How does an CSP Specification Execute?

February 28, 2009

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 Process Executes?

- Denotational Semantics
  - $traces(P \parallel Q) = \{t : \mathrm{seq}\, A \mid (t \upharpoonright \alpha P \in traces(P) \wedge (t \upharpoonright \alpha Q \in traces(Q))\}$ 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,

$$\frac{\mathit{Premises}}{\mathit{Conclusion}}$$

- e.g.,

$$\frac{P \ \xrightarrow{a} \ P'}{P \,\Box\, Q \ \xrightarrow{a} \ P'}$$

Operational Semantics: Primitives

- $\mathit{S\scriptstyle TOP}$,

- $\mathit{S\scriptstyle KIP}$,

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

- Prefixing,

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

Operational Semantics: Choices

- External choice[a],

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

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

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

[a] where 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.

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

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

Operational Semantics: Interrupt

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

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

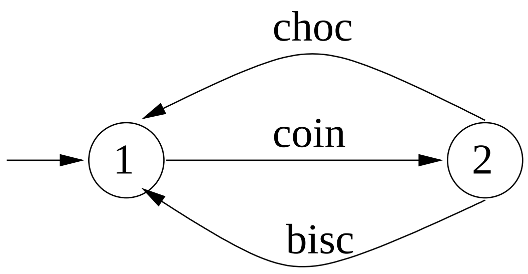
Example 1

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

$$
\begin{array}{lll}
step1. & VMS \stackrel{coin}{\rightarrow} (choc \rightarrow VMS \,\Box\, bisc \rightarrow VMS) & \text{– by rule } \textit{prefixing} \\
step2. & (choc \rightarrow VMS \,\Box\, bisc \rightarrow VMS) \stackrel{choc}{\rightarrow} VMS & \text{– by rule } \textit{extchoice}1 \\
step2. & (choc \rightarrow VMS \,\Box\, bisc \rightarrow VMS) \stackrel{bisc}{\rightarrow} VMS & \text{– by rule } \textit{extchoice}1
\end{array}
$$

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

Operational Semantics: Interleaving

In process $P \mid\mid\mid Q$, P and Q behaves independently[a].

$$\frac{P \xrightarrow{a} P'}{(P \mid\mid\mid Q) \xrightarrow{a} (P' \mid\mid\mid Q)} \; [\, \mathit{interleave}1 \,]$$

$$\frac{Q \xrightarrow{a} Q'}{(P \mid\mid\mid Q) \xrightarrow{a} (P \mid\mid\mid Q')} \; [\, \mathit{interleave}2 \,]$$

[a] except termination. Assume that a is not $\checkmark$.

Operational Semantics: Synchronization

In process $P \,|[\, X \,]|\, Q$, 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.

$$\frac{P \overset{a}{\rightarrow} P' \textit{ and } a \notin X}{(P \,|[\, X \,]|\, Q) \overset{a}{\rightarrow} (P' \,|[\, X \,]|\, Q)} \,[\, syn1 \,]$$

$$\frac{Q \overset{a}{\rightarrow} Q' \textit{ and } a \notin X}{(P \,|[\, X \,]|\, Q) \overset{a}{\rightarrow} (P \,|[\, X \,]|\, Q')} \,[\, syn2 \,]$$

$$\frac{P \overset{a}{\rightarrow} P' \textit{ and } Q \overset{a}{\rightarrow} Q' \textit{ and } a \in X}{(P \,|[\, X \,]|\, Q) \overset{a}{\rightarrow} (P' \,|[\, X \,]|\, Q')} \,[\, syn3 \,]$$

Operational Semantics: Example (cont'ed)

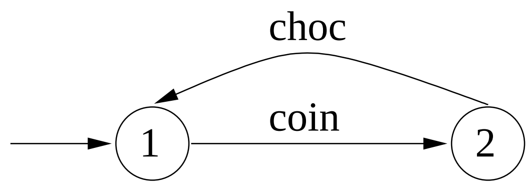
Given the process $a \rightarrow P \,|[\, a \,]|\, (c \rightarrow a \rightarrow Q)$.

$$\begin{array}{llr} step1: & (a \rightarrow P \,|[\, a \,]|\, (c \rightarrow a \rightarrow Q)) \xrightarrow{c} (a \rightarrow P \,|[\, a \,]|\, (a \rightarrow Q)) & \text{– rule } syn2 \\ step2: & (a \rightarrow P \,|[\, a \,]|\, (a \rightarrow Q)) \xrightarrow{a} (P \,|[\, a \,]|\, Q) & \text{– rule } syn3 \end{array}$$

Example 2

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

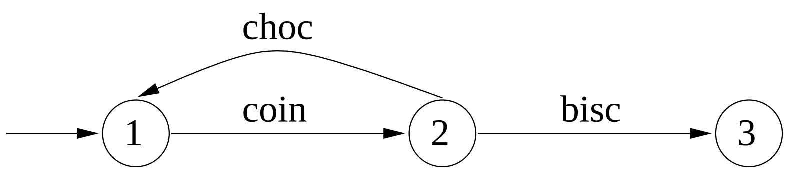
$$step1: \quad VMC \,|[\, A \,]|\, CHOCLOV \stackrel{coin}{\longrightarrow} \; ?$$
$$step2: \quad (choc \rightarrow VMC \,\Box\, bisc \rightarrow VMC) \,|[\, A \,]|\, (choc \rightarrow CHOCLOV) \stackrel{choc}{\longrightarrow} \; ?$$



Example 2 (cont'd)

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

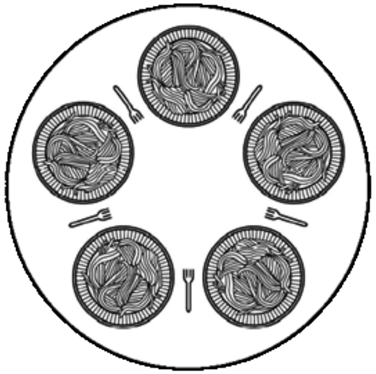
$$
\begin{array}{ll}
step1: & VMC \parallel CHOCLOV \stackrel{coin}{\rightarrow} ??? \\
step2: & (choc \rightarrow VMC \,\Box\, bisc \rightarrow VMC) \parallel (choc \rightarrow CHOCLOV) \stackrel{choc}{\rightarrow} ??? \\
step2: & (choc \rightarrow VMC \,\Box\, bisc \rightarrow VMC) \parallel (choc \rightarrow CHOCLOV) \stackrel{bisc}{\rightarrow} ???
\end{array}
$$



Case Study I: Dining Philosophers

Step 1: specify the dining philosophers,

$$
\begin{aligned}
Alice &= Alice.get.fork1 \rightarrow Alice.get.fork2 \rightarrow Alice.eat \\
&\quad \rightarrow Alice.put.fork1 \rightarrow Alice.put.fork2 \rightarrow Alice \\
Bob &= Bob.get.fork2 \rightarrow Bob.get.fork1 \rightarrow Bob.eat \\
&\quad \rightarrow Bob.put.fork2 \rightarrow Bob.put.fork1 \rightarrow Bob \\
Fork1 &= Alice.get.fork1 \rightarrow Alice.put.fork1 \rightarrow Fork1 \;\Box \\
&\quad Bob.get.fork1 \rightarrow Bob.put.fork1 \rightarrow Fork1 \\
Fork2 &= Alice.get.fork2 \rightarrow Alice.put.fork2 \rightarrow Fork2 \;\Box \\
&\quad Bob.get.fork2 \rightarrow Bob.put.fork2 \rightarrow Fork2 \\
College &= Alice \parallel Bob \parallel Fork1 \parallel Fork2
\end{aligned}
$$



Case Study I: Dining Philosophers (cont'ed)

Step 2: get the alphabets of each process,

$$
\begin{aligned}
\alpha Alice &= \{Alice.get.fork1, Alice.get.fork2, Alice.eat, Alice.put.fork1, Alice.put.fork2\} \\
\alpha Bob &= \{Bob.get.fork1, Bob.get.fork2, Bob.eat, Bob.put.fork1, Bob.put.fork2\} \\
\alpha Fork1 &= \{Alice.get.fork1, Alice.put.fork1, Bob.get.fork1, Bob.put.fork1\} \\
\alpha Fork2 &= \{Alice.get.fork2, Alice.put.fork2, Bob.get.fork2, Bob.put.fork2\}
\end{aligned}
$$

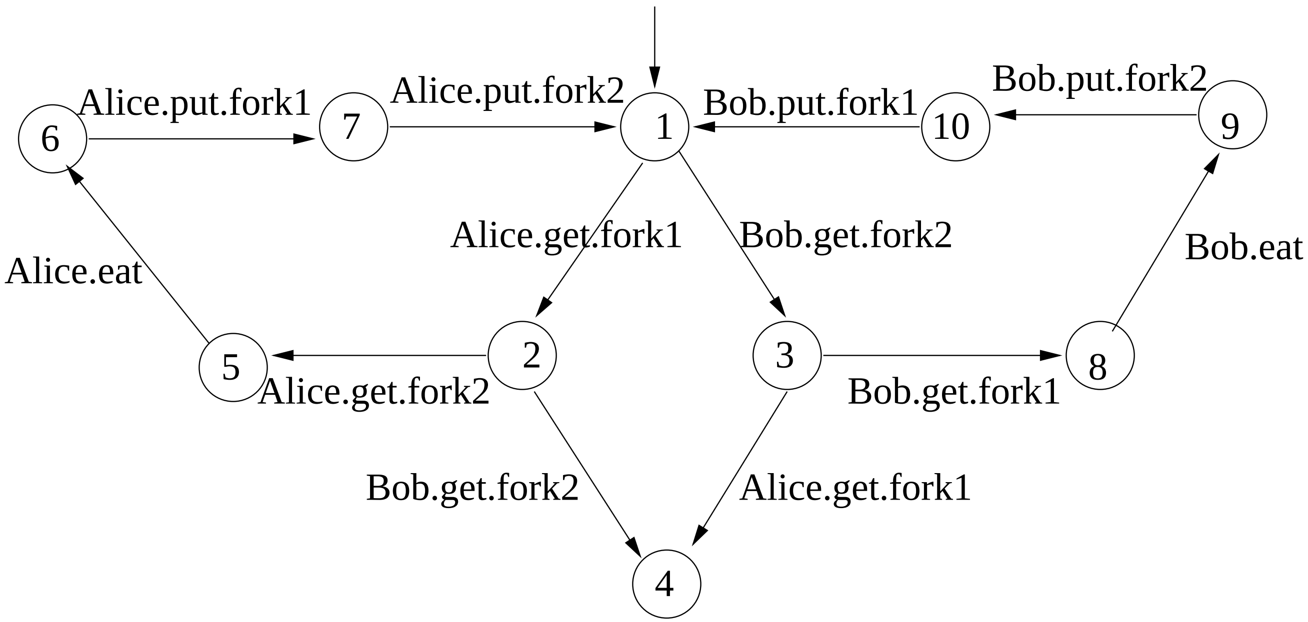
Case Study I: Dining Philosophers (cont'ed)

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.

$$\cdots \parallel \cdots \parallel \cdots \parallel \cdots$$

Case Study I: Dining Philosophers (cont'ed)



Case Study I: Dining Philosophers (cont'ed)

Step 4: analyze the Labeled Transition System,

- is the system deadlock-free?
- will Alice or Bob starve to death?
- $\cdots$

Tool Needed!

Process Analysis Toolkit

- PAT is a toolset designed for system modeling, simulation and verification.
- 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://pat.comp.nus.edu.sg/

Next lecture: how to use PAT to mechanically analyze CSP models?