

# Semantics and Verification of Software

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

Thomas Noll

Lehrstuhl für Informatik 2  
(Software Modeling and Verification)



[noll@cs.rwth-aachen.de](mailto:noll@cs.rwth-aachen.de)

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

Winter Semester 2011/12

1 Repetition: Channel Communication

2 CSP Examples

# 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
  - ⇒ no buffering (**synchronous** communication)
- New **syntactic domains**:

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

## Definition (Syntax of CSP)

The syntax of CSP is given by

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

- In  $c_1 \parallel c_2$ , statements  $c_1$  and  $c_2$  must **not use common variables** (only local store)
- **Guarded command**  $gc_1 \square gc_2$  represents an **alternative**
- In  $b \rightarrow c$ ,  $b$  acts as a **guard** that enables the execution of  $c$  only if evaluated to **true**
- $b \wedge \alpha?x \rightarrow c$  and  $b \wedge \alpha!a \rightarrow c$  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

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

⇒ 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**:

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

$$\langle \alpha!a; c', \sigma \rangle \xrightarrow{\alpha!z} \langle c', \sigma \rangle \quad (\text{if } \langle a, \sigma \rangle \rightarrow z)$$

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

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

- 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**  $\lambda$  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  $\sigma \in \Sigma$ :
  - $\sigma$  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  $\langle x := 0, \sigma \rangle$

## Definition (Semantics of CSP)

Rules for **statements**

$$\frac{}{\langle \text{skip}, \sigma \rangle \rightarrow \langle *, \sigma \rangle}$$

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

$$\frac{}{\langle c_1, \sigma \rangle \xrightarrow{\lambda} \langle c'_1, \sigma' \rangle}$$

$$\frac{}{\langle c_1; c_2, \sigma \rangle \xrightarrow{\lambda} \langle c'_1; c_2, \sigma' \rangle}$$

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

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

$$\frac{}{\langle c_1, \sigma \rangle \xrightarrow{\lambda} \langle c'_1, \sigma' \rangle}$$

$$\frac{}{\langle c_1 \parallel c_2, \sigma \rangle \xrightarrow{\lambda} \langle c'_1 \parallel c_2, \sigma' \rangle}$$

$$\frac{}{\langle c_1, \sigma \rangle \xrightarrow{\alpha?z} \langle c'_1, \sigma' \rangle, \langle c_2, \sigma \rangle \xrightarrow{\alpha!z} \langle c'_2, \sigma' \rangle}$$

$$\frac{}{\langle c_1 \parallel c_2, \sigma \rangle \rightarrow \langle c'_1 \parallel c'_2, \sigma' \rangle}$$

$$\frac{}{\langle a, \sigma \rangle \rightarrow z}$$

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

$$\frac{}{\langle a, \sigma \rangle \rightarrow z}$$

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

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

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

$$\frac{}{\langle gc, \sigma \rangle \rightarrow \text{fail}}$$

$$\frac{}{\langle \text{do } gc \text{ od}, \sigma \rangle \rightarrow \langle *, \sigma \rangle}$$

$$\frac{}{\langle c_2, \sigma \rangle \xrightarrow{\lambda} \langle c'_2, \sigma' \rangle}$$

$$\frac{}{\langle c_1 \parallel c_2, \sigma \rangle \xrightarrow{\lambda} \langle c_1 \parallel c'_2, \sigma' \rangle}$$

$$\frac{}{\langle c_1, \sigma \rangle \xrightarrow{\alpha!z} \langle c'_1, \sigma' \rangle, \langle c_2, \sigma \rangle \xrightarrow{\alpha?z} \langle c'_2, \sigma' \rangle}$$

$$\frac{}{\langle c_1 \parallel c_2, \sigma \rangle \rightarrow \langle c'_1 \parallel c'_2, \sigma' \rangle}$$

## Definition (Semantics of CSP; continued)

Rules for **guarded commands**:

$$\begin{array}{c}
 \frac{\langle b, \sigma \rangle \rightarrow \text{true}}{\langle b \rightarrow c, \sigma \rangle \rightarrow \langle c, \sigma \rangle} \\
 \frac{\langle b, \sigma \rangle \rightarrow \text{true}}{\langle b \wedge \alpha?x \rightarrow c, \sigma \rangle \xrightarrow{\alpha?z} \langle c, \sigma[x \mapsto z] \rangle} \\
 \frac{\langle b, \sigma \rangle \rightarrow \text{true}, \langle a, \sigma \rangle \rightarrow z}{\langle b \wedge \alpha!a \rightarrow c, \sigma \rangle \xrightarrow{\alpha!z} \langle c, \sigma \rangle} \\
 \frac{\langle gc_1, \sigma \rangle \xrightarrow{\lambda} \langle c, \sigma' \rangle}{\langle gc_1 \sqcap gc_2, \sigma \rangle \xrightarrow{\lambda} \langle c, \sigma' \rangle} \\
 \frac{\langle gc_1, \sigma \rangle \rightarrow \text{fail}, \langle gc_2, \sigma \rangle \rightarrow \text{fail}}{\langle gc_1 \sqcap gc_2, \sigma \rangle \rightarrow \text{fail}}
 \end{array}
 \qquad
 \begin{array}{c}
 \frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle b \rightarrow c, \sigma \rangle \rightarrow \text{fail}} \\
 \frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle b \wedge \alpha?x \rightarrow c, \sigma \rangle \rightarrow \text{fail}} \\
 \frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle b \wedge \alpha!a \rightarrow c, \sigma \rangle \rightarrow \text{fail}} \\
 \frac{\langle gc_2, \sigma \rangle \xrightarrow{\lambda} \langle c, \sigma' \rangle}{\langle gc_1 \sqcap gc_2, \sigma \rangle \xrightarrow{\lambda} \langle c, \sigma' \rangle}
 \end{array}$$



1 Repetition: Channel Communication

2 CSP Examples

## Example 19.1

(on the board)

①  $\text{do } (\text{true} \wedge \alpha?x \rightarrow \beta!x) \text{ od}$

describes a process that repeatedly receives a value along  $\alpha$  and forwards it along  $\beta$  (i.e., a **one-place buffer**)

## Example 19.1

(on the board)

- ①  $\text{do } (\text{true} \wedge \alpha?x \rightarrow \beta!x) \text{ od}$

describes a process that repeatedly receives a value along  $\alpha$  and forwards it along  $\beta$  (i.e., a **one-place buffer**)

- ②  $\text{do } \text{true} \wedge \alpha?x \rightarrow \beta!x \text{ od} \parallel \text{do } \text{true} \wedge \beta?y \rightarrow \gamma!y \text{ od}$

specifies a **two-place buffer** that receives along  $\alpha$  and sends along  $\gamma$  (using  $\beta$  for internal communication)

## Example 19.1

(on the board)

- ①  $\text{do } (\text{true} \wedge \alpha?x \rightarrow \beta!x) \text{ od}$

describes a process that repeatedly receives a value along  $\alpha$  and forwards it along  $\beta$  (i.e., a **one-place buffer**)

- ②  $\text{do } \text{true} \wedge \alpha?x \rightarrow \beta!x \text{ od} \parallel \text{do } \text{true} \wedge \beta?y \rightarrow \gamma!y \text{ od}$

specifies a **two-place buffer** that receives along  $\alpha$  and sends along  $\gamma$  (using  $\beta$  for internal communication)

- ③ Nondeterministic choice between input channels:

- ①  $\text{if } (\text{true} \wedge \alpha?x \rightarrow c_1 \square \text{true} \wedge \beta?y \rightarrow c_2) \text{ fi}$

- ②  $\text{if } (\text{true} \rightarrow (\alpha?x; c_1) \square \text{true} \rightarrow (\beta?y; c_2)) \text{ fi}$

Expected: progress whenever environment provides data on  $\alpha$  or  $\beta$

- ① correct

- ② incorrect (can **deadlock**)