Gamma \vdash \varphi, \Delta \quad \Gamma \vdash \bullet \Box \varphi, \Delta}{\Gamma \vdash \Box \varphi, \Delta} \text{ always right}$$

$$\frac{\psi, \Gamma \vdash \Delta \quad \varphi, \circ (\varphi \text{ until } \psi), \Gamma \vdash \Delta}{\varphi \text{ until } \psi, \Gamma \vdash \Delta} \text{ until left}$$

Verification – Unwinding TL (2)



$$\frac{n'' = n', m'' = m', \dots, \bullet \text{ while } \dots, \bullet \square \dots \vdash \text{last}, n = m, \circ \diamond \dots}{n'' = n', m'' = m', \dots, \circ \text{ while } \dots, \bullet \square \dots \vdash n = m, \circ \diamond n = m} \quad (3)$$
$$\frac{n'' = n', m'' = m', \dots, \circ \text{ while } \dots, \bullet \square \dots \vdash \diamond n = m}{\dots, \circ \text{ while } \dots, \square n'' = n' \wedge m'' = m' \vdash \diamond n = m} \quad (2)$$
$$\dots, \circ \text{ while } \dots, \square n'' = n' \wedge m'' = m' \vdash \diamond n = m \quad (1)$$

(1) *always left*

(2) *eventually right*

(3) *strong next left*

Final, simplified premise

$n = 0, n < m, n' = 1, n'' = 1, m'' = m,$

$\bullet \text{ while } n < m \text{ do } n := n + 1,$

$\bullet \square n'' = n' \wedge m'' = m'$

$\vdash \text{last}, \circ \diamond n = m$

Verification – Step (1)



$$\frac{\mathbf{last}, \gamma \vdash \delta \quad \gamma_1, \Gamma \vdash \delta_1, \Delta}{\gamma, \bullet \Gamma \vdash \delta, \circ \Delta} \textit{step}$$

where $\gamma_1 := \gamma[V \leftarrow V_0][V' \leftarrow V_1][V'' \leftarrow V][\mathbf{last} \leftarrow \mathbf{false}]$,

$\delta_1 := \dots$

with new variables V_0 and V_1

Verification – Step (2)



Example

(1) **last**, $n = 0, n < m, n' = 1, n'' = 1, m'' = m \vdash$ **last**

(2) $n_0 = 0, n_0 < m_0, n_1 = 1, n = 1, m = m_0,$

while $n < m$ **do** $n := n + 1,$

$\square n'' = n' \wedge m'' = m'$

\vdash **false**, $\diamond n = m$

step

$n = 0, n < m, n' = 1, n'' = 1, m'' = m,$

• **while** $n < m$ **do** $n := n + 1,$

• $\square n'' = n' \wedge m'' = m'$

\vdash **last**, $\circ \diamond n = m$

Verification – Trace Induction (1)



- *Generalisation*: Find a property which holds before and after execution of one loop

$$n \leq m$$

(Rules to apply: *cut formula* and *weakening*)

- *Induction Term*: Find an expression τ which disallows infinite repetitions

$$\tau := m - n$$

Rule

$$\frac{N = \tau, \Gamma, \bullet \square \text{IndHyp} \vdash \Delta}{\Gamma \vdash \Delta} \text{ trace induction}$$

where $\text{IndHyp} := (\tau < N \rightarrow (\Gamma \rightarrow \Delta))$

Verification – Trace Induction (2)



Example

$n = 0, \mathbf{while} \ n < m \ \mathbf{do} \ n := n + 1, \square \ \dots \vdash \diamond \ n = m$

$$\frac{\dots}{\frac{\frac{\frac{N = m - n, n \leq m, \mathbf{while} \ \dots, \square \ \dots, \bullet \ \square \ \text{IndHyp} \vdash \diamond \ \dots}{n \leq m, \mathbf{while} \ \dots, \square \ \dots \vdash \diamond \ \dots} (2)}{n = 0, n \leq m, \mathbf{while} \ \dots, \square \ \dots \vdash \diamond \ \dots} (1)}}{n = 0, \mathbf{while} \ \dots, \square \ \dots \vdash \diamond \ \dots} (3)$$

where $\text{IndHyp} := (m - n < N \rightarrow (\Gamma \rightarrow \Delta))$

(1) *cut formula* $n \leq m$

(2) *weakening*

(3) *trace induction* $m - n$

Verification – Automation



Heuristics

tl simple applies noncritical rules

tl apply induction applies induction, if possible

tl analogy searches proof for similar situations

tl induction starts induction (no generalisation, no induction term)

tl unwind unwinds noncritical *tl* operators

tl case distinction unwinds *tl* operators leading to case distinctions

tl step advances a step