並行分散計算特論 (2)

Shoji Yuen

2011/10/11

Alternative Semantics

Communicating process is more distinguishable than classic automata
(Automata theory = Language)

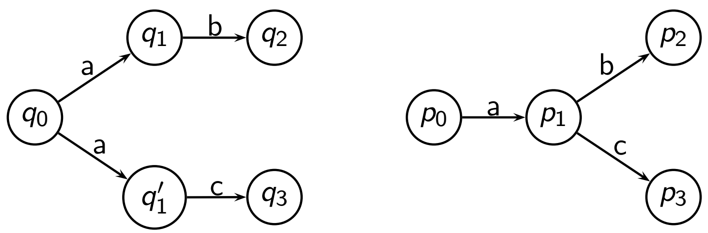
Communication enables to observe intermediate states

What could be the alternative semantics for languages?

Example3.2

Essential diffrence between B_1 and B_2

Nondeterminism after $2p$



a = insert a coin b = get coffee c = get tea

Labelled Transition System

- LTS = Automaton - FinalStates - InitialState

Definition

LTS

An LTS over Act: $(\mathcal{Q}, \mathcal{T})$
$\mathcal{Q}$: *Set of states*
$\mathcal{T}$: *Transitions*

$$q \xrightarrow{a} q' \quad \text{for } (q, a, q') \in \mathcal{T}$$

LTS as Behavioral Model

Principles:

- Communication is the only way to know what a communication process is;
- There is no way to know if a communicating proces is in the initila state nor in the final states.
- Reaction pattern of communications = semantics of communicating processes

Realized as an (equivalence) relation over LTS states

Strong simulation

A class of relations over LTS states

Definition

$\mathcal{S}$ is a simulation:
For all $(p, q) \in \mathcal{S}$ and $a \in Act$, $p \xrightarrow{a} p'$ implies that there exists q' such that $q \xrightarrow{a} q'$ and $(p', q') \in \mathcal{S}$.

If there is a simulation that includes (p, q), it is said that q strongly simulates p or p is strongly simulated by q

$$\begin{array}{cccc} p & \xrightarrow{a} & \forall & p' \\ \mathcal{S} & & & \mathcal{S} \\ q & \xrightarrow{a} & \exists & q' \end{array}$$

Intuitive meanings of simulation

q (strongly) simulates p

q is more capable in the communications than q at each point of communication from now on

p can be more nondeterministic than q

Example 3.4
$\{(q_0, p_0), (q_1, p_1), (q'_1, p_1), (q_2, p_2), (q_3, p_3)\}$

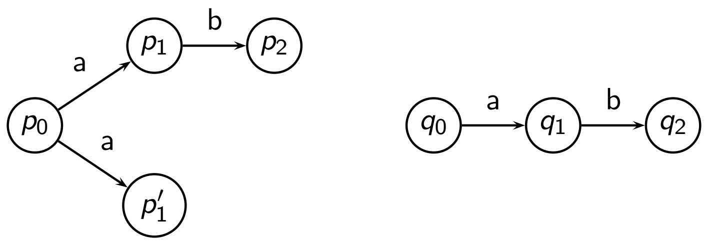
$$p_0 \text{ (strongly) simulates } q_0$$

Strong bisimulation

Definition

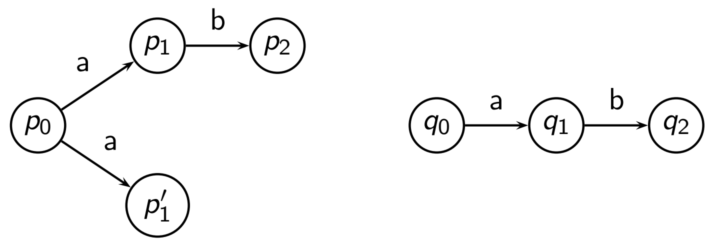
Given an LTS $(\mathcal{Q}, \mathcal{T})$, a simulation $\mathcal{S}$ over $\mathcal{Q}$:
$\mathcal{S}$ is a storng bisimulation *if $\mathcal{S}$ and $\mathcal{S}^{-1}$ are both strong simulations.*
$p \sim q$ if there exists a strong bisimulation $\mathcal{S}$ such that $(p, q) \in \mathcal{S}$

Bismularity



p_0 simulates q_0 and q_0 simulates p_0
But p is not bisimular to q

Bismularity



p_0 simulates q_0 and q_0 simulates p_0
But p is not bisimular to q
Two simulations are not in the reversal relation.

Properties of $\sim$

- $\sim$ is an equivalence (Prop.3.9)
- $\sim$ is a strong bisimulation (Prop.3.9)

Moreover, $\sim$ is the largest strong bisimulation.

Sequential process expression

$$P ::= A\langle a_1, \ldots, a_n \rangle \mid \Sigma_{i \in I} \alpha_i . P_i$$

I:finite index set
for each A: $A(\vec{a}) \stackrel{def}{=} P_A$

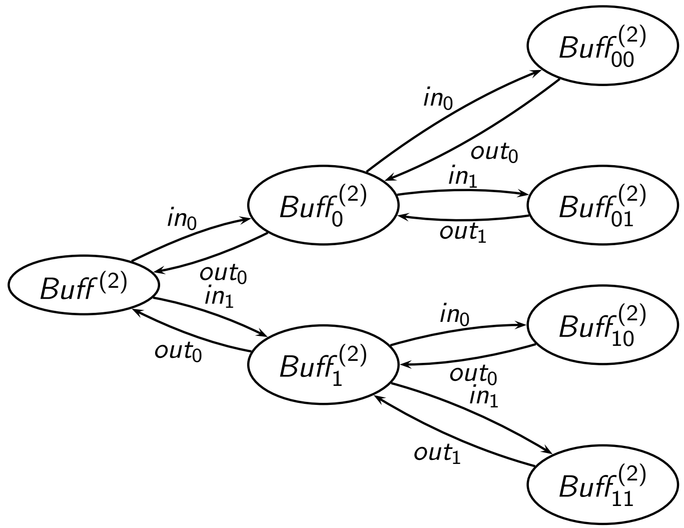
Process identifier A is for recusion

Structural congruence

- Alpha conversion for bound names
- A-C property of operators (Choice)

$\equiv$ is the trivial equivalence
We consider the quotiant of structural congruences over LTS

Boolean Buffer



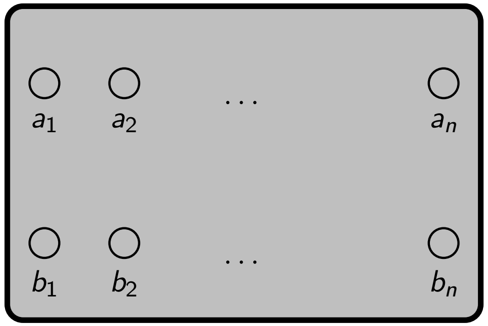
Expression of Boolean Buffer

$$Buff^{(2)} \overset{def}{=} \sum_{i\in\{0,1\}} in_i.Buff^{(2)}_i$$

$$Buff^{(2)}_i \overset{def}{=} \overline{out_i}.Buff^{(2)} + \sum_{j\in\{0,1\}} in_j.Buff^{(2)}_{ji}$$

$$Buff^{(2)}_{ij} \overset{def}{=} \overline{out_j}.Buff^{(2)}_i$$

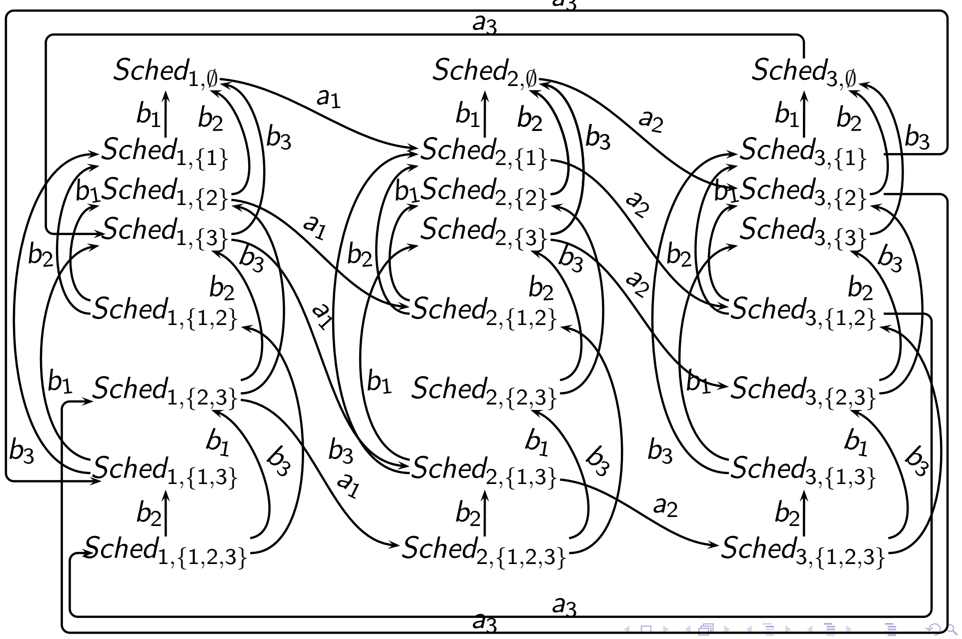
Scheduler specification



a_i: Job_i starts
b_j: job_j ends

Each job can be invoked only once at a time.
a_i is unavailable without pushing b_i.
A job can be overtaken by other jobs.
You can push b_j before b_i $(i < j)$ is pusehd.

Scheduler (n=3)



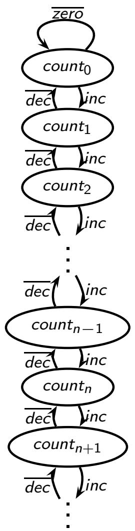
Expression of scheduler

$$Scheduler \stackrel{def}{=} Sched_{1,\emptyset}$$

$$Sched_{i,X} \stackrel{def}{=} \begin{cases} \sum_{j\in X} b_j.Sched_{i,X\setminus j} & i \in X \\ \sum_{j\in X} b_j.Sched_{i,X\setminus j} + a_i.Sched_{i+1,X\cup i} & i \notin X \end{cases}$$

$Sched_{i,X}$:a_i is in turn. X is the set of jobs whose b action is not yet done.

Counter



$$Count_0 \stackrel{def}{=} inc.Count_1 + \overline{zero}.Count_0$$

$$Count_{n+1} \stackrel{def}{=} inc.Count_{n+2} + \overline{dec}.Count_n$$

Counter is not in the regular class.
(More expressive than the sequential processes)
Infinite number of process identifiers.