

## Untimed Systems: Modeling

## Modeling: informal motivation

Model is a simplification of a system for understanding and analysing particular aspects

In e.g., control engineering, models describe relation between (real-valued) quantities

In protocol and software verification, models should describe possible combinations of sequences of computation steps

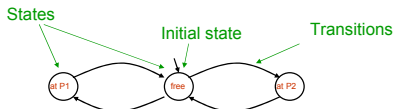
=>

Models will have discrete states and transitions between states.

## Transition Systems

Model to describe the operational behavior of systems

Example: Resource Allocation System:



Variables: **status** (what can be "observed")

Computation:

## Transition Systems (formal definition)

A Transition System is a tuple  $(S, S_0, \rightarrow, V, L)$  where

$S$  is a set of states

$S_0$  is a set of initial states

$\rightarrow$  (a subset of  $S \times S$ ) is a transition relation

$V$  is a set of variables

$L : S \rightarrow (V \rightarrow \text{Val})$  gives a valuation of variables in each state

Write  $s \rightarrow s'$  for  $(s, s') \in \rightarrow$

## Transition Systems (formal definition)

A Transition System is a tuple  $(S, S_0, \rightarrow, V, L)$  where

$S$  is a set of states

$S_0$  is a set of initial states

$\rightarrow$  (a subset of  $S \times S$ ) is a transition relation

$V$  is a set of variables

$L : S \rightarrow (V \rightarrow \text{Val})$  gives a valuation of variables in each state

Write  $s \rightarrow s'$  for  $(s, s') \in \rightarrow$

A Computation is a finite or infinite sequence of states

$s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9$

such that

- $s_0$  is an initial state
- $(s_i, s_{i+1}) \in \rightarrow$  for all relevant  $i$

## Structured Descriptions of Transition Systems

Drawing large graphs is very inconvenient.

We need constructs to define transition systems conveniently

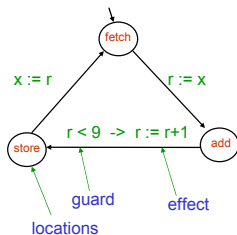
- variables
- processes
- communication channels

## Adding Variables (Action Systems)

Example: simple incrementation of stored variable

## Action System

Variables:  $x, r$ : integer  
Initially  $x = 0$



## Transition system

States:  
locations  $\times$  evaluations of variables  
(add, {x  $\rightarrow$  0, r  $\rightarrow$  0} )

Transitions:

```
(add, {x -> 0, r -> 0})  
->  
(store, {x -> 1, r -> 0})
```

Variables:  
loc x, r

Valuation:

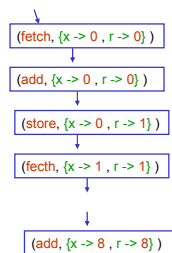
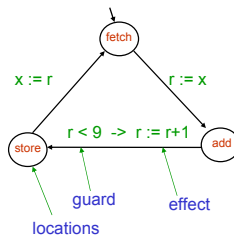
$$(\text{add}, \{x \rightarrow 0, r \rightarrow 0\})(x) = 0$$
$$(\text{add}, \{x \rightarrow 0, r \rightarrow 0\}) (\text{loc}) = \text{add}$$

7

## Corresponding Transition System

## Program graph

Variables: **x**, **r**: integer  
Initially **x = 0**

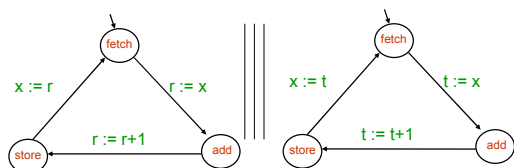


8

## Parallel Processes

Program graphs can be put in parallel

Variables:  $x, r, t$ : integer  
Initially  $x = 0$



States: (add, fetch, {x → 0, r → 0, t → 0})

Transitions: (add, fetch, {x → 0, r → 0, t → 0}) → (store, fetch, {x → 0, r → 1, t → 0})

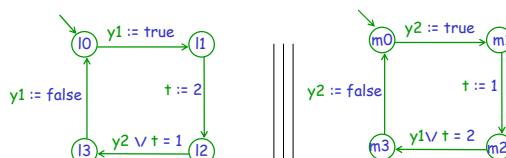
Variables: `loc[1]`, `loc[2]`, `x`, `r`, `t`

9

## Peterson-Fischer Mutual Exclusion prot.

Variables:  $y_1, y_2$ : boolean,  $t$ :  $\{1,2\}$

Initially  $y1 = y2 = \text{false}$ ,  $t = 1$



States: (l1, m1, {y1 → true, y2 → true, t → 1})

Initial States: ( l0, m0, {y1 -> false, y2 -> false, t -> 1} )

Transition:

$(l_0, m_0, \{y_1 \rightarrow \text{false}, y_2 \rightarrow \text{false}, t \rightarrow 1\}) \rightarrow (l_1, m_0, \{y_1 \rightarrow \text{true}, y_2 \rightarrow \text{false}, t \rightarrow 1\})$

10

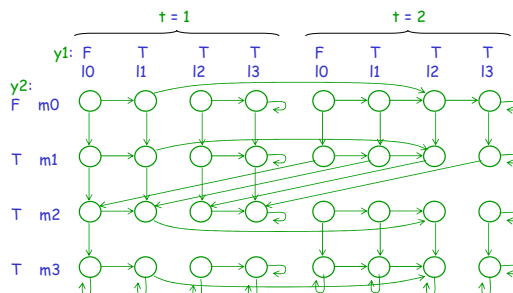
## Interleaving

Transitions of parallel components are merged or “interleaved”

- as if the model is “executed” on a single processor
- The order between transitions of different components is completely arbitrary
- as if there is a scheduler with an unknown strategy
  - possible that some components are scheduled “very seldom” or never at all
- Particular scheduling strategies can be enforced by adding a “scheduler” into the model
- e.g., by using shared variable(s) to model the state of a scheduler
- Certain minimal assumptions on scheduling can be represented using *fairness* (more about this later)

11

## Peterson-Fischer: Transition System

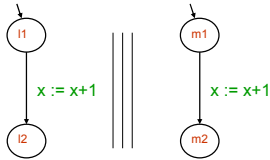


12

## Atomicity

Each transition is atomic, i.e., it is an undivisible operation, which has no "intermediate states", i.e., can not be stopped "in the middle"

Variables:  $x$ : integer  
Initially  $x = 0$

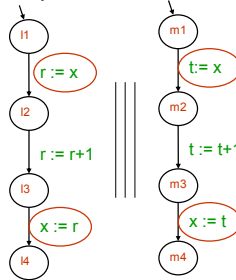


This transition system will "terminate" with the value of  $x$  being 2

13

## How much detail must be modeled?

Variables:  $x, r, t$ : integer  
Initially  $x = 0$



All possible sequences of "externally interesting" transitions must be modeled

**Critical event** is an operation, for which is makes a difference how it is interleaved

- Reading shared variable
- Writing to shared variable

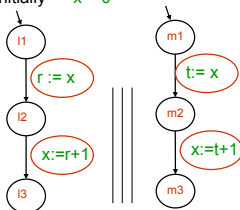
Recepy:

- Each transition should involve at most one critical event

14

## Also a faithful model

Variables:  $x, r, t$ : integer  
Initially  $x = 0$



All possible sequences of "externally interesting" transitions must be modeled

**Critical event** is an operation, for which is makes a difference how it is interleaved

- Reading shared variable
- Writing to shared variable

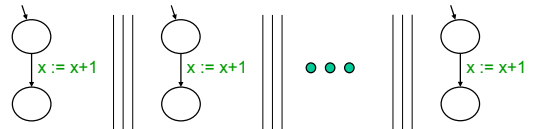
Recepy:

- Each transition should involve at most one critical event

15

## Arguments for using Interleaving

- "Principle of relativity" (simultaneity can not be observed)
- Fewer transitions to consider in analysis



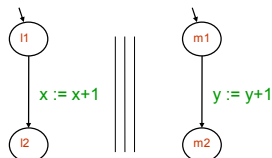
- In general:  $2^n$  possible transitions
- With interleaving:  $n$  possible transitions

16

## Limitations of interleaving

- Some (noninteresting) "pathological" properties are satisfied when using interleaving

Variables:  $x, y$ : integer  
Initially  $x = y = 0$



- Interleaving makes the system satisfy the (uninteresting) property: "there must be a (short) instant when  $x$  and  $y$  are different"

17

## Small Idealized Example from Reality

Small Example: Mars Pathfinder 1997

Typical properties of synchronization in real-time systems

- Mutual exlusion
  - A process cannot access the data-bus unless it owns a mutex-lock
- Scheduling priority
  - Saving data to memory has higher priority than processing data
  - Low priority process cannot execute when high priority process is ready to execute

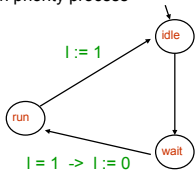
18

## Idealized model of processes

Variables:  $l$ : integer

Initially  $l = 1$

High priority process



Low priority process

