Semantics and Verification of Software

Lecture 19: Nondeterminism and Parallelism II (Communicating Sequential Processes)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH AACHEN UNIVERSITY

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/svsw11/

Winter Semester 2011/12

Outline

Communicating Sequential Processes

- Approach: Communicating Sequential Processes (CSP) by T. Hoare and R. Milner
- Models system of processors that
  - have (only) local store and
  - run a sequential program ("process")
- Communication proceeds in the following way:
  - processes communicate along channels
  - process can send/receive on a channel if another process simultaneously performs the complementary I/O operation

  $\Rightarrow$ no buffering (synchronous communication)
- New syntactic domains:

Channel names:	$\alpha, \beta, \gamma, \ldots \in Chn$
Input operations:	$\alpha?x$ where $\alpha \in Chn$, $x \in Var$
Output operations:	$\alpha!a$ where $\alpha \in Chn$, $a \in AExp$
Guarded commands:	$gc \in GCmd$

Syntax of CSP

Definition (Syntax of CSP)

The syntax of CSP is given by

$$\begin{array}{rcl} a & ::= & z \mid x \mid a_1{+}a_2 \mid a_1{-}a_2 \mid a_1{*}a_2 \in AExp \\ b & ::= & t \mid a_1{=}a_2 \mid a_1{>}a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in BExp \\ c & ::= & \texttt{skip} \mid x := a \mid \alpha?x \mid \alpha!a \mid \\ & & c_1; c_2 \mid \texttt{if}\ gc\ \texttt{fi} \mid \texttt{do}\ gc\ \texttt{od} \mid c_1 \parallel c_2 \in Cmd \\ gc & ::= & b \to c \mid b \wedge \alpha?x \to c \mid b \wedge \alpha!a \to c \mid gc_1 \,\Box\, gc_2 \in GCmd \end{array}$$

- In $c_1 \parallel c_2$, statements c_1 and c_2 must not use common variables (only local store)
- Guarded command $gc_1 \,\Box\, gc_2$ represents an alternative
- In $b \to c$, b acts as a guard that enables the execution of c only if evaluated to true
- $b \wedge \alpha?x \to c$ and $b \wedge \alpha!a \to c$ additionally require the respective I/O operation to be enabled
- If none of its alternatives is enabled, a guarded command gc fails (state fail)
- `if` nondeterministically picks an enabled alternative
- A `do` loop is iterated until its body fails

Semantics of CSP I

- Most important aspect: I/O operations
- E.g., $\langle \alpha?x; c, \sigma \rangle$ can only execute if a parallel statement provides corresponding output

$\Rightarrow$ Indicate communication potential by labels

$$L = \{\alpha?z \mid \alpha \in Chn, z \in \mathbb{Z}\} \cup \{\alpha!z \mid \alpha \in Chn, z \in \mathbb{Z}\}$$

- Yields following labeled transitions:

$$\langle \alpha?x; c, \sigma \rangle \xrightarrow{\alpha?z} \langle c, \sigma[x \mapsto z] \rangle \quad (\text{for all } z \in \mathbb{Z})$$
$$\langle \alpha!a; c', \sigma \rangle \xrightarrow{\alpha!z} \langle c', \sigma \rangle \quad (\text{if } \langle a, \sigma \rangle \to z)$$

- Now both statements, if running in parallel, can communicate:

$$\langle (\alpha?x; c) \parallel (\alpha!a; c'), \sigma \rangle \to \langle c \parallel c', \sigma[x \mapsto z] \rangle.$$

- To allow communication with other processes, the following transitions should also be possible (for all $z' \in \mathbb{Z}$):

$$\langle (\alpha?x; c) \parallel (\alpha!a; c'), \sigma \rangle \xrightarrow{\alpha?z'} \langle c \parallel (\alpha!a; c'), \sigma[x \mapsto z'] \rangle$$
$$\langle (\alpha?x; c) \parallel (\alpha!a; c'), \sigma \rangle \xrightarrow{\alpha!z} \langle (\alpha?x; c) \parallel c', \sigma \rangle$$

Definition of labeled transition relation

$$\xrightarrow{\lambda} \subseteq (Cmd \times \Sigma) \times (Cmd \times \Sigma) \cup (GCmd \times \Sigma) \times (Cmd \times \Sigma \cup \{\text{fail}\})$$

(see following slides)

- Marking λ can be a label or empty: $\lambda \in L \cup \{\varepsilon\}$
- Uniform treatment of configurations of the form $\langle c, \sigma \rangle \in Cmd \times \Sigma$ and $\sigma \in \Sigma$:
  - σ interpreted as $\langle *, \sigma \rangle$ with "empty" statement $*$
  - $*$ satisfies $*; c = c; * = * \parallel c = c \parallel * = c$
- Thus: read $\langle x := 0 \parallel *, \sigma \rangle$ as $\langle x := 0, \sigma \rangle$

Semantics of CSP III

Definition (Semantics of CSP)

Rules for statements

$$\frac{}{\langle \texttt{skip}, \sigma\rangle \to \langle *, \sigma\rangle}$$

$$\frac{\langle a, \sigma\rangle \to z}{\langle x := a, \sigma\rangle \to \langle *, \sigma[x \mapsto z]\rangle}$$

$$\frac{}{\langle \alpha?x, \sigma\rangle \xrightarrow{\alpha?z} \langle *, \sigma[x \mapsto z]\rangle}$$

$$\frac{\langle a, \sigma\rangle \to z}{\langle \alpha!a, \sigma\rangle \xrightarrow{\alpha!z} \langle *, \sigma\rangle}$$

$$\frac{\langle c_1, \sigma\rangle \xrightarrow{\lambda} \langle c_1', \sigma'\rangle}{\langle c_1; c_2, \sigma\rangle \xrightarrow{\lambda} \langle c_1'; c_2, \sigma'\rangle}$$

$$\frac{\langle gc, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}{\langle \texttt{if}\ gc\ \texttt{fi}, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}$$

$$\frac{\langle gc, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}{\langle \texttt{do}\ gc\ \texttt{od}, \sigma\rangle \xrightarrow{\lambda} \langle c; \texttt{do}\ gc\ \texttt{od}, \sigma'\rangle}$$

$$\frac{\langle gc, \sigma\rangle \to \mathsf{fail}}{\langle \texttt{do}\ gc\ \texttt{od}, \sigma\rangle \to \langle *, \sigma\rangle}$$

$$\frac{\langle c_1, \sigma\rangle \xrightarrow{\lambda} \langle c_1', \sigma'\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \xrightarrow{\lambda} \langle c_1' \parallel c_2, \sigma'\rangle}$$

$$\frac{\langle c_2, \sigma\rangle \xrightarrow{\lambda} \langle c_2', \sigma'\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \xrightarrow{\lambda} \langle c_1 \parallel c_2', \sigma'\rangle}$$

$$\frac{\langle c_1, \sigma\rangle \xrightarrow{\alpha?z} \langle c_1', \sigma'\rangle, \langle c_2, \sigma\rangle \xrightarrow{\alpha!z} \langle c_2', \sigma\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \to \langle c_1' \parallel c_2', \sigma'\rangle}$$

$$\frac{\langle c_1, \sigma\rangle \xrightarrow{\alpha!z} \langle c_1', \sigma\rangle, \langle c_2, \sigma\rangle \xrightarrow{\alpha?z} \langle c_2', \sigma'\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \to \langle c_1' \parallel c_2', \sigma'\rangle}$$

Semantics of CSP IV

Definition (Semantics of CSP; continued)

Rules for guarded commands:

$$\frac{\langle b, \sigma \rangle \to \mathsf{true}}{\langle b \to c, \sigma \rangle \to \langle c, \sigma \rangle} \qquad \frac{\langle b, \sigma \rangle \to \mathsf{false}}{\langle b \to c, \sigma \rangle \to \mathsf{fail}}$$

$$\frac{\langle b, \sigma \rangle \to \mathsf{true}}{\langle b \wedge \alpha?x \to c, \sigma \rangle \xrightarrow{\alpha?z} \langle c, \sigma[x \mapsto z] \rangle} \qquad \frac{\langle b, \sigma \rangle \to \mathsf{false}}{\langle b \wedge \alpha?x \to c, \sigma \rangle \to \mathsf{fail}}$$

$$\frac{\langle b, \sigma \rangle \to \mathsf{true}, \langle a, \sigma \rangle \to z}{\langle b \wedge \alpha!a \to c, \sigma \rangle \xrightarrow{\alpha!z} \langle c, \sigma \rangle} \qquad \frac{\langle b, \sigma \rangle \to \mathsf{false}}{\langle b \wedge \alpha!a \to c, \sigma \rangle \to \mathsf{fail}}$$

$$\frac{\langle gc_1, \sigma \rangle \xrightarrow{\lambda} \langle c, \sigma' \rangle}{\langle gc_1 \,\Box\, gc_2, \sigma \rangle \xrightarrow{\lambda} \langle c, \sigma' \rangle} \qquad \frac{\langle gc_2, \sigma \rangle \xrightarrow{\lambda} \langle c, \sigma' \rangle}{\langle gc_1 \,\Box\, gc_2, \sigma \rangle \xrightarrow{\lambda} \langle c, \sigma' \rangle}$$

$$\frac{\langle gc_1, \sigma \rangle \to \mathsf{fail}, \langle gc_2, \sigma \rangle \to \mathsf{fail}}{\langle gc_1 \,\Box\, gc_2, \sigma \rangle \to \mathsf{fail}}$$

Outline

CSP Examples

Example 19.1

(on the board)

1. $\mathtt{do}\ (\mathsf{true} \wedge \alpha?x \rightarrow \beta!x)\ \mathtt{od}$
   describes a process that repeatedly receives a value along α and forwards it along β (i.e., a one-place buffer)
2. $\mathtt{do}\ \mathsf{true} \wedge \alpha?x \rightarrow \beta!x\ \mathtt{od} \parallel \mathtt{do}\ \mathsf{true} \wedge \beta?y \rightarrow \gamma!y\ \mathtt{od}$
   specifies a two-place buffer that receives along α and sends along γ (using β for internal communication)
3. Nondeterministic choice between input channels:
   1. $\mathtt{if}\ (\mathsf{true} \wedge \alpha?x \rightarrow c_1 \,\Box\, \mathsf{true} \wedge \beta?y \rightarrow c_2)\ \mathtt{fi}$
   2. $\mathtt{if}\ (\mathsf{true} \rightarrow (\alpha?x; c_1) \,\Box\, \mathsf{true} \rightarrow (\beta?y; c_2))\ \mathtt{fi}$

   Expected: progress whenever environment provides data on α or β
   1. correct
   2. incorrect (can deadlock)