

Systematic Analysis of CSP Models (1)

October 3, 2008

Outline

- Operational semantics
 - Given a process specifying some system, how does it execute?
- Mechanical system analysis
 - Given a process specifying some system, how do we know whether it is correct or not?

How a Given Process Executes?

- Denotational Semantics
 - $traces(P \parallel Q) = \{t : \text{seq } A \mid (t \upharpoonright \alpha P \in traces(P) \wedge (t \upharpoonright \alpha Q \in traces(Q)))\}$
where $A = \alpha P \cup \alpha Q$.
- Operational Semantics
 - Given a system state, what are the possible actions the system can perform and what are the outcomes?
 - $P \xrightarrow{a} Q$

Operational Semantics

- Operational Semantics can be presented using a set of inference rules of the following form,

$$\frac{\textit{Premises}}{\textit{Conclusion}}$$

- e.g.,

$$\frac{P \xrightarrow{a} P'}{P \square Q \xrightarrow{a} P'}$$

Operational Semantics: Primitives

- *STOP*,

- *SKIP*,

$$\frac{}{SKIP \xrightarrow{\checkmark} STOP} [skip]$$

- Prefixing,

$$\frac{}{(a \rightarrow P) \xrightarrow{a} P} [prefixing]$$

Operational Semantics: Choices

- External choice^a,

$$\frac{P \xrightarrow{a} P'}{(P \sqcap Q) \xrightarrow{a} P'} \quad [\textit{extchoice1}] \qquad \frac{Q \xrightarrow{a} Q'}{(P \sqcap Q) \xrightarrow{a} Q'} \quad [\textit{extchoice2}]$$

- Internal choice, let τ be the silent invisible event,

$$\frac{}{(P \sqcap Q) \xrightarrow{\tau} P} \quad [\textit{intchoice1}] \qquad \frac{}{(P \sqcap Q) \xrightarrow{\tau} Q} \quad [\textit{intchoice2}]$$

^awhere a is a visible event, some other rules are omitted.

Operational Semantics: Sequential Composition

In process $P; Q$, P takes control first and Q starts only when P has finished. Let \checkmark be a distinguished event denoting termination.

$$\frac{P \xrightarrow{a} P'}{(P; Q) \xrightarrow{a} (P'; Q)} \text{ [seq1]}$$

$$\frac{P \xrightarrow{\checkmark} P'}{(P; Q) \xrightarrow{\tau} Q} \text{ [seq2]}$$

Operational Semantics: Interrupt

In process $P \nabla Q$, whenever an event is engaged by Q , P is interrupted and the control transfer to Q .

$$\frac{P \xrightarrow{a} P'}{(P \nabla Q) \xrightarrow{a} (P' \nabla Q)} [interrupt1] \qquad \frac{Q \xrightarrow{a} Q'}{(P \nabla Q) \xrightarrow{a} Q'} [interrupt1]$$

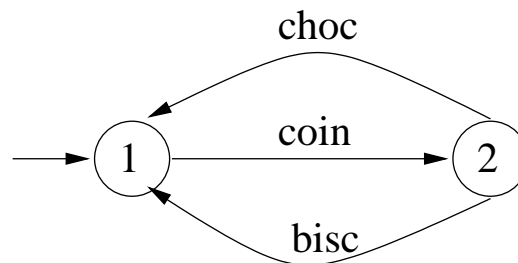
Operational Semantics: Example

Let $VMS = coin \rightarrow (choc \rightarrow VMS \sqcap bisc \rightarrow VMS)$.

- $step1.$ $VMS \xrightarrow{coin} (choc \rightarrow VMS \sqcap bisc \rightarrow VMS)$ – by rule *prefixing*
- $step2.$ $(choc \rightarrow VMS \sqcap bisc \rightarrow VMS) \xrightarrow{choc} VMS$ – by rule *extchoice1*
- $step2.$ $(choc \rightarrow VMS \sqcap bisc \rightarrow VMS) \xrightarrow{bisc} VMS$ – by rule *extchoice1*

Labeled Transition System

A Labeled Transition System contains a set of states, an initial state (where the system starts from) and a (labeled) transition relation.



where $VMS = coin \rightarrow (choc \rightarrow VMS \square bisc \rightarrow VMS)$, state 1 represents the process VMS and state 2 represents the process $(choc \rightarrow VMS \square bisc \rightarrow VMS)$.

Operational Semantics: Interleaving

In process $P \parallel Q$, P and Q behaves independently^a.

$$\frac{P \xrightarrow{a} P'}{(P \parallel Q) \xrightarrow{a} (P' \parallel Q)} \text{ [} \textit{interleave1} \text{]}$$

$$\frac{Q \xrightarrow{a} Q'}{(P \parallel Q) \xrightarrow{a} (P \parallel Q')} \text{ [} \textit{interleave2} \text{]}$$

^aexcept termination. Assume that a is not \checkmark .

Operational Semantics: Synchronization

In process $P \parallel [X] \parallel Q$, no event from X may occur unless jointly by both P and Q . When events from X do occur, they occur in both P and Q simultaneously.

$$\frac{P \xrightarrow{a} P' \text{ and } a \notin X}{(P \parallel [X] \parallel Q) \xrightarrow{a} (P' \parallel [X] \parallel Q)} \text{ [syn1]}$$

$$\frac{Q \xrightarrow{a} Q' \text{ and } a \notin X}{(P \parallel [X] \parallel Q) \xrightarrow{a} (P \parallel [X] \parallel Q')} \text{ [syn2]}$$

$$\frac{P \xrightarrow{a} P' \text{ and } Q \xrightarrow{a} Q' \text{ and } a \in X}{(P \parallel [X] \parallel Q) \xrightarrow{a} (P' \parallel [X] \parallel Q')} \text{ [syn3]}$$

Operational Semantics: Example (cont'ed)

Given the process $a \rightarrow P \parallel [a] (c \rightarrow a \rightarrow Q)$.

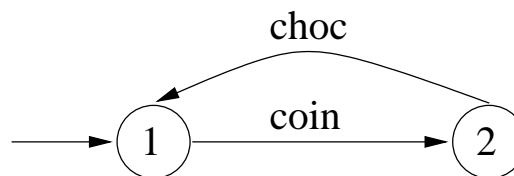
$$\begin{array}{ll} \textit{step1} : & (a \rightarrow P \parallel [a] (c \rightarrow a \rightarrow Q)) \xrightarrow{c} (a \rightarrow P \parallel [a] (a \rightarrow Q)) & \text{-- rule } \textit{syn2} \\ \textit{step2} : & (a \rightarrow P \parallel [a] (a \rightarrow Q)) \xrightarrow{a} (P \parallel [a] Q) & \text{-- rule } \textit{syn3} \end{array}$$

Operational Semantics: Example (cont'ed)

- $VMC = coin \rightarrow (choc \rightarrow VMC \square bisc \rightarrow VMC)$
- $CHOCLOV = choc \rightarrow CHOCLOV \square coin \rightarrow choc \rightarrow CHOCLOV$
- How process $VMC \parallel [A] \parallel CHOCLOV$ where $A = \{coin, choc, bisc\}$ behaves?

step1 : $VMC \parallel [A] \parallel CHOCLOV \xrightarrow{coin} ?$

step2 : $(choc \rightarrow VMC \square bisc \rightarrow VMC) \parallel [A] \parallel (choc \rightarrow CHOCLOV) \xrightarrow{choc} ?$



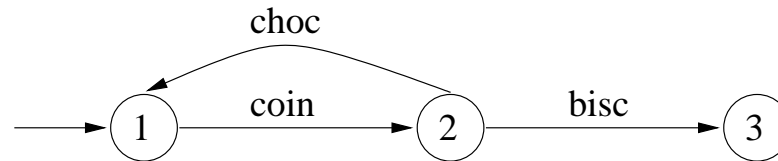
Operational Semantics: Example (cont'ed)

- $VMC = coin \rightarrow (choc \rightarrow VMC \square bisc \rightarrow VMC)$
- $CHOCLOV = choc \rightarrow CHOCLOV \square coin \rightarrow choc \rightarrow CHOCLOV$
- $VMC \parallel [coin, choc] \parallel CHOCLOV$ or equivalently $VMC \parallel CHOCLOV$ behaves as follows,

step1 : $VMC \parallel CHOCLOV \xrightarrow{coin} ???$

step2 : $(choc \rightarrow VMC \square bisc \rightarrow VMC) \parallel (choc \rightarrow CHOCLOV) \xrightarrow{choc} ???$

step2 : $(choc \rightarrow VMC \square bisc \rightarrow VMC) \parallel (choc \rightarrow CHOCLOV) \xrightarrow{bisc} ???$



Operational Semantics: Example (cont'ed)

Step 1: specify the dining philosophers,

$$\begin{aligned} Alice &= Alice.get.fork1 \rightarrow Alice.get.fork2 \rightarrow Alice.eat \\ &\rightarrow Alice.put.fork1 \rightarrow Alice.put.fork2 \rightarrow Alice \end{aligned}$$

$$\begin{aligned} Bob &= Bob.get.fork2 \rightarrow Bob.get.fork1 \rightarrow Bob.eat \\ &\rightarrow Bob.put.fork2 \rightarrow Bob.put.fork1 \rightarrow Bob \end{aligned}$$

$$\begin{aligned} Fork1 &= Alice.get.fork1 \rightarrow Alice.put.fork1 \rightarrow Fork1 \square \\ &Bob.get.fork1 \rightarrow Bob.put.fork1 \rightarrow Fork1 \end{aligned}$$

$$\begin{aligned} Fork2 &= Alice.get.fork2 \rightarrow Alice.put.fork2 \rightarrow Fork2 \square \\ &Bob.get.fork2 \rightarrow Bob.put.fork2 \rightarrow Fork2 \end{aligned}$$

$$College = Alice \parallel Bob \parallel Fork1 \parallel Fork2$$

Operational Semantics: Example (cont'ed)

Step 2: get the alphabets of each process,

$$\alpha_{Alice} = \{Alice.get.fork1, Alice.get.fork2, Alice.eat, Alice.put.fork1, Alice.put.fork2\}$$

$$\alpha_{Bob} = \{Bob.get.fork1, Bob.get.fork2, Bob.eat, Bob.put.fork1, Bob.put.fork2\}$$

$$\alpha_{Fork1} = \{Alice.get.fork1, Alice.put.fork1, Bob.get.fork1, Bob.put.fork1\}$$

$$\alpha_{Fork2} = \{Alice.get.fork2, Alice.put.fork2, Bob.get.fork2, Bob.put.fork2\}$$

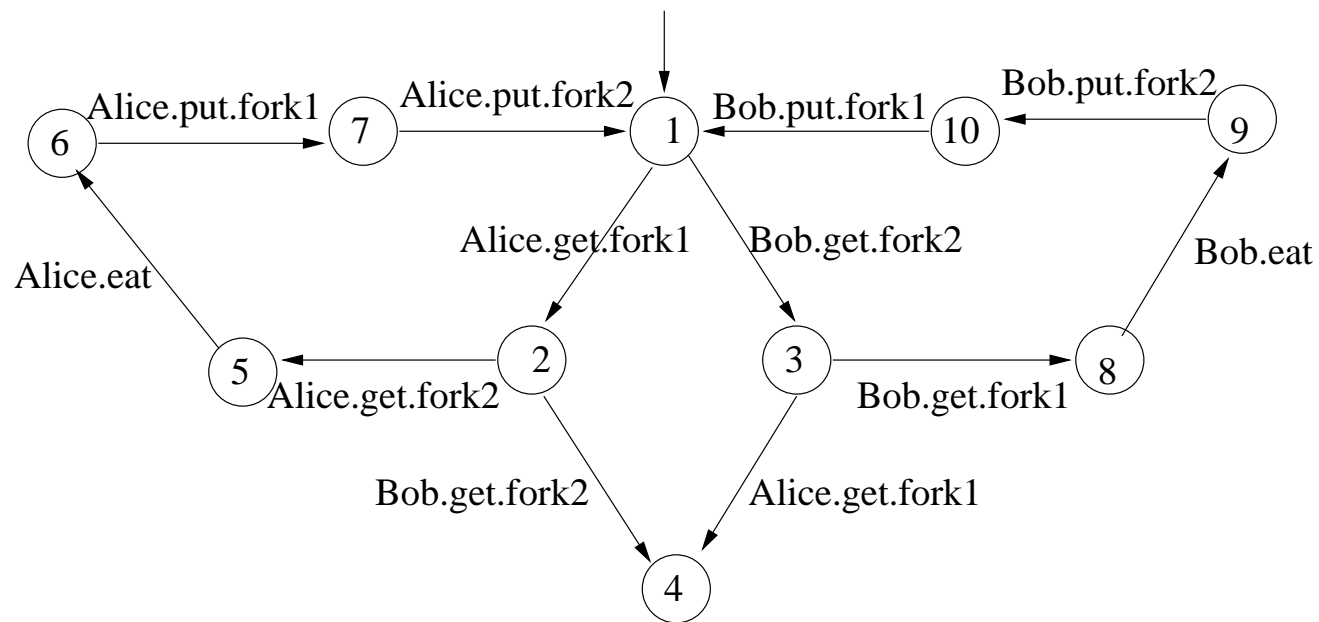
Operational Semantics: Example (cont'ed)

Step 3: apply the operational semantics rules (one at a time) to build the Labeled Transition System, e.g, initially,

- *Alice* can perform *Alice.get.fork1*;
- *Bob* can perform *Bob.get.fork2*;
- *Fork1* can perform *Alice.get.fork1* or *Bob.get.fork1*;
- *Fork2* can perform *Alice.get.fork2* or *Bob.get.fork2*;
- By rule *syn3*, *College* can perform either *Alice.get.fork1* or *Bob.get.fork2*, and then a state of the form.

... || ... || ... || ...

Operational Semantics: Example (cont'ed)



Operational Semantics: Example (cont'ed)

Step 4: analyze the Labeled Transition System,

- is the system deadlock-free?
- will Alice or Bob starve to death?
- ...

Tool Needed!

Process Analysis Toolkit

- PAT is a tool developed by our group for simulating and verifying systems.
- You specify the system, PAT simulates system behaviors.
- You specify the system, you ask the question, PAT answers (yes, or no with a counterexample).
- PAT is available at <http://www.comp.nus.edu.sg/~liuyang/pat>
- Next lecture: how to use PAT (or model checking in general) for system analysis (e.g., solving puzzles automatically).