

Temporal Logic and State Systems

Prof. Dr. Fred Kröger

Verification of Concurrent Programs: Programs as Transition Systems

Concurrent Programs

(Concurrent) programs can be viewed as STS. They are built with the **parallel statement (PS)** **cobegin** $\Pi_1 \parallel \dots \parallel \Pi_p$ **coend** where Π_1, \dots, Π_p are programs, called **processes**. For simplicity, processes are **sequential**.

Interleaving model (Basic principle for operational semantics)

- The statements contained in the parallel statement are composed of distinguished **atomic steps**,
- an execution sequence of the parallel statement is created by sequentially (i.e., one after the other) executing single atomic steps such that steps belonging to different processes occur in arbitrary order but steps of the same process Π_i are executed in the order given by Π_i .

Example(Concurrent Program):

$$\Pi \equiv \mathbf{cobegin} \alpha; \beta \parallel \gamma; \delta \mathbf{coend}$$

with: $\alpha : b := b * 2, \beta : V(a), \gamma : b := b + 1, \delta : P(a)$

at program start: $a = 0$ (P and V are the usual **semaphore operations**.)

$$\Pi \text{ as rfSTS } \Pi = (X_{\Pi}, V_{\Pi}, S_{\Pi}, T_{\Pi}, Act_{\Pi}, start_{\Pi}, enabled_{\lambda})$$

(with $SIG_{\Pi} = (\{NAT\}, \dots)$, $|S_{\Pi}| = \mathbb{N}_0$ and “standard interpretation” S_{Π}):

$$Act_{\Pi} = \{\alpha, \beta, \gamma, \delta\}, X_{\Pi} = \{a, b\}, V_{\Pi} = \{exec\alpha, exec\beta, exec\gamma, exec\delta, at\alpha, at\beta, at\gamma, at\delta\}$$

($at\lambda$: “ λ is one of the next possible atomic steps”, “program counter is at λ ”)

$$start_{\Pi} \equiv at\alpha \wedge at\gamma \wedge a = 0$$

$$enabled_{\alpha} \equiv at\alpha, enabled_{\beta} \equiv at\beta, enabled_{\gamma} \equiv at\gamma, enabled_{\delta} \equiv at\delta \wedge a > 0$$

$S = \{\eta : \{a, b\} \rightarrow \mathbb{N}_0, V_{\Pi} \rightarrow \{\mathbf{ff}, \mathbf{tt}\} \mid \eta(exec \lambda) = \mathbf{tt} \text{ for at most one } \lambda \in Act_{\Pi},$
 $\text{not } \eta(at \alpha) = \eta(at \beta) = \mathbf{tt}, \text{ not } \eta(at \gamma) = \eta(at \delta) = \mathbf{tt},$
 $\eta \text{ satisfies constraints for fSTS}\}$

$T = tot(T')$ with:

$(\eta, \eta') \in T' \Leftrightarrow \eta(exec \alpha) = \mathbf{tt}, \eta'(a) = \eta(a), \eta'(b) = \eta(b) * 2,$

$\eta'(at \alpha) = \mathbf{ff}, \eta'(at \beta) = \mathbf{tt}, \eta'(at \gamma) = \eta(at \gamma), \eta'(at \delta) = \eta(at \delta)$

or $\eta(exec \beta) = \mathbf{tt}, \eta'(a) = \eta(a) + 1, \eta'(b) = \eta(b)$

$\eta'(at \beta) = \mathbf{ff}, \eta'(at \lambda) = \eta(at \lambda) \text{ for } \lambda \not\equiv \beta$

or $\eta(exec \gamma) = \mathbf{tt}, \eta'(a) = \eta(a), \eta'(b) = \eta(b) + 1$

$\eta'(at \gamma) = \mathbf{ff}, \eta'(at \delta) = \mathbf{tt}, \eta'(at \alpha) = \eta(at \alpha), \eta'(at \beta) = \eta(at \beta)$

or $\eta(exec \delta) = \mathbf{tt}, \eta'(a) = \eta(a) - 1, \eta'(b) = \eta(b)$

$\eta'(at \delta) = \mathbf{ff}, \eta'(at \lambda) = \eta(at \lambda) \text{ for } \lambda \not\equiv \delta$

Verification of Concurrent Programs: A Simple Programming Language

PAR

Definition(The Language PAR):

For a program Π let be given:

- a signature SIG_{Π} , a structure S_{Π} for SIG_{Π} , $\mathcal{L}_{\Pi} = \mathcal{L}_{FOL}(SIG_{\Pi})$,
- a set \mathcal{E} of **elementary statements**, which change the “program variables” (these are distinguished constants of \mathcal{L}_{Π}), e.g. assignments,
- a set \mathcal{M} of **labels**.

Program structure: **initial** R ;
 cobegin $\Pi_1 || \dots || \Pi_p$ **coend** ($p \geq 1$),

where: R : formula of \mathcal{L}_{Π} (**initialisation condition**),
 Π_i : **parallel component** (for $1 \leq i \leq p$)

with the following BNF-syntax:

$\langle \text{parallel component} \rangle ::= \mathbf{loop} \langle \text{statement list} \rangle \mathbf{end}$

$\langle \text{statement list} \rangle ::= \langle \text{statement} \rangle \{ ; \langle \text{statement} \rangle \}^*$

$\langle \text{statement} \rangle ::= \langle \text{label} \rangle : \langle \text{unlabeled statement} \rangle$

$\langle \text{unlabeled statement} \rangle ::= \langle \text{Element of } \mathcal{E} \rangle \mid$

$\mathbf{await} \langle \text{condition} \rangle \{ \mathbf{then} \langle \text{element of } \mathcal{E} \rangle \}_0^1 \mid$

$\mathbf{if} \langle \text{condition} \rangle \mathbf{then} \langle \text{statement list} \rangle$

$\{ \mathbf{else} \langle \text{statement list} \rangle \}_0^1 \mathbf{endif}$

$\langle \text{condition} \rangle ::= \langle \text{formula of } \mathcal{L}_{\Pi} \rangle$

$\langle \text{label} \rangle ::= \langle \text{element of } \mathcal{M} \rangle$

Labelling condition: All labels occurring in a program are pairwise distinct.