Software Modeling

- Motivation
- Different Software Systems
- States & State Spaces
- Transition Systems
- Granularity problems
- Examples

Motivation for Formal Software Models

- Reliability
- Software Size
- Cost of Testing & Bugs

Software systems

- Sequential Systems
- Concurrent Systems
- Reactive Systems

State Spaces

- States
- *state space* of a system is a graph $< S, \Delta, I >$
- S is set of Spaces. $\Delta \subseteq S \times S$ is a transition function from states to states and $I \subseteq S$ are the initial states. This system is called an automaton.
- A *run* of a system is sequence $s_0 s_1 s_2 ...$ $s_0 \in I$. Run must be maximal, ie. it is either infinite or the system reaches a state with no successor.
- *Interleaving Model*

Transition Systems

- Transition system $< S, T, \Theta >$
- First order *structure* S includes *signature* G, domain(s), relations, functions and interpretation function. Signature G includes set of variables V (Program variables + program counters)
- Transition system implicitly defines state by set of variables V.
- Finite set T *transitions* $t \in T$ is of the form

$$p \longrightarrow (v_1, v_2, ..., v_n) := (e_1, e_2, ..., e_n)$$

- Initial Condition Θ, first order formula over V

Transition systems

- Transition $t \in T : p \longrightarrow (v_1, v_2, ..., v_n) := (e_1, e_2, ..., e_n)$ can execute from any state that satisfies the enabledness condition p.
- If t satisfies p at state s then $s \models^S p$.
- New state obtained by transition can be denoted $s' = t(s)$.
- All values of transition are calculated before installing them to the variables $v \in V$.
- Execution is an infinite sequence of states $s_0, s_1, s_2, ...,$ and $s_0 \models^S \Theta$ and for all $i \geq N$ holds either.
  1. $\exists t \in T : p \longrightarrow (v_1, v_2, ..., v_n) := (e_1, e_2, ..., e_n), s.e.s_i \models^S p.$
  2. There is no transition enabled for state s_i. Then $\forall j,\ j > i$ $s_j = s_i$

Scheduler

Start from some initial state $\imath$ such that $\imath \models^S \Theta$. Set $s := \imath$

loop: If there is no enabled transition from s, goto *extend*.

Pick up a transition t that is currently enabled at s.

Apply the transition t to the current state s,

obtaining a new current state $s := t(s)$.

goto *loop*.

extend: Repeat state s forever.

Integer Division (Example of Transition System)

$m1 : y1 := 0;$

$m2 : y2 := x1;$

$m3 : \mathit{while}\ y2 >= x2\ \mathit{do}$

$m4 : y1 := y1 + 1;$

$m5 : y2 := y2 - x2;$

$m6 : \mathit{end};$

Integer Division (Continued)

- $V = \{y1, y2, x1, x2, pc\}$
- Function symbols $'+'$ and $'-'$
- Domains: $Naturals$ and labels $L = \{m1, m2, m3, m4, m5, m6\}$
- Initial states are those that satisfy $\Theta : x2 > 0 \wedge pc \equiv m1$.

$pc = m1 \longrightarrow (pc, y1) := (m2, 0)$
$pc = m2 \longrightarrow (pc, y2) := (m3, x1)$
$pc = m3 \wedge y2 \geq x2 \longrightarrow pc := m4$
$pc = m3 \wedge y2 < x2 \longrightarrow pc := m6$
$pc = m4 \longrightarrow (pc, y1) := (m5, y1 + 1)$
$pc = m5 \longrightarrow (pc, y2) := (m3, y2 - 2)$

Granularity of Transitions

- How to choose correct amount of granularity?
- Granularity too small $\longrightarrow$ too many states
- Granularity too large $\longrightarrow$ information lost
- Simple example, Processes P_1 and P_2, $x = 2$ and $y = 3$
- $P_1 : x := x + y$ $P_2 : y := y + x$

Calculating Combinations

- The program consists of two parallel processes

$$
\begin{array}{l}
l_1 : if\ y1 = (n-k)\ then\ halt \\
l_2 : y3 := y3 \times y1 \\
l_3 : y1 := y1 - 1 \\
l_4 : goto\ l_1
\end{array}
\quad \Big\| \quad
\begin{array}{l}
r_1 : if\ y2 = k\ then\ halt \\
r_2 : y2 := y2 + 1 \\
r_3 : await\ y2 \leq n - y1 \\
r_4 : y3 := y3/y2 \\
r_5 : goto\ r_1
\end{array}
$$

Calculating Combinations (Continued)

- The initial condition Θ is
  $\Theta : y1 \equiv n \wedge y2 \equiv 0 \wedge y3 \equiv 1 \wedge pc_l \equiv l_1 \wedge pc_r \equiv r_1 \wedge n > 0 \wedge k > 0$
- The await statement in r_3 means P_r waits until P_l has done more executions than P_r.
- Function symbols: $'\times'$, $'+'$, $'-'$, '/' and $' \leq'$
- Variables V: $\{l_1, ..., l_4, r_1, ..., r_5, halt_l, halt_r, pc_l, pc_r, y1, y2, y3\}$

Calculating Combinations

$t_1 : pc_l \equiv l_1 \longrightarrow pc_l := if(y1 \equiv (n-k), halt_l, l_2)$

$t_2 : pc_l \equiv l_2 \longrightarrow (pc_l, y3) := (l_3, y3 \times y1)$

$t_3 : pc_l \equiv l_3 \longrightarrow (pc_l, y1) := (l_4, y1 - 1)$

$t_4 : pc_l \equiv l_4 \longrightarrow pc_l := l_1$

$t_5 : pc_r \equiv r_1 \longrightarrow pc_r := if(y2 \equiv k, halt_r, r_2)$

$t_6 : pc_r \equiv r_2 \longrightarrow (pc_r, y2) := (r_3, y2 + 1)$

$t_7 : pc_r \equiv r_3 \wedge y2 \leq n - y1 \longrightarrow pc_r := r_4$

$t_8 : pc_r \equiv r_4 \longrightarrow (pc_r, y3) := (r_5, y3/y2)$

$t_9 : pc_r \equiv r_5 \longrightarrow pc_r := r_1$

Mutual Exclusion

- Processes compete over Critical Section
- Goals: Exclusiveness & Liveness
- Attempt to provide such a protocol:
  boolean c,1 c2 initially 1;
- The initial condition Θ is
  $\Theta : pc_1 \equiv m1 \wedge pc_2 \equiv n1 \wedge c1 \equiv 1 \wedge c2 \equiv 1$

$P1 :: m1 : while true do$	$\|\|$	$P2 :: n1 : while true do$
$m2 : (*noncritical section 1*)$		$n2 : (*noncritical section 2*)$
$m3 : c1 := 0;$		$n3 : c2 := 0;$
$m4 : wait until c2 = 1;$		$n4 : wait until c1 = 1;$
$m5(*critical section 1*)$		$n5 : (*critical section 2*)$
$m6 : c1 := 1$		$n6 : c2 := 1$
end		end

Mutex (Continued)

$\tau_1 : pc_1 \equiv m1 \longrightarrow pc_1 := m2$

$\tau_2 : pc_1 \equiv m2 \longrightarrow pc_1 := m3$

$\tau_3 : pc_1 \equiv m3 \longrightarrow (pc_1, c1) := (m4, 0)$

$\tau_4 : pc_1 \equiv m4 \wedge c2 \equiv 1 \longrightarrow pc_1 := m5$

$\tau_5 : pc_1 \equiv m5 \longrightarrow pc_1 := m6$

$\tau_6 : pc_1 \equiv m6 \longrightarrow (pc_1, c1) := (m1, 1)$

$\tau_7 : pc_2 \equiv n1 \longrightarrow pc_2 := n2$

$\tau_8 : pc_2 \equiv n2 \longrightarrow pc_2 := n3$

$\tau_9 : pc_2 \equiv n3 \longrightarrow (pc_2, c2) := (n4, 0)$

$\tau_{10} : pc_2 \equiv n4 \wedge c1 \equiv 1 \longrightarrow pc_2 := n5$

$\tau_{11} : pc_2 \equiv n5 \longrightarrow pc_2 := n6$

$\tau_{12} : pc_2 \equiv n6 \longrightarrow (pc_2, c2) := (n1, 1)$