19. Denotational Semantics

Goal: · If (deterministic) program c computes a partial function

$$C[\![c]\!] : State \hookrightarrow State.$$

· Denotational semantics study the partial function.
In particular, they try to capture the function
without making reference to the syntax
in the definition of the semantics.

· This suggests that we should not be interested in programs
but in the partial functions that they realize, or equivalently,
in classes of programs obtained by factorizing along functional equivalence.

Definition (Functional equivalence):

$c_1 \sim c_2$, if $\{(\sigma,\sigma') \mid (c_1,\sigma) \Downarrow \sigma'\} = \{(\sigma,\sigma') \mid (c_2,\sigma) \Downarrow \sigma'\}$.

· We already know this:
We are not interested in having an NFA A or B
as long as $L(A) = L(B)$.

History: The pioneers here were Christopher Strachey and Dana Scott.

Definition (Denotational semantics of simple commands):

The denotational semantics $C : Prog \to (State \hookrightarrow State)$
assigns to each program $c \in Prog$ a partial function $C[\![c]\!] : State \hookrightarrow State$
as follows:

$$C[\![skip]\!] := \{(\sigma,\sigma) \mid \sigma \in State\}$$

$$C[\![x := a]\!] := \{(\sigma, \sigma[x := S[\![a]\!](\sigma)]) \mid \sigma \in State\}$$

$$C[\![c_1; c_2]\!] := C[\![c_2]\!] \circ C[\![c_1]\!]$$

$$C[\![\text{if } b \text{ then } c_1 \text{ else } c_2 \text{ fi}]\!] := \{(\sigma,\sigma') \mid S[\![b]\!](\sigma) = true \text{ and } (\sigma,\sigma') \in C[\![c_1]\!]\}$$
$$\cup \{(\sigma,\sigma') \mid S[\![b]\!](\sigma) = false \text{ and } (\sigma,\sigma') \in C[\![c_2]\!]\}.$$

19.1 Denotational Semantics of While-Programs:

Let $w := \underline{\text{while}}\ b\ \underline{\text{do}}\ c\ \underline{\text{od}}$.

We have the small-step transition

$$(w, \sigma) \to (\underline{\text{if}}\ b\ \underline{\text{then}}\ c;w\ \underline{\text{else}}\ \text{skip}\ \underline{\text{fi}}, \sigma) \quad \text{for all } \sigma \in \text{State}.$$

Hence, the state reached upon termination is the same,

which means $\quad w \sim \underline{\text{if}}\ b\ \underline{\text{then}}\ c;w\ \underline{\text{else}}\ \text{skip}\ \underline{\text{fi}}$

So to define the denotational semantics for while-programs
in accordance with the small-step and the big-step operational semantics,
we should require

$$\begin{aligned}
&C[\![w]\!] \\
(\text{Requirement}) &= C[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ c;w\ \underline{\text{else}}\ \text{skip}\ \underline{\text{fi}}]\!] \\
(\text{Def. } C[\![-]\!] \text{ for if}) &= \{(\sigma,\sigma') \mid \mathcal{B}[\![b]\!](\sigma) = \text{true and } (\sigma,\sigma') \in C[\![c;w]\!]\} \\
&\quad \cup \{(\sigma,\sigma) \mid \mathcal{B}[\![b]\!](\sigma) = \text{false}\}.
\end{aligned}$$

If we apply the definition of $C[\![-]\!]$ for ;,
we obtain the equation:

$$(*) \quad \begin{aligned} C[\![w]\!] = &\{(\sigma,\sigma') \mid \mathcal{B}[\![b]\!](\sigma) = \text{true and } (\sigma,\sigma') \in C[\![w]\!] \circ C[\![c]\!]\} \\ &\cup \{(\sigma,\sigma) \mid \mathcal{B}[\![b]\!](\sigma) = \text{false}\}. \end{aligned}$$

Equation (*) is <u>recursive</u>, the function $C[\![w]\!]$ that we try to define
occurs on the left- <u>and</u> on the right-hand side.
Hence, we need to compute it as a fixed point.
To this end, we understand the right-hand side of Equation (*)
as a function $\Gamma_{b,c}$ on partial functions.

<u>Definition</u> ($\Gamma_{b,c} : (\text{State} \hookrightarrow \text{State}) \to (\text{State} \hookrightarrow \text{State})$):

$$\begin{aligned} \Gamma_{b,c}(f) := &\{(\sigma,\sigma') \mid \mathcal{B}[\![b]\!](\sigma) = \text{true and } (\sigma,\sigma') \in f \circ C[\![c]\!]\} \\ &\cup \{(\sigma,\sigma) \mid \mathcal{B}[\![b]\!](\sigma) = \text{false}\}. \end{aligned}$$

With this definition of $T_{b,c}$, Equation (*) becomes

$$C[\![w]\!] = T_{b,c}(C[\![w]\!]),$$

so the $C[\![w]\!]$ of interest is a fixed point of $T_{b,c}$.
We take the least fixed point as the denotation of while-programs.
We will show in a moment that it is guaranteed to exist.

Definition (Denotational semantics for while-programs):

$$C[\![\text{while } b \text{ do } c \text{ od}]\!] := lfp(T_{b,c}).$$

Note:
- The denotational semantics is truly compositional in the sense that the denotation of a composed program is constructed from the denotations of the subprograms.
- This does not hold for the big-steps semantics: For while programs, the program reappears in the premise of the rule.

To establish the existence of the least fixed point, we appeal to a variant of Kleene's fixed point theorem. It needs the following lemma.

Lemma:
(1) The set of partial functions ordered by inclusion, $(State \hookrightarrow State, \subseteq)$,
  (a) forms a partial order,
  (b) with least element $\emptyset$, the function that is undefined everywhere, and
  (c) where every ascending chain $f_0 \subseteq f_1 \subseteq \ldots$ contains a join $\bigcup_{i \in \mathbb{N}} f_i$.
(2) $T_{b,c} : (State \hookrightarrow State) \to (State \hookrightarrow State)$ is $\sqcup$-continuous, and therefore monotone.

Proof:
We show (2).

Let $K = h_1 \subseteq h_2 \subseteq \ldots$ be a chain in $State \to\!\!\!\!\to State$.

→ For every $i \in \mathbb{N}$, we have

$$\begin{aligned} T_{b,c}(h_i) &= \{(\sigma,\sigma') \mid \mathcal{S}[\![b]\!](\sigma) = true \text{ and } (\sigma,\sigma') \in h_i \circ C[\![c]\!]\} \\ &\quad \cup \{(\sigma,\sigma) \mid \mathcal{S}[\![b]\!](\sigma) = false\} \\ &\subseteq \{(\sigma,\sigma') \mid \mathcal{S}[\![b]\!](\sigma) = true \text{ and } (\sigma,\sigma') \in \left(\bigcup_{i\in\mathbb{N}} h_i\right) \circ C[\![c]\!]\} \\ &\quad \cup \{(\sigma,\sigma) \mid \mathcal{S}[\![b]\!](\sigma) = false\} \\ &= T_{b,c}\left(\bigcup_{i\in\mathbb{N}} h_i\right). \end{aligned}$$

Since the inclusion holds for every element, $T_{b,c}(\bigcup_{i\in\mathbb{N}} h_i)$ is an upper bound of $\{T_{b,c}(h_i) \mid i \in \mathbb{N}\}$.

Hence, $\bigcup_{i\in\mathbb{N}} T_{b,c}(h_i) \subseteq T_{b,c}(\bigcup_{i\in\mathbb{N}} h_i)$.

→ For the reverse inclusion, consider

$$\begin{aligned} (\sigma,\sigma') \in T_{b,c}\left(\bigcup_{i\in\mathbb{N}} h_i\right) = \{(\sigma,\sigma') \mid \mathcal{S}[\![b]\!](\sigma) = true \text{ and} \\ (\sigma,\sigma') \in \left(\bigcup_{i\in\mathbb{N}} h_i\right) \circ C[\![c]\!]\} \\ \cup \{(\sigma,\sigma) \mid \mathcal{S}[\![b]\!](\sigma) = false\}. \end{aligned}$$

- If $(\sigma,\sigma') \in \{(\sigma,\sigma) \mid \mathcal{S}[\![b]\!](\sigma) = false\}$,
then $(\sigma,\sigma') \in T_{b,c}(h_i)$ for all $i \in \mathbb{N}$.
- If $(\sigma,\sigma') \in \{(\sigma,\sigma') \mid \mathcal{S}[\![b]\!](\sigma) = true \text{ and } (\sigma,\sigma') \in \left(\bigcup_{i\in\mathbb{N}} h_i\right) \circ C[\![c]\!]\}$,
then here is $\sigma'' \in State$ with $(\sigma,\sigma'') \in C[\![c]\!]$ and $(\sigma'',\sigma') \in \bigcup_{i\in\mathbb{N}} h_i$.
Since $(\sigma'',\sigma') \in \bigcup_{i\in\mathbb{N}} h_i$, there is $i_0 \in \mathbb{N}$ with $(\sigma'',\sigma') \in h_{i_0}$.
Otherwise, the infinite union would not contain the pair.
(Indeed, since $h_1 \subseteq h_2 \subseteq \ldots$ we have $(\sigma'',\sigma') \in h_j$ for all $j \geq i_0$.)
From $(\sigma'',\sigma') \in h_{i_0}$ (and $(\sigma,\sigma'') \in C[\![c]\!]$ and $\mathcal{S}[\![b]\!](\sigma) = true$),
we can conclude

$$\begin{aligned} (\sigma,\sigma') \in \{(\sigma,\sigma') \mid \mathcal{S}[\![b]\!] = true \text{ and } (\sigma,\sigma') \in h_{i_0} \circ C[\![c]\!]\} \subseteq T_{b,c}(h_{i_0}) \\ \subseteq \bigcup_{i\in\mathbb{N}} T_{b,c}(h_i). \end{aligned}$$

To see that $\sqcup$-continuity implies monotonicity,
consider $h_1 \subseteq h_2$ with $h_1, h_2 : State \to State$.

We have

$$T_{b,c}(h_1) \subseteq T_{b,c}(h_1) \cup T_{b,c}(h_2)$$

$$\overset{(\sqcup\text{-continuity})}{\subseteq} T_{b,c}(h_1 \cup h_2)$$

$$\overset{(h_1 \subseteq h_2)}{=} T_{b,c}(h_2).$$

□

If a partial order satisfies Requirement (c) in the lemma,
it is said to be a complete partial order.
If it satisfies Requirement (b), it is said to have a $\perp$ element.
The variant of Kleene's theorem that we will refer to is given next.
The proof is identical with the proof we had previously.

Theorem (Kleene):
Let $f : D \to D$ be a $\sqcup$-continuous function
on a complete partial order $(D, \leq)$ that has a $\perp$ element.
Then $$lfp(f) = \bigsqcup_{i \in \mathbb{N}} f^i(\perp).$$

Combined with the above lemma, we obtain:

Lemma:
$lfp(T_{b,c})$ is guaranteed to exist.

19.2 Correspondence of Denotational and Operational Semantics

Goal: Show that the big-step operational semantics
and the denotational semantics coincide.

We start with a lemma on while-programs that will be helpful.

Lemma:
Let $w = \underline{\text{while}}\ b\ \underline{\text{do}}\ c\ \underline{\text{od}}$. Then $C[\![w]\!] = C[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ c;w\ \underline{\text{else}}\ \text{skip}\ \underline{\text{fi}}]\!]$.

Proof:

$$
\begin{aligned}
& C[\![w]\!] \\
(\text{Def. } C[\![-]\!]) &= \mathit{lfp}(T_{b,c}) \\
(\text{Fixed point}) &= T_{b,c}(C[\![w]\!]) \\
(\text{Def. } T_{b,c}) &= \{(\sigma,\sigma') \mid S[\![b]\!](\sigma) = \text{true and } (\sigma,\sigma') \in C[\![w]\!] \circ C[\![c]\!]\} \\
&\quad \cup \{(\sigma,\sigma) \mid S[\![b]\!](\sigma) = \text{false}\} \\
(\text{Def. } C[\![-]\!]) &= \{(\sigma,\sigma') \mid S[\![b]\!](\sigma) = \text{true and } (\sigma,\sigma') \in C[\![c;w]\!]\} \\
&\quad \cup \{(\sigma,\sigma) \mid S[\![b]\!](\sigma) = \text{false}\} \\
(\text{Def. } C[\![-]\!]) &= C[\![\underline{\text{if}}\ b\ \underline{\text{then}}\ c;w\ \underline{\text{else}}\ \text{skip}\ \underline{\text{fi}}]\!].
\end{aligned}
$$

□

We are now prepared to show the correspondence.

Theorem (Correspondence of big-step operational and denotational semantics):
$(c,\sigma) \Downarrow \sigma'$ iff $(\sigma,\sigma') \in C[\![c]\!]$.

Proof:
$\Rightarrow$ We proceed by Noetherian induction on the height of the derivation tree and show the implication for all $c \in \text{Prog}$ and all $\sigma,\sigma' \in \text{State}$.
We discuss the complicated cases in more detail and skip the routine ones.

Base case: $(\text{skip},\sigma) \Downarrow \sigma$ and $(x := a,\sigma) \Downarrow \sigma[x := S[\![a]\!](\sigma)]$ are both simple.

Induction step: Assume the implication holds for all derivations of height $\leq n$, and consider a derivation of height $n+1$.
We do a case distinction along the possible last derivation steps.

(bs_iftrue) Consider $\dfrac{(c_0,\sigma)\Downarrow\sigma'}{(\text{if } b \text{ then } c_0 \text{ else } c_1 \text{ fi},\sigma)\Downarrow\sigma'}$, $S[\![b]\!](\sigma)=\text{true}$.

Since the height of the derivation for $(c_0,\sigma)\Downarrow\sigma'$ is at most n, we have

$(\sigma,\sigma')\in C[\![c_0]\!]$ by the induction hypothesis.

Hence,

$(\sigma,\sigma')\in\{(\sigma,\sigma') \mid S[\![b]\!](\sigma)=\text{true and } (\sigma,\sigma')\in C[\![c_0]\!]\}$

$\subseteq C[\![\text{if } b \text{ then } c_0 \text{ else } c_1 \text{ fi}]\!]$.

(bs_iffalse) Similar.

(bs_seq) $\dfrac{(c_0,\sigma)\Downarrow\sigma' \quad (c_1,\sigma')\Downarrow\sigma''}{(c_0;c_1,\sigma)\Downarrow\sigma''}$.

By the induction hypothesis, we have

$(\sigma,\sigma')\in C[\![c_0]\!]$ and $(\sigma',\sigma'')\in C[\![c_1]\!]$.

Hence, by definition of relational composition:

$(\sigma,\sigma'')\in C[\![c_1]\!]\circ C[\![c_0]\!] = C[\![c_0;c_1]\!]$.

(bs_while_false) $\dfrac{}{(\text{while } b \text{ do } c \text{ od},\sigma)\Downarrow\sigma}$, $S[\![b]\!](\sigma)=\text{false}$.

We have $C[\![\text{while } b \text{ do } c \text{ od}]\!] = \text{lfp}(T_{b,c})$.

Hence, $C[\![\text{while } b \text{ do } c \text{ od}]\!] = T_{b,c}(C[\![\text{while } b \text{ do } c \text{ od}]\!])$

$\supseteq \{(\sigma,\sigma) \mid S[\![b]\!](\sigma)=\text{false}\}$.

The given pair (σ,σ) is in the latter set.

(bs_while_true) $\dfrac{(c,\sigma)\Downarrow\sigma' \quad (\text{while } b \text{ do } c \text{ od},\sigma')\Downarrow\sigma''}{(\text{while } b \text{ do } c \text{ od},\sigma)\Downarrow\sigma''}$, $S[\![b]\!](\sigma)=\text{true}$.

By the induction hypothesis,

$(\sigma, \sigma') \in C[\![c]\!]$ and $(\sigma', \sigma'') \in C[\![w]\!]$

with $w :=$ *while* b *do* c *od*.

Hence

$(\sigma, \sigma'') \in C[\![w]\!] \circ C[\![c]\!] = C[\![c; w]\!]$.

Together with $\mathcal{B}[\![b]\!](\sigma) = true$, we arrive at

$(\sigma, \sigma') \in C[\![$ *if* b *then* $c; w$ *else skip fi* $]\!]$

$= C[\![$ *while* b *do* c *od* $]\!]$.

(Lemma above)

"$\subseteq$" We proceed by induction on the structure of programs and again concentrate on the more complicated cases.

Base case: For skip and $x := a$, the reasoning is simple.

Induction step: Assume the implication holds for c, c_1, and c_2.

- Consider

  $(\sigma, \sigma') \in C[\![c_1; c_2]\!] = C[\![c_2]\!] \circ C[\![c_1]\!]$.

  By definition of relational composition $\circ$,

  there is $\sigma'' \in$ State with $(\sigma, \sigma'') \in C[\![c_1]\!]$ and $(\sigma'', \sigma') \in C[\![c_2]\!]$.

  By the induction hypothesis, we get

  $(c_1, \sigma) \Downarrow \sigma''$ and $(c_2, \sigma'') \Downarrow \sigma'$.

  By Rule (Seq), we conclude

  $(c_1; c_2, \sigma) \Downarrow \sigma'$.

- Consider $(\sigma, \sigma') \in C[\![$ *if* b *then* c_1 *else* c_2 *fi* $]\!]$.

  Assume $\mathcal{B}[\![b]\!](\sigma) = true$, the false-case is similar.

  By definition of $C[\![-]\!]$, this means $(\sigma, \sigma') \in C[\![c_1]\!]$.

By the induction hypothesis,

$(c_1, \sigma) \Downarrow \sigma'$.

With Rule (bs_if_true), we have

$(\underline{\text{if}}\ b\ \underline{\text{then}}\ c_1\ \underline{\text{else}}\ c_2\ \underline{\text{fi}}, \sigma) \Downarrow \sigma'$.

- Consider $(\sigma, \sigma') \in \mathcal{C}[\![w]\!]$ with $w = \underline{\text{while}}\ b\ \underline{\text{do}}\ c\ \underline{\text{od}}$.

We have

$$\mathcal{C}[\![w]\!] = \mathrm{lfp}(\Gamma_{b,c}) = \bigcup_{i \in \mathbb{N}} \Gamma_{b,c}^i(\perp),$$

where the second equation holds by Kleene's Theorem.

This means

$(\sigma, \sigma') \in \Gamma_{b,c}^i(\perp)$ for some $i \in \mathbb{N}$.

We show by induction on $i \in \mathbb{N}$ that

for all $(\sigma_1, \sigma_2) \in \Gamma_{b,c}^i(\perp)$ we have $(w, \sigma_1) \Downarrow \sigma_2$.

Base: We have $\Gamma_{b,c}^0(\perp) = \perp = \emptyset$.

Case $i = 0$: Since there is no pair (σ_1, σ_2) in $\emptyset$, the claim trivially holds.

Induction step $i \to i+1$: Assume for all $(\sigma_1, \sigma_2) \in \Gamma_{b,c}^i(\perp)$ we have $(w, \sigma_1) \Downarrow \sigma_2$.

Consider

$$\begin{aligned}(\sigma_1, \sigma_2) \in \Gamma_{b,c}^{i+1}(\perp) &= \Gamma_{b,c}(\Gamma_{b,c}^i(\perp)) \\ &= \{(\sigma, \sigma') \mid \mathcal{S}[\![b]\!](\sigma) = \text{true and } (\sigma, \sigma') \in \Gamma_{b,c}^i(\perp) \circ \mathcal{C}[\![c]\!]\} \\ &\quad \cup \{(\sigma, \sigma) \mid \mathcal{S}[\![b]\!](\sigma) = \text{false}\}.\end{aligned}$$

↪ If $\mathcal{S}[\![b]\!](\sigma_1) = \text{false}$, this means $\sigma_2 = \sigma_1$.

By Rule (bswhilefalse), we indeed get

$$(w, \sigma_1) \Downarrow \sigma_1.$$

↳ If $S[\![b]\!](\sigma_1) = \text{true}$, then $(\sigma_1, \sigma_2) \in T_{b,c}^{i}(\bot) \circ C[\![c]\!]$

implies there is $\tilde{\sigma} \in \text{Stat}$ with $(\sigma_1, \tilde{\sigma}) \in C[\![c]\!]$

and $(\tilde{\sigma}, \sigma_2) \in T_{b,c}^{i}(\bot)$.

By the induction hypothesis for $T_{b,c}^{i}(\bot)$,

$$(w, \tilde{\sigma}) \Downarrow \sigma_2.$$

Moreover, by the hypothesis for c from the outer induction,

$$(c, \sigma_1) \Downarrow \tilde{\sigma}.$$

With Rule (bswhiletrue):

$$\frac{(c, \sigma_1) \Downarrow \tilde{\sigma} \quad (w, \tilde{\sigma}) \Downarrow \sigma_2}{(w, \sigma_1) \Downarrow \sigma_2}$$

□