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)

RWTH AACHEN UNIVERSITY

noll@cs.rwth-aachen.de

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

Summer Semester 2013

Outline

Motivation

- Essential question: what is the meaning of

$$c_1 \parallel c_2$$

(parallel execution of $c_1, c_2 \in Cmd$)?

Motivation

- Essential question: what is the meaning of

$$c_1 \parallel c_2$$

(parallel execution of $c_1, c_2 \in Cmd$)?

- Easy to answer when state spaces are disjoint:

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

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

Motivation

- Essential question: what is the meaning of

$$c_1 \parallel c_2$$

(parallel execution of $c_1, c_2 \in Cmd$)?

- Easy to answer when state spaces are disjoint:

$$\langle \texttt{x := 1} \parallel \texttt{y := 2}, \sigma\rangle \to \sigma[\texttt{x} \mapsto 1, \texttt{y} \mapsto 2]$$

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

- But what if variables are shared?

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

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

Motivation

- Essential question: what is the meaning of

$$c_1 \parallel c_2$$

(parallel execution of $c_1, c_2 \in Cmd$)?

- Easy to answer when state spaces are disjoint:

$$\langle \texttt{x := 1} \parallel \texttt{y := 2}, \sigma\rangle \to \sigma[\texttt{x} \mapsto 1, \texttt{y} \mapsto 2]$$

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

- But what if variables are shared?

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

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

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

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2)$$

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2)$$

- At first glance: x is assigned 3

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2)$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$
$$1$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$
$$\quad 1 \qquad\qquad\qquad 2$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(\underbrace{x := x+1}_{1} \parallel \underbrace{x := x+2}_{2}) \qquad \text{value of } x\text{: } 1$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := \underset{2}{x + 2}) \qquad \text{value of } x\text{: } 2$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2,

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2,

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$
$$1$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2,

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(\underset{1}{x} := x + 1 \parallel x := \underset{2}{x} + 2) \qquad \text{value of } x\text{: } 0$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2,

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(\underset{1}{x := x + 1} \parallel \underset{2}{x := x + 2}) \qquad \text{value of } x\text{: } 2$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2,

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 1$$
$$1$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2, 1,

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2, 1,

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$
$$2$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2, 1,

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 2$$
$$2$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2, 1,

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 2$$
$$3$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2, 1,

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 3$$
$$3$$

- 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

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 16.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2)$$

- 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
  $\Rightarrow$ only possible outcome: 3

Parallelism and Interaction

The problem arises due to the combination of

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

Parallelism and Interaction

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.

Reactive Systems

- Thus: "classical" model for sequential systems

$$\text{System} : \text{Input} \to \text{Output}$$

  (transformational systems) is not adequate
- Missing: aspect of interaction

Reactive Systems

- Thus: “classical” model for sequential systems

$$\text{System} : \text{Input} \to \text{Output}$$

  (transformational systems) is not adequate
- Missing: aspect of interaction
- Rather: reactive systems which interact with environment and among themselves

Reactive Systems

- Thus: "classical" model for sequential systems

$$\text{System} : \text{Input} \rightarrow \text{Output}$$

  (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, ...

Overview

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

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

Outline

The ParWHILE Language

Definition 16.2 (Syntax of ParWHILE)

$$
\begin{aligned}
a &::= z \mid x \mid a_1\texttt{+}a_2 \mid a_1\texttt{-}a_2 \mid a_1\texttt{*}a_2 \in AExp \\
b &::= t \mid a_1\texttt{=}a_2 \mid a_1\texttt{>}a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in BExp \\
c &::= \texttt{skip} \mid x \texttt{ := } a \mid c_1\texttt{;}c_2 \mid \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \mid \texttt{while } b \texttt{ do } c \mid \\
&\qquad c_1 \parallel c_2 \in Cmd
\end{aligned}
$$

Semantics of ParWHILE

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

$\Rightarrow$ Reduction of parallelism to nondeterminism + sequential execution (similar to multitasking on sequential computers)

Semantics of ParWHILE

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

$\Rightarrow$ 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 $\langle c, \sigma \rangle \in Cmd \times \Sigma$ 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,

$$\to_1 \subseteq (Cmd \times \Sigma) \times (Cmd \times \Sigma),$$

is defined by the following rules:

$$\frac{}{\langle \texttt{skip}, \sigma\rangle \to_1 \langle \downarrow, \sigma\rangle} \qquad \frac{\langle a, \sigma\rangle \to z}{\langle x := a, \sigma\rangle \to_1 \langle \downarrow, \sigma[x \mapsto z]\rangle}$$

$$\frac{\langle b, \sigma\rangle \to \mathsf{true}}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, \sigma\rangle \to_1 \langle c_1, \sigma\rangle} \qquad \frac{\langle b, \sigma\rangle \to \mathsf{false}}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, \sigma\rangle \to_1 \langle c_2, \sigma\rangle}$$

$$\frac{\langle b, \sigma\rangle \to \mathsf{true}}{\langle \texttt{while}\ b\ \texttt{do}\ c, \sigma\rangle \to_1 \langle c; \texttt{while}\ b\ \texttt{do}\ c, \sigma\rangle}$$

$$\frac{\langle b, \sigma\rangle \to \mathsf{false}}{\langle \texttt{while}\ b\ \texttt{do}\ c, \sigma\rangle \to_1 \langle \downarrow, \sigma\rangle} \qquad \frac{\langle c_1, \sigma\rangle \to_1 \langle c_1', \sigma'\rangle}{\langle c_1; c_2, \sigma\rangle \to_1 \langle c_1'; c_2, \sigma'\rangle}$$

$$\frac{\langle c_1, \sigma\rangle \to_1 \langle c_1', \sigma'\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \to_1 \langle c_1' \parallel c_2, \sigma'\rangle} \qquad \frac{\langle c_2, \sigma\rangle \to_1 \langle c_2', \sigma'\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \to_1 \langle c_1 \parallel c_2', \sigma'\rangle}$$

Example 16.4

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

Example 16.4

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

$$\langle c, \sigma \rangle \rightarrow_1 \langle x := 2; \texttt{if}\ x = 1\ \texttt{then}\ c_1\ \texttt{else}\ c_2, \sigma[x \mapsto 1] \rangle$$

$$\text{since } \frac{\dfrac{\overline{\langle 1, \sigma \rangle \rightarrow 1}}{\langle x := 1, \sigma \rangle \rightarrow_1 \langle \downarrow, \sigma[x \mapsto 1] \rangle}}{\langle x := 1 \parallel x := 2, \sigma \rangle \rightarrow_1 \langle \downarrow \parallel x := 2, \sigma[x \mapsto 1] \rangle}$$

Example 16.4

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

$$\langle c, \sigma \rangle \to_1 \langle x := 2; \texttt{if}\ x = 1\ \texttt{then}\ c_1\ \texttt{else}\ c_2, \sigma[x \mapsto 1] \rangle$$

$$\text{since}\ \frac{\dfrac{\overline{\langle 1, \sigma \rangle \to 1}}{\langle x := 1, \sigma \rangle \to_1 \langle \downarrow, \sigma[x \mapsto 1] \rangle}}{\langle x := 1 \parallel x := 2, \sigma \rangle \to_1 \langle \downarrow \parallel x := 2, \sigma[x \mapsto 1] \rangle}$$

$$\to_1 \langle \texttt{if}\ x = 1\ \texttt{then}\ c_1\ \texttt{else}\ c_2, \sigma[x \mapsto 2] \rangle$$

$$\text{since}\ \frac{\overline{\langle 2, \sigma \rangle \to 2}}{\langle x := 2, \sigma \rangle \to_1 \langle \downarrow, \sigma[x \mapsto 2] \rangle}$$

Example 16.4

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

$$\langle c, \sigma\rangle \to_1 \langle x := 2; \texttt{if } x = 1 \texttt{ then } c_1 \texttt{ else } c_2, \sigma[x \mapsto 1]\rangle$$

$$\text{since } \frac{\dfrac{\overline{\langle 1, \sigma\rangle \to 1}}{\langle x := 1, \sigma\rangle \to_1 \langle \downarrow, \sigma[x \mapsto 1]\rangle}}{\langle x := 1 \parallel x := 2, \sigma\rangle \to_1 \langle \downarrow \parallel x := 2, \sigma[x \mapsto 1]\rangle}$$

$$\to_1 \langle \texttt{if } x = 1 \texttt{ then } c_1 \texttt{ else } c_2, \sigma[x \mapsto 2]\rangle$$

$$\text{since } \frac{\overline{\langle 2, \sigma\rangle \to 2}}{\langle x := 2, \sigma\rangle \to_1 \langle \downarrow, \sigma[x \mapsto 2]\rangle}$$

$$\to_1 \langle c_2, \sigma[x \mapsto 2]\rangle$$

$$\text{since } \frac{\overline{\langle x, \sigma[x \mapsto 2]\rangle \to 2} \quad \overline{\langle 1, \sigma[x \mapsto 2]\rangle \to 1}}{\langle x = 1, \sigma[x \mapsto 2]\rangle \to \mathsf{false}}$$

Example 16.4

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

$$\langle c, \sigma \rangle \to_1 \langle x := 2; \texttt{if } x = 1 \texttt{ then } c_1 \texttt{ else } c_2, \sigma[x \mapsto 1] \rangle$$

$$\text{since } \frac{\dfrac{\dfrac{}{\langle 1, \sigma \rangle \to 1}}{\langle x := 1, \sigma \rangle \to_1 \langle \downarrow, \sigma[x \mapsto 1] \rangle}}{\langle x := 1 \parallel x := 2, \sigma \rangle \to_1 \langle \downarrow \parallel x := 2, \sigma[x \mapsto 1] \rangle}$$

$$\to_1 \langle \texttt{if } x = 1 \texttt{ then } c_1 \texttt{ else } c_2, \sigma[x \mapsto 2] \rangle$$

$$\text{since } \frac{\dfrac{}{\langle 2, \sigma \rangle \to 2}}{\langle x := 2, \sigma \rangle \to_1 \langle \downarrow, \sigma[x \mapsto 2] \rangle}$$

$$\to_1 \langle c_2, \sigma[x \mapsto 2] \rangle$$

$$\text{since } \frac{\dfrac{}{\langle x, \sigma[x \mapsto 2] \rangle \to 2} \quad \dfrac{}{\langle 1, \sigma[x \mapsto 2] \rangle \to 1}}{\langle x = 1, \sigma[x \mapsto 2] \rangle \to \text{false}}$$

Analogously: $\langle c, \sigma \rangle \to_1^3 \langle c_1, \sigma[x \mapsto 1] \rangle$

Outline

Communicating Sequential Processes

- Approach: Communicating Sequential Processes (CSP) by T. Hoare and R. Milner

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")

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)

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:	$gc \in GCmd$

Definition 16.5 (Syntax of CSP)

The syntax of CSP is given by

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

Definition 16.5 (Syntax of CSP)

The syntax of CSP is given by

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

- In $c_1 \parallel c_2$, commands 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 command provides corresponding output

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

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

- Yields following labeled transitions:

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

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

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

- 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 commands, if running in parallel, can communicate:

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

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

⇒ 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 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 $z' \in \mathbb{Z}$, $\langle a, \sigma \rangle \to z$):

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

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:

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

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

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

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

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

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

Definition 16.6 (Semantics of CSP; continued)

Rules for guarded commands:

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

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

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

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

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