Semantics and Verification of Software

Lecture 17: 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/svsw13/

Summer Semester 2013

Outline

1. Recapitulation: Channel Communication
2. CSP Examples
3. Fairness in CSP

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

Syntax of CSP

Definition (Syntax of CSP)

The syntax of CSP is given by

$$
\begin{aligned}
a &::= z \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \in \mathit{AExp} \\
b &::= t \mid a_1 = a_2 \mid a_1 > a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in \mathit{BExp} \\
c &::= \texttt{skip} \mid x := a \mid \alpha?x \mid \alpha!a \mid \\
&\quad\; c_1; c_2 \mid \texttt{if } \mathit{gc} \texttt{ fi} \mid \texttt{do } \mathit{gc} \texttt{ od} \mid c_1 \parallel c_2 \in \mathit{Cmd} \\
\mathit{gc} &::= b \to c \mid b \wedge \alpha?x \to c \mid b \wedge \alpha!a \to c \mid \mathit{gc}_1 \,\square\, \mathit{gc}_2 \in \mathit{GCmd}
\end{aligned}
$$

- In $c_1 \parallel c_2$, commands c_1 and c_2 must not use common variables (only local store)
- Guarded command $gc_1 \,\square\, gc_2$ 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 command provides corresponding output

$\Rightarrow$ Indicate communication potential by labels

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

- Yields following labeled transitions:

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

- Now both commands, 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 $z' \in \mathbb{Z}$, $\langle a, \sigma \rangle \to z$):

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

Semantics of CSP II

Definition of transition relation

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

(see following slides)

- Marking λ can be a label or empty: $\lambda \in L \cup \{\varepsilon\}$
- Again: uniform treatment of configurations of the form $\langle c, \sigma \rangle \in \mathit{Cmd} \times \Sigma$ and $\sigma \in \Sigma$:
  - σ interpreted as $\langle \downarrow, \sigma \rangle$ with "terminated" command $\downarrow$
  - $\downarrow$ satisfies $\downarrow; c = c; \downarrow = \downarrow \parallel c = c \parallel \downarrow = c$

Definition (Semantics of CSP)

Rules for commands:

$$\frac{}{\langle \texttt{skip}, \sigma\rangle \to \langle \downarrow, \sigma\rangle} \qquad \frac{\langle a, \sigma\rangle \to z}{\langle x := a, \sigma\rangle \to \langle \downarrow, \sigma[x \mapsto z]\rangle}$$

$$\frac{}{\langle \alpha?x, \sigma\rangle \xrightarrow{\alpha?z} \langle \downarrow, \sigma[x \mapsto z]\rangle} \qquad \frac{\langle a, \sigma\rangle \to z}{\langle \alpha!a, \sigma\rangle \xrightarrow{\alpha!z} \langle \downarrow, \sigma\rangle}$$

$$\frac{\langle c_1, \sigma\rangle \xrightarrow{\lambda} \langle c_1', \sigma'\rangle}{\langle c_1; c_2, \sigma\rangle \xrightarrow{\lambda} \langle c_1'; c_2, \sigma'\rangle} \qquad \frac{\langle gc, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}{\langle \texttt{if } gc \texttt{ fi}, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}$$

$$\frac{\langle gc, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}{\langle \texttt{do } gc \texttt{ od}, \sigma\rangle \xrightarrow{\lambda} \langle c; \texttt{do } gc \texttt{ od}, \sigma'\rangle} \qquad \frac{\langle gc, \sigma\rangle \to \mathsf{fail}}{\langle \texttt{do } gc \texttt{ od}, \sigma\rangle \to \langle \downarrow, \sigma\rangle}$$

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

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

Definition (Semantics of CSP; continued)

Rules for guarded commands:

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

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

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

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

$$\frac{\langle gc_1, \sigma\rangle \to \text{fail} \;\; \langle gc_2, \sigma\rangle \to \text{fail}}{\langle gc_1 \,\square\, gc_2, \sigma\rangle \to \text{fail}}$$

Outline

1. Recapitulation: Channel Communication
2. CSP Examples
3. Fairness in CSP

CSP Examples

Example 17.1

(on the board)

1. $\texttt{do}\ (\text{true} \wedge \alpha?x \to \beta!x)\ \texttt{od}$
   describes a process that repeatedly receives a value along α and forwards it along β (i.e., a one-place buffer)

CSP Examples

Example 17.1

(on the board)

1. $\texttt{do}\ (\text{true} \wedge \alpha?x \to \beta!x)\ \texttt{od}$
   describes a process that repeatedly receives a value along α and forwards it along β (i.e., a one-place buffer)
2. $\texttt{do}\ \text{true} \wedge \alpha?x \to \beta!x\ \texttt{od} \parallel \texttt{do}\ \text{true} \wedge \beta?y \to \gamma!y\ \texttt{od}$
   specifies a two-place buffer that receives along α and sends along γ (using β for internal communication)

CSP Examples

Example 17.1

(on the board)

1. $\texttt{do}\ (\text{true} \wedge \alpha?x \to \beta!x)\ \texttt{od}$
   describes a process that repeatedly receives a value along α and forwards it along β (i.e., a one-place buffer)
2. $\texttt{do}\ \text{true} \wedge \alpha?x \to \beta!x\ \texttt{od} \parallel \texttt{do}\ \text{true} \wedge \beta?y \to \gamma!y\ \texttt{od}$
   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. $\texttt{if}\ (\text{true} \wedge \alpha?x \to c_1 \,\square\, \text{true} \wedge \beta?y \to c_2)\ \texttt{fi}$
   2. $\texttt{if}\ (\text{true} \to (\alpha?x; c_1) \,\square\, \text{true} \to (\beta?y; c_2))\ \texttt{fi}$

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

Outline

1. Recapitulation: Channel Communication
2. CSP Examples
3. Fairness in CSP

Fairness I

- Informally: unfair behaviour excludes processes from being executed
- Here: consider parallel composition of $n \geq 1$ sequential programs with executions of the form $\kappa_0 \to \kappa_1 \to \kappa_2 \to \ldots$ where $\kappa_j = \langle c_1^{(j)} \parallel \ldots \parallel c_n^{(j)}, \sigma_j \rangle$ and, for some $1 \leq i \leq n$ and $k_0 \in \mathbb{N}$, $c_i^{(k)} = c_i^{(k_0)}$ for all $k \geq k_0$
- But: only unfair if c_i not enabled

Fairness I

- Informally: unfair behaviour excludes processes from being executed
- Here: consider parallel composition of $n \geq 1$ sequential programs with executions of the form $\kappa_0 \to \kappa_1 \to \kappa_2 \to \ldots$ where $\kappa_j = \langle c_1^{(j)} \parallel \ldots \parallel c_n^{(j)}, \sigma_j \rangle$ and, for some $1 \leq i \leq n$ and $k_0 \in \mathbb{N}$, $c_i^{(k)} = c_i^{(k_0)}$ for all $k \geq k_0$
- But: only unfair if c_i not enabled

Definition 17.2 (Enabledness)

c_i is enabled in configuration $\kappa = \langle c_1 \parallel \ldots \parallel c_n, \sigma \rangle$ if there exists $\kappa' = \langle c_1' \parallel \ldots \parallel c_n', \sigma' \rangle$ with $\kappa \to \kappa'$ and $c_i' \neq c_i$.

Fairness I

- Informally: unfair behaviour excludes processes from being executed
- Here: consider parallel composition of $n \geq 1$ sequential programs with executions of the form $\kappa_0 \to \kappa_1 \to \kappa_2 \to \ldots$ where $\kappa_j = \langle c_1^{(j)} \parallel \ldots \parallel c_n^{(j)}, \sigma_j \rangle$ and, for some $1 \leq i \leq n$ and $k_0 \in \mathbb{N}$, $c_i^{(k)} = c_i^{(k_0)}$ for all $k \geq k_0$
- But: only unfair if c_i not enabled

Definition 17.2 (Enabledness)

c_i is enabled in configuration $\kappa = \langle c_1 \parallel \ldots \parallel c_n, \sigma \rangle$ if there exists $\kappa' = \langle c_1' \parallel \ldots \parallel c_n', \sigma' \rangle$ with $\kappa \to \kappa'$ and $c_i' \neq c_i$.

Example 17.3

1. $x := 0$ enabled in $\langle x := 0 \parallel y := 1, \sigma \rangle$ (actually always enabled)
2. $\alpha?x$ enabled in $\langle \alpha?x \parallel \alpha!0, \sigma \rangle$
3. $\alpha?x$ not enabled in $\langle \alpha?x \parallel \beta!1, \sigma \rangle$

Fairness II

Definition 17.4 (Fairness)

An execution $\kappa_0 \to \kappa_1 \to \kappa_2 \to \ldots$ where $\kappa_j = \langle c_1^{(j)} \parallel \ldots \parallel c_n^{(j)}, \sigma_j \rangle$ and, for some $1 \leq i \leq n$ and $k_0 \in \mathbb{N}$, $c_i^{(k)} = c_i^{(k_0)}$ for all $k \geq k_0$ is called

1. strongly unfair if $c_i^{(k)}$ is enabled in κ_k for all $k \geq k_0$
2. weakly unfair if $c_i^{(k)}$ is enabled in κ_k for infinitely many $k \geq k_0$

Fairness II

Definition 17.4 (Fairness)

An execution $\kappa_0 \to \kappa_1 \to \kappa_2 \to \ldots$ where $\kappa_j = \langle c_1^{(j)} \parallel \ldots \parallel c_n^{(j)}, \sigma_j \rangle$ and, for some $1 \leq i \leq n$ and $k_0 \in \mathbb{N}$, $c_i^{(k)} = c_i^{(k_0)}$ for all $k \geq k_0$ is called

1. strongly unfair if $c_i^{(k)}$ is enabled in κ_k for all $k \geq k_0$
2. weakly unfair if $c_i^{(k)}$ is enabled in κ_k for infinitely many $k \geq k_0$

Example 17.5

1. $\langle \text{do true} \to x := x + 1 \text{ od} \parallel y := y + 1, \ldots \rangle$
   $\to \quad \langle x := x + 1; \text{do true} \to x := x + 1 \text{ od} \parallel y := y + 1, \ldots \rangle$
   $\to \quad \langle \text{do true} \to x := x + 1 \text{ od} \parallel y := y + 1, \ldots \rangle \to \ldots$
   is strongly unfair since $y := y + 1$ is always enabled

Fairness II

Definition 17.4 (Fairness)

An execution $\kappa_0 \to \kappa_1 \to \kappa_2 \to \ldots$ where $\kappa_j = \langle c_1^{(j)} \parallel \ldots \parallel c_n^{(j)}, \sigma_j \rangle$ and, for some $1 \leq i \leq n$ and $k_0 \in \mathbb{N}$, $c_i^{(k)} = c_i^{(k_0)}$ for all $k \geq k_0$ is called

1. strongly unfair if $c_i^{(k)}$ is enabled in κ_k for all $k \geq k_0$
2. weakly unfair if $c_i^{(k)}$ is enabled in κ_k for infinitely many $k \geq k_0$

Example 17.5

1. $\langle \text{do true} \to x := x + 1 \text{ od} \parallel y := y + 1, \ldots \rangle$
   $\to \quad \langle x := x + 1; \text{do true} \to x := x + 1 \text{ od} \parallel y := y + 1, \ldots \rangle$
   $\to \quad \langle \text{do true} \to x := x + 1 \text{ od} \parallel y := y + 1, \ldots \rangle \to \ldots$
   is strongly unfair since $y := y + 1$ is always enabled
2. $\langle \text{do true} \to x := x + 1 \text{ od} \parallel \alpha!1 \parallel \alpha?y, \ldots \rangle$
   $\to \quad \langle x := x + 1; \text{do true} \to x := x + 1 \text{ od} \parallel \alpha!1 \parallel \alpha?y, \ldots \rangle$
   $\to \quad \langle \text{do true} \to x := x + 1 \text{ od} \parallel \alpha!1 \parallel \alpha?y, \ldots \rangle \to \ldots$
   is strongly unfair since both I/O operations are always enabled

Fairness II

Definition 17.4 (Fairness)

An execution $\kappa_0 \to \kappa_1 \to \kappa_2 \to \ldots$ where $\kappa_j = \langle c_1^{(j)} \parallel \ldots \parallel c_n^{(j)}, \sigma_j \rangle$ and, for some $1 \leq i \leq n$ and $k_0 \in \mathbb{N}$, $c_i^{(k)} = c_i^{(k_0)}$ for all $k \geq k_0$ is called

1. strongly unfair if $c_i^{(k)}$ is enabled in κ_k for all $k \geq k_0$
2. weakly unfair if $c_i^{(k)}$ is enabled in κ_k for infinitely many $k \geq k_0$

Example 17.5

1. $\langle \texttt{do true} \to x := x + 1 \texttt{ od} \parallel y := y + 1, \ldots \rangle$
   $\to \quad \langle x := x + 1; \texttt{do true} \to x := x + 1 \texttt{ od} \parallel y := y + 1, \ldots \rangle$
   $\to \quad \langle \texttt{do true} \to x := x + 1 \texttt{ od} \parallel y := y + 1, \ldots \rangle \to \ldots$
   is strongly unfair since $y := y + 1$ is always enabled
2. $\langle \texttt{do true} \to x := x + 1 \texttt{ od} \parallel \alpha!1 \parallel \alpha?y, \ldots \rangle$
   $\to \quad \langle x := x + 1; \texttt{do true} \to x := x + 1 \texttt{ od} \parallel \alpha!1 \parallel \alpha?y, \ldots \rangle$
   $\to \quad \langle \texttt{do true} \to x := x + 1 \texttt{ od} \parallel \alpha!1 \parallel \alpha?y, \ldots \rangle \to \ldots$
   is strongly unfair since both I/O operations are always enabled
3. $\langle \texttt{do } \alpha!1 \to \texttt{skip od} \parallel \texttt{do } \alpha?x \to \texttt{skip od} \parallel \alpha?y, \ldots \rangle$
   $\to \quad \langle \texttt{skip; do } \alpha!1 \to \texttt{skip od} \parallel \texttt{skip; do } \alpha?x \to \texttt{skip od} \parallel \alpha?y, \ldots \rangle$
   $\to \quad \langle \texttt{skip; do } \alpha!1 \to \texttt{skip od} \parallel \texttt{do } \alpha?x \to \texttt{skip od} \parallel \alpha?y, \ldots \rangle$
   $\to \quad \langle \texttt{do } \alpha!1 \to \texttt{skip od} \parallel \texttt{do } \alpha?x \to \texttt{skip od} \parallel \alpha?y, \ldots \rangle \to \ldots$
   is weakly unfair since $\alpha?y$ is enabled in every third configuration