Lecture 5: Operational Semantics

Concurrency Theory

Summer 2024

Dr. Stephan Mennicke

May 7th, 2024

TU Dresden, Knowledge-Based Systems Group

Overview

Part 0: Completing the Introduction

- learning about *bisimilarity* and *bisimulations*

Part 1: Semantics of (Sequential) Programming Languages

- WHILE – an old friend
- denotational semantics (a baseline and an exercise of the inductive method)
- natural semantics and (structural) operational semantics (**today**)

Part 2: Towards Parallel Programming Languages

- the Calculus of Communicating Processes (CCS)
- algebraic properties of CCS
- the untold story of Hennessy and Milner
- bisimilarity and its success story
- deep-dive into induction and coinduction

Overview

Part 3: Expressive Power

- Calculus of Communicating Systems (CCS)
- Petri nets

Review: Direct Style Semantics

Theorem 1: Let $f : D \to D$ be a continuous function on the ccpo $\langle D, \preccurlyeq \rangle$ with least element $\bot$. Then

$$\mathsf{FIX}\ f = \bigsqcup \{ f^n \bot \mid n \geq 0 \}$$

defines an element of D, and this element is the least fixed point of f.

Proof: Since f is continuous, it is monotone and $\bigsqcup\{f\,d \mid d \in Y\} = f(\bigsqcup Y)$ for all non-empty chains Y.

First observe that $\{f^n \bot \mid n \geq 0\}$ is non-empty by $f^0 \bot = \bot$. It holds that $f^0 \bot = \bot \preccurlyeq f^1 \bot = f \bot$ since $\bot$ is the least element of D. By an inductive argument, we get that $f^m \bot \preccurlyeq f^{m+1} \bot$ for all $m \geq 0$ since f is monotone. By reflexivity and transitivity of $\preccurlyeq$ we get $f^m \bot \preccurlyeq f^n \bot$ whenever $m \leq n$. Therefore, $\{f^n \bot \mid n \geq 0\}$ is a non-empty chain and, thus, $\bigsqcup\{f^n \perp \mid n \geq 0\}$ exists (i.e., defines an element of D). We next show that it is a fixed point of f:

$$\begin{aligned} f(\bigsqcup\{f^n \perp \mid n \geq 0\}) &= \bigsqcup\{f(f^n) \perp \mid n \geq 0\} \\ &= \bigsqcup\{f^n \perp \mid n \geq 1\} \\ &= \bigsqcup(\{f^n \perp \mid n \geq 1\} \cup \{\perp\}) \\ &= \bigsqcup\{f^n \perp \mid n \geq 0\} \end{aligned}$$

It remains to be shown that $\mathsf{FIX}\ f$ is the least fixed point of f. For an arbitrary fixed point d of f, we have that $f\,d = d$ and, clearly, $\perp \preccurlyeq d$. By monotonicity of f and an induction on n, we get $f^n \perp \preccurlyeq f^n\,d = d$ for all $n \geq 0$. Hence, d is an upper bound for the chain $\{f^n \perp \mid n \geq 0\}$ and since $\mathsf{FIX}\ f$ is the least upper bound of that chain, we directly obtain $\mathsf{FIX}\ f \preccurlyeq d$. ■

The Direct Style Semantics in One Slide

- $\mathcal{S}_{\mathsf{ds}}[\![x \texttt{ := } a]\!]\, s := s[x \mapsto \mathcal{A}[\![a]\!]\, s]$
- $\mathcal{S}_{\mathsf{ds}}[\![\texttt{skip}]\!] := \mathrm{id}$
- $\mathcal{S}_{\mathsf{ds}}[\![S_1 \texttt{ ; } S_2]\!] := \mathcal{S}_{\mathsf{ds}}[\![S_1]\!] \circ \mathcal{S}_{\mathsf{ds}}[\![S_1]\!]$
- $\mathcal{S}_{\mathsf{ds}}[\![\texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2]\!] := \mathsf{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{\mathsf{ds}}[\![S_1]\!], \mathcal{S}_{\mathsf{ds}}[\![S_2]\!])$
- $\mathcal{S}_{\mathsf{ds}}[\![\texttt{while } b \texttt{ do } S]\!]\, s = \mathsf{FIX}\ F$

where $F = \mathsf{cond}(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{\mathsf{ds}}[\![S]\!], \mathrm{id})$

The Direct Style Semantics Exists

Theorem 2: $\mathcal{S}_{\mathsf{ds}}[\![\cdot]\!]$ is a total function.

Proof: We need to show that for all While programs S, $\mathcal{S}_{\mathsf{ds}}[\![S]\!]$ yields a partial function $g : \mathbf{State} \hookrightarrow \mathbf{State}$. Therefore note that, since all states $\mathrm{Var} \to \mathbb{Z}$ are total functions, also $\mathcal{B}[\![b]\!]$ and $\mathcal{A}[\![a]\!]$ are total for any Boolean expression b and arithmetic expression a. The proof follows a structural induction on S.

Base Cases For $S = x \equiv a$ and $S =$`skip`, $\mathcal{S}_{\mathsf{ds}}[\![S]\!]$ is certainly total.

Step Since $\mathcal{S}_{\mathsf{ds}}[\![S_1]\!]$ and $\mathcal{S}_{\mathsf{ds}}[\![S_2]\!]$ are total functions (by induction hypothesis), $\mathcal{S}_{\mathsf{ds}}[\![S_2]\!] \circ \mathcal{S}_{\mathsf{ds}}[\![S_1]\!]$ yields a total function as well, meaning $\mathcal{S}_{\mathsf{ds}}[\![S_1\,;S_2]\!]$ is total.

Function cond is total as well because of the induction hypothesis and the fact that $\mathcal{B}[\![b]\!]$ is a total function.

For the last case, assume F is continuous (a proof we deliver in Lemma 3). Then $\mathsf{FIX}\ F$ yields a unique partial function by Theorem 1 and, therefore, $\mathcal{S}_{\mathsf{ds}}[\![\texttt{while}\ b\ \texttt{do}\ S']\!]$ yields a partial function.

Thus, $\mathcal{S}_{\mathsf{ds}}[\![\cdot]\!]$ is total and, therefore, exists. ∎

Lemma 3: Functional F, as used in the definition of $\mathcal{S}_{\mathsf{ds}}[\![\texttt{while}\ b\ \texttt{do}\ S]\!]$, is continuous.

Proof: We first show that functionals F_1 with

$$F_1\ g = \mathsf{cond}(p, g, \mathrm{id}\,)$$

where g : **State** $\hookrightarrow$ **State** and p : **State** $\to \mathbb{B}$, are continuous. Lut us start by showing that F_1 is monotone. Let $g_1 \sqsubseteq g_2$ and s an arbitrary state. We need to show that $(F_1\ g_1)s = s'$ implies (by assumption) $(F_2\ g_2)s = s'$. If $p\ s = \texttt{tt}$, then $s' = (F_1\ g_1)s = g_1\ s$ implies $s' = g_2\ s = (F_1\ g_2)s$.

Let Y be a non-empty chain of **State** $\hookrightarrow$ **State**. By monotonicity of F_1, we get

$$\bigsqcup\{F_1\ d \mid d \in Y\} \sqsubseteq F_1(\bigsqcup Y)$$

Let s be a state such that $F_1(\bigsqcup Y)s = s'$. If $p\,s = \mathtt{ff}$, then $F_1(\bigsqcup Y)s = \mathrm{id}\,s = s'$ and, surely, $(F_1\,g)s = \mathrm{id}\,s = s'$ for all $g \in Y$. If $p\,s = \mathtt{tt}$, then $(\bigsqcup Y)s = s'$ (since $(F(\bigsqcup Y))s = (\bigsqcup Y)s$) we need to show that there is a $g \in Y$ such that $g\,s = s'$. Note, $g\,s$ is the same for all $g \in Y$ defined for s. Suppose, $g\,s = \mathtt{undef}$ for all $g \in Y$. Then certainly $(\bigsqcup Y)[s \mapsto \mathtt{undef}]$ is an upper bound of Y. But $(\bigsqcup Y)$ being already the least upper bound of Y entails a contradiction. Thus, there is a $g \in Y$ with $g\,s = s'$ and, thus, $\bigsqcup\{(F_1\,g) \mid g \in Y\}s = s'$.

Next, we show that functionals F_2 with

$$F_2\,g = g \circ g_0$$

where $g_0 : \mathbf{State} \hookrightarrow \mathbf{State}$, are continuous. Again, we start with monotonicity: Let $g_1 \sqsubseteq g_2$ and we need to show that $F_2\,g_1 \sqsubseteq F_2\,g_2$. But this is immediate from the fact that $F_2\,g_i = g_i \circ g_0$, so if $g_0\ s = s_1$, then $g_1\ s_1 = s'$ implies $g_2\ s_1 = s'$.

Continuity of While-Functionals

Let Y be a non-empty chain over **State** $\hookrightarrow$ **State**. We get $\bigsqcup\{F_2\, g \mid g \in Y\} \sqsubseteq F_2(\bigsqcup Y)$ by monotonicity of F_2. For state s, we get $(F_2(\bigsqcup Y))s = ((\bigsqcup Y) \circ g_0)s = (\bigsqcup Y)(g_0\, s) = s'$ we obtain there must be a $g \in Y$ such that $g(g_0\, s) = s'$. Hence, $F_2(\bigsqcup Y) \sqsubseteq \bigsqcup\{F_2\, g \mid g \in Y\}$.

Then $F_2 \circ F_1$ is continuous as well, making $\mathcal{S}_{\mathsf{ds}}[\![\cdot]\!]$ well-defined for while-loops. ■

Operational Semantics

Was bisher geschah:

$$a ::= n \mid x \mid a \oplus a \mid a \star a \mid a \ominus a$$

$$b ::= \texttt{true} \mid \texttt{false} \mid a \equiv a \mid a \leqq a \mid \neg b \mid b \wedge b$$

$$S ::= x \texttt{ := } a \mid \texttt{skip} \mid S \texttt{ ; } S \mid \texttt{if } b \texttt{ then } S \texttt{ else } S \mid \texttt{while } b \texttt{ do } S$$

where $n \in \mathbf{Num}$ and $x \in \mathbf{Var}$.

- functions describe the effect compositionally: $\mathcal{S}_{\mathsf{ds}}[\![\cdot]\!]$ relates inputs with outputs
- does this semantics tell us why/how a program computes what it computes?

The Operational Approach

- describe the semantics in terms of *transitions* that perform the actual state change
- we consider two different styles:

  natural semantics $\rightarrow$ relates program-state pairs with states;
  every natural step comes with a proof;
  sometimes referred to as *big step semantics*

  structural operational semantics $\Rightarrow$ relates program-state pairs with program-state pairs or just states;
  also known as *small step semantics*
- both styles are formalized by a finite set of rules

$$[\text{axiom}]\,\frac{\text{empty premise}}{\text{conclusion}} \qquad\qquad [\text{rule}]\,\frac{\text{premise}}{\text{conclusion}}\;\text{if} \ldots \text{condition}$$

- key principle: rule induction

Rule Induction by Examples

1. Lists over alphabet Σ

$$[\text{nil}]\frac{}{nil \in \mathcal{L}} \qquad [\text{cons}]\frac{s \in \mathcal{L} \quad a \in \Sigma}{\langle a \rangle \bullet s \in \mathcal{L}}$$

$\mathcal{L}$ is the **smallest set** satisfying rule [nil] and [cons].

2. Finite Trace Process Pr. Let $T = (Q, \Sigma, \rightarrow)$ be an LTS.

$$[\text{dead}]\frac{\forall \mu \in \Sigma : P \overset{\mu}{\nrightarrow}}{P\downharpoonleft} \qquad [\text{trans}]\frac{P \overset{\mu}{\rightarrow} P' \quad P'\downharpoonleft}{P\downharpoonleft}$$

The set of finite trace processes is the **smallest set** $\downharpoonleft$ satisfying rules [dead] and [trans].

Smells Like Fixed Points

$$\frac{}{x \texttt{:=} a \in \mathbf{WHILE}} \text{ if } x \in \mathbf{Var} \text{ and } a \in \mathbf{Aexp} \qquad \frac{}{\texttt{skip} \in \mathbf{WHILE}}$$

$$\frac{S_1 \in \mathbf{WHILE} \quad S_2 \in \mathbf{WHILE}}{S_1 \texttt{;} S_2 \in \mathbf{WHILE}} \qquad \frac{b \in \mathbf{Bexp} \quad S_1 \in \mathbf{WHILE} \quad S_2 \in \mathbf{WHILE}}{\texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \in \mathbf{WHILE}}$$

$$\frac{b \in \mathbf{Bexp} \quad S \in \mathbf{WHILE}}{\texttt{while}\ b\ \texttt{do}\ S \in \mathbf{WHILE}}$$

The language of all **WHILE** statements is the **smallest set** satisfying the rules above.

Natural Semantic Rules

- assignments and skip statements form the induction base
- as in $\mathcal{S}_{\mathsf{ds}}[\![\cdot]\!]$, assignments alter the state while skip leaves it identical

$$[\mathrm{ass}_{\mathrm{ns}}]\,\frac{}{\langle x\,\texttt{:=}\,a,\, s\rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]\, s]} \qquad [\mathrm{skip}_{\mathrm{ns}}]\,\frac{}{\langle \texttt{skip},\, s\rangle \Rightarrow s}$$

- we want to prove that the sequential composition $S_1\,\texttt{;}\,S_2$, initiated in state s, yields s'
- then we need to show that there is a state s'', such that statement S_1 in s yields s'' and statement S_2 in s'' finally yields s'

$$[\mathrm{seq}_{\mathrm{ns}}]\,\frac{\langle S_1,\, s\rangle \Rightarrow s'' \quad \langle S_2,\, s''\rangle \Rightarrow s'}{\langle S_1\,\texttt{;}\,S_2,\, s\rangle \Rightarrow s'}$$

- for conditionals, the proof depends on the evaluation of the branching condition

$$[\text{if}_{\text{ns}}^{\texttt{tt}}]\ \frac{\langle S_1, s\rangle \rightarrow s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \rightarrow s'}\ \text{ if } \mathcal{B}[\![b]\!]\ s = \texttt{tt}$$

$$[\text{if}_{\text{ns}}^{\texttt{ff}}]\ \frac{\langle S_2, s\rangle \rightarrow s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \rightarrow s'}\ \text{ if } \mathcal{B}[\![b]\!]\ s = \texttt{ff}$$

- also for while-loops, we distinguish alongside the cases of the loop condition
- here, the proof unravels the computation by one iteration

Natural Semantic Rules

$$[\text{while}_{\text{ns}}^{\texttt{tt}}]\ \frac{\langle S, s\rangle \rightarrow s' \quad \langle \texttt{while}\ b\ \texttt{do}\ S, s'\rangle \rightarrow s''}{\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \rightarrow s''}\ \text{ if } \mathcal{B}[\![b]\!]\, s = \texttt{tt}$$

$$[\text{while}_{\text{ns}}^{\texttt{ff}}]\ \frac{}{\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \rightarrow s}\ \text{ if } \mathcal{B}[\![b]\!]\, s = \texttt{ff}$$

An Example: $y = x!$

Consider the statement

$$y \texttt{ := } 1\texttt{; while } \neg(x \equiv 1) \texttt{ do } (y \texttt{ := } y \star x\texttt{; } x \texttt{ := } x \ominus 1)$$

in state s with $s\,x = 3$. We use the semantic rules to show that the statement in state s yields $s[x \mapsto 1][y \mapsto 6]$.

Therefore, note that on any state s, $\langle (y \texttt{ := } y \star x\texttt{; } x \texttt{ := } x \ominus 1)\,, s\rangle \rightarrow s[x \mapsto s\,x - 1][y \mapsto s\,y \cdot s\,x]$

$$[\mathrm{seq_{ns}}]\dfrac{[\mathrm{ass_{ns}}]\dfrac{}{\langle y \texttt{:=} y \star x,\, s\rangle \rightarrow s[y \mapsto \mathcal{A}[\![y \star x]\!]\, s]} \quad [\mathrm{ass_{ns}}]\dfrac{}{\langle x \texttt{:=} x \ominus 1,\, s\rangle \rightarrow s[x \mapsto \mathcal{A}[\![x \ominus 1]\!]\, s]}}{\langle y \texttt{ := } y \star x\texttt{; } x \texttt{ := } x \ominus 1,\, s\rangle \rightarrow s[y \mapsto s\,y \cdot s\,x][x \mapsto s\,x - 1]}$$

We subsequently abbreviate $(y \texttt{ := } y \star x\texttt{; } x \texttt{ := } x \ominus 1)$ by $S^\star$ and we abbreviate the proof tree above by $[S^\star]$.

$$[\text{seq}_{\text{ns}}]\ \dfrac{[\text{ass}_{\text{ns}}]\ \dfrac{}{\langle y\texttt{:=1},\, s\rangle \Rightarrow s[y \mapsto \mathcal{A}[\![\texttt{1}]\!]\, s]} \qquad [\text{while}^{\texttt{tt}}_{\text{ns}}]\ \dfrac{}{\langle \texttt{while}\ \neg(x \equiv 1)\ \texttt{do}\ S^\star, s[y \mapsto 1]\rangle \Rightarrow s[x \mapsto 1][y \mapsto 6]}}{\langle y\ \texttt{:=}\ \texttt{1;}\ \texttt{while}\ \neg(x \equiv 1)\ \texttt{do}\ S^\star, s\rangle \Rightarrow s[x \mapsto 1][y \mapsto 6]}$$

An Example: $y = x!$

$$
[\text{while}^{\texttt{tt}}_{\text{ns}}]\,\dfrac{[S^\star]\,\dfrac{}{\langle S^\star, s[y \mapsto 1]\rangle \rightarrow s[x \mapsto 2][y \mapsto 3] = s'} \qquad [\text{while}^{\texttt{tt}}_{\text{ns}}]\,\dfrac{}{\langle \texttt{while}\ \neg(x \equiv 1)\ \texttt{do}\ S^\star, s'\rangle \rightarrow s''}}{\langle \texttt{while}\ \neg(x \equiv 1)\ \texttt{do}\ S^\star, s[y \mapsto 1]\rangle \rightarrow s[x \mapsto 1][y \mapsto 6] = s''}
$$

An Example: $y = x!$

$$
[\text{while}^{\texttt{tt}}_{\text{ns}}]\,\dfrac{[S^\star]\,\dfrac{}{\langle S^\star, s[x \mapsto 2][y \mapsto 3]\rangle \to s[x \mapsto 1][y \mapsto 6] = s'} \qquad [\text{while}^{\texttt{ff}}_{\text{ns}}]\,\dfrac{}{\langle \texttt{while}\ \neg(x \equiv 1)\ \texttt{do}\ S^\star, s'\rangle \to s''}}{\langle \texttt{while}\ \neg(x \equiv 1)\ \texttt{do}\ S^\star, s[y \mapsto 1]\rangle \to s[x \mapsto 1][y \mapsto 6] = s''}
$$

$$[\text{ass}_{\text{ns}}]\ \frac{}{\langle x \texttt{:=} a,\ s\rangle \rightarrow s[x \mapsto \mathcal{A}[\![a]\!]\ s]} \qquad [\text{skip}_{\text{ns}}]\ \frac{}{\langle \texttt{skip},\ s\rangle \rightarrow s} \qquad [\text{seq}_{\text{ns}}]\ \frac{\langle S_1,\ s\rangle \rightarrow s'' \quad \langle S_2,\ s''\rangle \rightarrow s'}{\langle S_1 \texttt{;} S_2,\ s\rangle \rightarrow s'}$$

$$[\text{if}^{\texttt{tt}}_{\text{ns}}]\ \frac{\langle S_1,\ s\rangle \rightarrow s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \rightarrow s'} \quad \text{if } \mathcal{B}[\![b]\!]\ s = \texttt{tt}$$

$$[\text{if}^{\texttt{ff}}_{\text{ns}}]\ \frac{\langle S_2,\ s\rangle \rightarrow s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \rightarrow s'} \quad \text{if } \mathcal{B}[\![b]\!]\ s = \texttt{ff}$$

$$[\text{while}^{\texttt{tt}}_{\text{ns}}]\ \frac{\langle S,\ s\rangle \rightarrow s' \quad \langle \texttt{while}\ b\ \texttt{do}\ S,\ s'\rangle \rightarrow s''}{\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \rightarrow s''} \quad \text{if } \mathcal{B}[\![b]\!]\ s = \texttt{tt}$$

$$[\text{while}^{\texttt{ff}}_{\text{ns}}]\ \frac{}{\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \rightarrow s''} \quad \text{if } \mathcal{B}[\![b]\!]\ s = \texttt{ff}$$

Theorem 4: The natural semantics is deterministic.

Proof: **Exercise** ∎

Theorem 5: The semantic function of the natural semantics $\mathcal{S}_{\mathsf{ns}}[\![\cdot]\!] : \mathbf{Stm} \to (\mathbf{State} \hookrightarrow \mathbf{State})$ given by

$$\mathcal{S}_{\mathsf{ns}}[\![S]\!]\, s = \begin{cases} s' & \text{if } \langle S, s\rangle \twoheadrightarrow s' \\ \texttt{undef} & \text{otherwise} \end{cases}$$

exists (and is well-defined).

An Equivalence Result

Theorem 6: The natural semantics and the direct style semantics coincide, that is

$$\mathcal{S}_{\mathsf{ds}}[\![S]\!] = \mathcal{S}_{\mathsf{ns}}[\![S]\!]$$

for all statements S of the While-language.

Proof: **Exercise** ■

The Natural Semantics = Big Step Semantics

- as for $\mathcal{S}_{\text{ds}}[\![\cdot]\!]$, the transition rules provide us with proofs relating inputs with outputs of program execution
- a more fine-grained approach is taken by the *structural operational semantics*
- as the name states, this semantics defines the operational behavior (i.e., the transitions) in terms of the program structure
- small step transitions have the following shape: $\langle S, s\rangle \Rightarrow \gamma$
  - γ can be of the form $\langle S', s'\rangle$
  - γ can be of the form s' (in case of termination)

Rules of the Structural Operational Semantics (SOS)

$$[\text{ass}_{\text{SOS}}]\ \frac{}{\langle x \texttt{:=} a,\, s\rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]\, s]}$$

$$[\text{skip}_{\text{SOS}}]\ \frac{}{\langle \texttt{skip},\, s\rangle \Rightarrow s}$$

$$[\text{seq}^1_{\text{SOS}}]\ \frac{\langle S_1,\, s\rangle \Rightarrow \langle S_1',\, s'\rangle}{\langle S_1 \texttt{;} S_2,\, s\rangle \Rightarrow \langle S_1' \texttt{;} S_2,\, s'\rangle}$$

$$[\text{seq}^2_{\text{SOS}}]\ \frac{\langle S_1,\, s\rangle \Rightarrow s'}{\langle S_1 \texttt{;} S_2,\, s\rangle \Rightarrow \langle S_2,\, s'\rangle}$$

$$[\text{if}^{\texttt{tt}}_{\text{SOS}}]\ \frac{}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\, s\rangle \Rightarrow \langle S_1,\, s\rangle}\quad \text{if } \mathcal{B}[\![b]\!]\, s = \texttt{tt}$$

$$[\text{if}^{\texttt{ff}}_{\text{SOS}}]\ \frac{}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\, s\rangle \Rightarrow \langle S_2,\, s\rangle}\quad \text{if } \mathcal{B}[\![b]\!]\, s = \texttt{ff}$$

$$[\text{while}_{\text{SOS}}]\ \frac{}{\langle \texttt{while}\ b\ \texttt{do}\ S,\, s\rangle \Rightarrow \langle \texttt{of}\ b\ \texttt{then}\ S\ \texttt{else}\ \texttt{skip},\, s\rangle}$$

The Semantic Function of SOS

Theorem 7: The structural operational semantics is deterministic.

$$\mathcal{S}_{\mathsf{sos}}[\![S]\!]\, s = \begin{cases} s' & \text{if } \langle S, s \rangle \Rightarrow s' \\ \texttt{undef} & \text{otherwise} \end{cases}$$

Theorem 8: For all statements S, $\mathcal{S}_{\mathsf{ns}}[\![S]\!] = \mathcal{S}_{\mathsf{sos}}[\![S]\!]$.

Direct Consequence: All three semantics are equivalent.

We learned about three different yet equivalent styles of (sequential) programming language semantics:

denotational semantics computation = function application

natural semantics computation = step-by-step proofs (derivation tree)

structural operational semantics computation = step-by-step computation (??)

Next:

- the Calculus of Communicating Systems (CCS)
- which semantic style to choose for CCS?
- an old friend around the corner: bisimilarity is a congruence
- the untold story of Matthew Hennessy and Robin Milner
- justifying bisimilarity