

Semantics and Verification of Software

Lecture 16: Nondeterminism and Parallelism I (Shared-Variables and Channel Communication)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)



noll@cs.rwth-aachen.de

<http://www-i2.informatik.rwth-aachen.de/i2/svsw13/>

Summer Semester 2013

- 1 Introduction
- 2 Shared-Variables Communication
- 3 Channel Communication

- Essential question: what is the meaning of

$$c_1 \parallel c_2$$

(**parallel** execution of $c_1, c_2 \in \text{Cmd}$)?

- Easy to answer when state spaces are **disjoint**:

$$\langle x := 1 \parallel y := 2, \sigma \rangle \rightarrow \sigma[x \mapsto 1, y \mapsto 2]$$

(no interaction $\Rightarrow$ corresponds to sequential execution)

- But what if variables are **shared**?

$$(x := 1 \parallel x := 2); \text{if } x = 1 \text{ then } c_1 \text{ else } c_2$$

(runs c_1 or c_2 depending on execution order of initial assignments)

- Even more complicated for **non-atomic assignments**...

Observation: **parallelism** introduces new phenomena

Example 16.1

$$\begin{array}{c} x := 0; \\ (x := x + 1 \parallel x := x + 2) \end{array} \quad \text{value of } x: 0123$$

$$\begin{array}{cc} 13 & 2 \end{array}$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2, 1, or 3
- If **exclusive (write) access** to shared memory and **atomic execution** of assignments guaranteed
⇒ only possible outcome: 3

The problem arises due to the combination of

- **parallelism** and
- **interaction** (here: via shared memory)

Conclusion

When modeling parallel systems, the precise description of the mechanisms of both **parallelism** and **interaction** is crucially important.

- Thus: “classical” model for sequential systems

System : Input $\rightarrow$ Output

(**transformational systems**) is not adequate

- Missing: aspect of **interaction**
- Rather: **reactive systems** which interact with environment and among themselves
- Main interest: not terminating computations but **infinite behavior** (system maintains ongoing interaction with environment)
- Examples:
 - operating systems
 - embedded systems controlling mechanical or electrical devices (planes, cars, home appliances, ...)
 - power plants, production lines, ...

Here: study of parallelism in connection with different kinds of interaction

- ① Shared-variables communication (ParWHILE)
- ② Channel communication (CSP)
- ③ Algebraic approaches (CCS)

- 1 Introduction
- 2 Shared-Variables Communication
- 3 Channel Communication

Definition 16.2 (Syntax of ParWHILE)

$a ::= z \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \in AExp$

$b ::= t \mid a_1 = a_2 \mid a_1 > a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in BExp$

$c ::= \text{skip} \mid x := a \mid c_1; c_2 \mid \text{if } b \text{ then } c_1 \text{ else } c_2 \mid \text{while } b \text{ do } c \mid$
 $c_1 \parallel c_2 \in Cmd$

- Approach for defining semantics:
 - assignments are executed **atomically**
 - parallelism is modeled by **interleaving**, i.e., the actions of parallel statements are merged

⇒ Reduction of parallelism to **nondeterminism + sequential execution**
(similar to multitasking on sequential computers)

- Requires **single-step execution relation** for statements
(cf. Exercise 2.1)
- To minimize number of rules: uniform treatment of configurations of the form $\langle c, \sigma \rangle \in \text{Cmd} \times \Sigma$ and $\sigma \in \Sigma$:
 - σ interpreted as $\langle \downarrow, \sigma \rangle$ with **“terminated” command $\downarrow$**
 - $\downarrow$ satisfies $\downarrow; c = c; \downarrow = \downarrow \parallel c = c \parallel \downarrow = c$
- Thus: read $\langle x := 0 \parallel \downarrow, \sigma \rangle$ as $\langle x := 0, \sigma \rangle$

Definition 16.3 (Single-step execution relation)

The **single-step execution relation**,

$$\rightarrow_1 \subseteq (Cmd \times \Sigma) \times (Cmd \times \Sigma),$$

is defined by the following rules:

$$\frac{}{\langle \text{skip}, \sigma \rangle \rightarrow_1 \langle \downarrow, \sigma \rangle}$$

$$\frac{\langle a, \sigma \rangle \rightarrow z}{\langle x := a, \sigma \rangle \rightarrow_1 \langle \downarrow, \sigma[x \mapsto z] \rangle}$$

$$\langle b, \sigma \rangle \rightarrow \text{true}$$

$$\frac{}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow_1 \langle c_1, \sigma \rangle}$$

$$\langle b, \sigma \rangle \rightarrow \text{false}$$

$$\frac{}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow_1 \langle c_2, \sigma \rangle}$$

$$\langle b, \sigma \rangle \rightarrow \text{true}$$

$$\frac{}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow_1 \langle c; \text{while } b \text{ do } c, \sigma \rangle}$$

$$\langle b, \sigma \rangle \rightarrow \text{false}$$

$$\frac{}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow_1 \langle \downarrow, \sigma \rangle}$$

$$\langle c_1, \sigma \rangle \rightarrow_1 \langle c'_1, \sigma' \rangle$$

$$\frac{}{\langle c_1; c_2, \sigma \rangle \rightarrow_1 \langle c'_1; c_2, \sigma' \rangle}$$

$$\langle c_1, \sigma \rangle \rightarrow_1 \langle c'_1, \sigma' \rangle$$

$$\frac{}{\langle c_1 \parallel c_2, \sigma \rangle \rightarrow_1 \langle c'_1 \parallel c_2, \sigma' \rangle}$$

$$\langle c_2, \sigma \rangle \rightarrow_1 \langle c'_2, \sigma' \rangle$$

$$\frac{}{\langle c_1 \parallel c_2, \sigma \rangle \rightarrow_1 \langle c_1 \parallel c'_2, \sigma' \rangle}$$

Example 16.4

Let $c := (x := 1 \parallel x := 2); \text{if } x = 1 \text{ then } c_1 \text{ else } c_2$ and $\sigma \in \Sigma$.

$$\begin{aligned}
 & \langle c, \sigma \rangle \rightarrow_1 \langle x := 2; \text{if } x = 1 \text{ then } c_1 \text{ else } c_2, \sigma[x \mapsto 1] \rangle \\
 & \quad \frac{\langle 1, \sigma \rangle \rightarrow 1}{\langle x := 1, \sigma \rangle \rightarrow_1 \langle \downarrow, \sigma[x \mapsto 1] \rangle} \\
 & \text{since } \frac{\langle x := 1, \sigma \rangle \rightarrow_1 \langle \downarrow, \sigma[x \mapsto 1] \rangle}{\langle x := 1 \parallel x := 2, \sigma \rangle \rightarrow_1 \langle \downarrow \parallel x := 2, \sigma[x \mapsto 1] \rangle} \\
 & \rightarrow_1 \langle \text{if } x = 1 \text{ then } c_1 \text{ else } c_2, \sigma[x \mapsto 2] \rangle \\
 & \quad \frac{\langle 2, \sigma \rangle \rightarrow 2}{\langle x := 2, \sigma \rangle \rightarrow_1 \langle \downarrow, \sigma[x \mapsto 2] \rangle} \\
 & \rightarrow_1 \langle c_2, \sigma[x \mapsto 2] \rangle \\
 & \text{since } \frac{\langle x, \sigma[x \mapsto 2] \rangle \rightarrow 2 \quad \langle 1, \sigma[x \mapsto 2] \rangle \rightarrow 1}{\langle x = 1, \sigma[x \mapsto 2] \rangle \rightarrow \text{false}}
 \end{aligned}$$

Analogously: $\langle c, \sigma \rangle \rightarrow_1^3 \langle c_1, \sigma[x \mapsto 1] \rangle$

- 1 Introduction
- 2 Shared-Variables Communication
- 3 Channel Communication

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 16.5 (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 $c_1 \parallel c_2$, 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 $b \rightarrow c$, 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 command provides corresponding output

⇒ Indicate **communication potential** by labels

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

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

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

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

Definition of **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: $\lambda \in L \cup \{\varepsilon\}$
- Again: uniform treatment of configurations of the form $\langle c, \sigma \rangle \in Cmd \times \Sigma$ 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 16.6 (Semantics of CSP)

Rules for **commands**:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Definition 16.6 (Semantics of CSP; continued)

Rules for **guarded commands**:

$$\frac{\langle b, \sigma \rangle \rightarrow \text{true}}{\langle b \rightarrow c, \sigma \rangle \rightarrow \langle c, \sigma \rangle}$$

$$\frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle b \rightarrow c, \sigma \rangle \rightarrow \text{fail}}$$

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

$$\frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle b \wedge \alpha?x \rightarrow c, \sigma \rangle \rightarrow \text{fail}}$$

$$\frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle a, \sigma \rangle \rightarrow z}{\langle b \wedge \alpha!a \rightarrow c, \sigma \rangle \xrightarrow{\alpha!z} \langle c, \sigma \rangle}$$

$$\frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle b \wedge \alpha!a \rightarrow c, \sigma \rangle \rightarrow \text{fail}}$$

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

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

$$\frac{\langle gc_1, \sigma \rangle \rightarrow \text{fail} \quad \langle gc_2, \sigma \rangle \rightarrow \text{fail}}{\langle gc_1 \square gc_2, \sigma \rangle \rightarrow \text{fail}}$$