Notes on Foundations of Programming Languages Denotational Semantics

Sandra Alves

December 7, 2021

Abstract

In the following sections we will explore some basic notions on domain theory and denotational semantics. For full details on the notions here presented see [Winskel, 1993, Nielson and Nielson, 1992].

1 Fixed-point theory

In this section we will develop the theory of fixed-points, with the purpose of defining the least fixed-point of a function, which is the fixed point that shares its results with the remaining fixed-points of the function. We formally define the fact that FIX F shares its results with all the other fixed-points for F, by considering an ordering relation between partial functions of type State → State.

Definition 1.1 *Let* g_1, g_2 : State → State, *be two partial functions, then* $g_1 \sqsubseteq g_2$ *if and only if, for all* $\sigma, sigma' \in$ State :

$$g_1\sigma = \sigma' \Rightarrow g_1\sigma = \sigma'.$$

For example, for the following g_1, g_2, g_3, g_4 : State → State:

$$\begin{aligned}
g_1\ \sigma &= \sigma \text{ for all } \sigma \\
g_2\ \sigma &= \begin{cases} \sigma & \text{if } \sigma(x) \geq 0 \\ \text{not def} & \text{otherwise} \end{cases} \\
g_3\ \sigma &= \begin{cases} \sigma & \text{if } \sigma(x) = 0 \\ \text{not def} & \text{otherwise} \end{cases} \\
g_4\ \sigma &= \begin{cases} \sigma & \text{if } \sigma(x) \leq 0 \\ \text{not def} & \text{otherwise} \end{cases}
\end{aligned}$$

we have:

$$\begin{aligned}
&g_1 \sqsubseteq g_1, \\
&g_2 \sqsubseteq g_1,\ g_2 \sqsubseteq g_2, \\
&g_3 \sqsubseteq g_1,\ g_3 \sqsubseteq g_2,\ g_3 \sqsubseteq g_3,\ g_3 \sqsubseteq g_4, \\
&g_4 \sqsubseteq g_1,\ g_4 \sqsubseteq g_4
\end{aligned}$$

The set of functions State → State, together with $\sqsubseteq$ defines a partial order set (*poset*).

Definition 1.2 *A* poset *is a pair* $(D, \sqsubseteq_D)$, *where* D *is a set and* $\sqsubseteq_D$ *is a relation:*

- *reflexive:* $\forall d \in D.d \sqsubseteq_D d$
- *transitive:* $\forall d_1, d_2, d_3 \in D.d_1 \sqsubseteq_D d_2 \wedge d_2 \sqsubseteq_D d_3 \Rightarrow d_1 \sqsubseteq_D d_3$
- *anti-symmetric:* $\forall d_1, d_2 \in D.d_1 \sqsubseteq_D d_2 \wedge d_2 \sqsubseteq_D d_1 \Rightarrow d_1 = d_2$

We call $\sqsubseteq_D$, *a partial order in* D. *If* $d \in D$ *satisfies* $\forall d' \in D.\ d \sqsubseteq_D d'$, *then* d *is called a minimal element.*

Lemma 1.3 *Let* $(D, \sqsubseteq)$ *be a poset. If* $(D, \sqsubseteq)$ *has a minimal element, then it is unique.*

Proof: Let d_1, d_2 be two minimal elements of $(D, \sqsubseteq)$. Since d_1 is a minimal, then $d_1 \sqsubseteq d_2$. Since d_2 is a minimal, then $d_2 \sqsubseteq d_1$. Following anti-symmetry of $\sqsubseteq$, then $d_1 = d_2$. □

Definition 1.4 *The minimal element of* $(D, \sqsubseteq)$ *is called bottom, and denoted by* $\perp_D$ *(or simply* $\perp$*).*

Lemma 1.5 *The pair* $(\mathtt{State} \to \mathtt{State}, \sqsubseteq)$ *is a poset, and the partial function* $\bot : \mathtt{State} \to \mathtt{State}$*, defined as:*

$$\bot\ \sigma = \mathit{not\ def}, \quad \mathit{for\ all}\ \sigma \in \Sigma$$

is the minimal element of $(\mathtt{State} \to \mathtt{State}, \sqsubseteq)$*.*

Proof: We start by showing that $\sqsubseteq$ is a partial order:

- **reflexive:** $g \sqsubseteq g$, since $g\ \sigma = \sigma'$ trivially implies $g\ \sigma = \sigma'$.
- **transitive:** Let $g_1 \sqsubseteq g_2$ and $g_2 \sqsubseteq g_3$ and let $g_1\ \sigma = \sigma'$. From $g_1 \sqsubseteq g_2$ it follows that $g_2\ \sigma = \sigma'$ and from $g_1 \sqsubseteq g_2$ it follows that $g_3\ \sigma = \sigma'$. Therefore $g_1 \sqsubseteq g_3$.
- **anti-symmetric:** Let $g_1 \sqsubseteq g_2$ and $g_2 \sqsubseteq g_1$. Let $g_1\ \sigma = \sigma'$, then $g_2\ \sigma = \sigma'$, which means that g_1 and g_2 are equal for σ. If $g_1\ sigma =$ not def then it follows from $g_2 \sqsubseteq g_1$ that $g_2\ sigma =$ not def, otherwise having $g_2\ sigma = \sigma'$ would imply $g_1\ sigma = \sigma'$, which contradicts $g_1\ sigma =$ not def. Therefore $g_1 = g_2$.

We now show that $\bot$ is the minimal element of the poset. Clearly $\bot$ is an element of $\mathtt{State} \to \mathtt{State}$. Furthermore, $\bot \sqsubseteq g$, for every g, since $\bot\ \sigma = \sigma'$ trivially implies $g\ \sigma = \sigma'$. □

We now establish our conditions on $\mathtt{FIX}\ F$:

- $\mathtt{FIX}\ F$ is a fix-point of F, that is, $F(\mathtt{FIX}\ F) = \mathtt{FIX}\ F$;
- $\mathtt{FIX}\ F$ is the least fix-point of F,

$$F\ g = g \ \Rightarrow\ \mathtt{FIX}\ F \sqsubseteq g.$$

Definition 1.6 *Let* $(D, \sqsubseteq)$ *be a poset and let* $Y \subseteq D$*. Consider an upper bound of* Y*, that is,* $d \in D$ *such that:*

$$\forall d' \in Y.\ d' \sqsubseteq d.$$

The least upper bound d *is such that, for all the upper bound* d' *of* Y*, then* $d \sqsubseteq d'$*. We denote the least upper bound of* Y *(if such exists), as* $\bigsqcup Y$*.*

A subset $Y \subseteq D$ *is called a* chain *if and only if:*

$$\forall d_1, d_2 \in Y.\ d_1 \sqsubseteq d_2\ \mathit{or}\ d_2 \sqsubseteq d_1.$$

Definition 1.7 *A poset* $(D, \sqsubseteq)$ *is a complete partially ordered set (CPO) if*

1. $(D, \sqsubseteq)$ *has a bottom element,* $\bot$*,*
2. $\bigsqcup Y$ *is defined for every chain* Y*.*

Some authors do not require a CPO to have a bottom element and so omit the first condition in the previous definition, using the term pointed CPO when both conditions hold.

2 Continuous functions

We will define the notion of continuous functions on CPOs, and show that these functions always have least fixed points.

Definition 2.1 *Let* $(D, \sqsubseteq_D)$ *and* $(E, \sqsubseteq_E)$ *be two CPOs and let* $f : D \to E$ *be a total function. We say that* f *is monotone if and only if:*

$$d_1 \sqsubseteq_D d_2 \Rightarrow f(d_1) \sqsubseteq_E f(d_2), \quad \mathit{for\ all}\ d_1, d_2 \in D.$$

For example, considering $(\mathcal{P}(\{a,b,c\}), \subseteq)$, $(\mathcal{P}(\{d,e\}), \subseteq)$ and $f_1, f_2 : \mathcal{P}(\{a,b,c\}) \to \mathcal{P}(\{d,e\})$ defined as follows:

$$
\begin{array}{l}
f_1(\emptyset) = \emptyset \\
f_1(\{a\}) = \{d\},\ f_1(\{b\}) = \{d\},\ f_1(\{c\}) = \{e\} \\
f_1(\{a,b\}) = \{d\},\ f_1(\{a,c\}) = \{d,e\},\ f_1(\{b,c\}) = \{d,e\} \\
f_1(\{a,b,c\}) = \{d,e\} \\
\\
f_2(\emptyset) = \{e\} \\
f_2(\{a\}) = \{d\},\ f_2(\{b\}) = \{e\},\ f_2(\{c\}) = \{e\} \\
f_2(\{a,b\}) = \{d\},\ f_2(\{a,c\}) = \{d\},\ f_2(\{b,c\}) = \{e\} \\
f_1(\{a,b,c\}) = \{d\}
\end{array}
$$

f_1 is monotone, but f_2 is not.

Lemma 2.2 *Let* $(A, \sqsubseteq_A)$, $(B, \sqsubseteq_B)$ *and* $(C, \sqsubseteq_C)$ *be three CPOs and* $f_1 : A \to B$, $f_2 : B \to C$ *two monotone functions. Then* $f_2 \circ f_1 : A \to C$ *is monotone.*

Proof: Let $a_1 \sqsubseteq_A a_2$. By monotonicity of f_1 we have $f_1(a_1) \sqsubseteq_B f_1(a_2)$, and by monotonicity of f_2 we have $f_2(f_1(a_1)) \sqsubseteq_C f_2(f_1(a_2))$. Thus

$$a_1 \sqsubseteq_A a_2 \Rightarrow f_2 \circ f_1(a_1) \sqsubseteq_C f_2 \circ f_1(a_2),$$

therefore $f_2 \circ f_1$ is monotone. □

Lemma 2.3 *Let* $(D, \sqsubseteq_D)$, $(E, \sqsubseteq_E)$ *be two CPOs and* $f : D \to E$, *a monotone function. If* $X = \langle x_1, \ldots, x_n \rangle$ *is a chain in* D, *then* $f\,X = \langle f\,x_1, \ldots, f\,x_n \rangle$ *is a chain in* E, *and:*

$$\bigsqcup \langle f\,x_i \rangle \sqsubseteq_E f(\bigsqcup \langle x_i \rangle)$$

Proof: If $X = \emptyset$, then the result follows trivially since $\bot_E \sqsubseteq_E f(\bot_D)$, therefore we will assume that $X \neq \emptyset$. We start by showing that $\langle f\ x_i \rangle$ is a chain in E. Let e_1, e_2 be two elements of $\langle f\ x_i \rangle$, then there exist $d_1, d_2 \in X$, such that $e_1 = f(d_1)$ and $e_2 = f(d_2)$. Since X is a chain, then either $d_1 \sqsubseteq_D d_2$ or $d_2 \sqsubseteq_D d_1$. By monotonicity of f, then either $f(d_1) \sqsubseteq_E f(d_2)$ or $f(d_2) \sqsubseteq_E f(d_1)$. That is, for every e_1, e_2 in $\langle f\ x_i \rangle$, either $e_1 \sqsubseteq_E e_2$ or $e_2 \sqsubseteq_E e_1$. Therefore $\langle f\ x_i \rangle$ is a chain.

We now need to show that:

$$\bigsqcup f\,X \sqsubseteq_E f(\bigsqcup X).$$

Let d be an arbitrary element of X. Since $\bigsqcup X$ is the least upper bound of X, then $d \sqsubseteq_D \bigsqcup X$. By monotonicity of f, we have $f(d) \sqsubseteq_E f(\bigsqcup X)$, since $f(d) \sqsubseteq_E f(\bigsqcup X)$ for every $d \in X$, then $f(\bigsqcup X)$ is an upper bound of $f\,X$, therefore, for the least upper bound of $f\,X$, we have

$$\bigsqcup f\,X \sqsubseteq_E f(\bigsqcup X).$$

□

Definition 2.4 *A function* $f : D \to E$ *between CPOs* $(D, \sqsubseteq_D)$, $(E, \sqsubseteq_E)$ *is continuous if it is monotone and, for all the non-empty chains* X*:*

$$\bigsqcup f\,X = f(\bigsqcup X).$$

If the property holds for the empty chain, that is $\bot = f\,\bot$, *then we say that* f *is strict.*

In general the least upper bounds of chains are not maintained. For example, let $(\mathcal{P}(\mathbb{N} \cup \{a\}), \subseteq)$ be a CPO and consider $f : \mathcal{P}(\mathbb{N} \cup \{a\}) \to \mathcal{P}(\mathbb{N} \cup \{a\})$ defined in the following way:

$$f(x) = \begin{cases} x & \text{if } x \text{ is finite} \\ x \cup \{a\} & \text{if } x \text{ is infinite} \end{cases}$$

The function f is monotone: if $x_1 \subseteq x_2$ then $f(x_1) \subseteq f(x_2)$. However, consider the chain $X = \{\{0, 1, \ldots, n\} \mid n \geq 0\} = \langle\{0\}, \{0, 1\}, \{0, 1, 2\}, \ldots\rangle$. The least upper bound of this chain is $\mathbb{N}$. The chain $f\,X = \{f\,\{0, 1, \ldots, n\} \mid n \geq 0\} = X$, therefore $\bigsqcup f\,X = \bigsqcup X = \mathbb{N}$. However $f(\bigsqcup X) = f(\mathbb{N}) = \mathbb{N} \cup \{a\}$.

Consider the function $f_1 : \mathcal{P}(\{a, b, c\}) \to \mathcal{P}(\{d, e\})$ defined before. Let X be a non-empty chain in $(\mathcal{P}(\{a, b, c\}), \subseteq)$. The least upper bound of X is the biggest set $x_0 \in X$. Then $f_1(\bigsqcup X) = f_1(x_0)$, and since $x_0 \in X$ then $f_1(x_0) \subseteq \bigsqcup f_1 X$. But f_1 is monotone, therefore $\bigsqcup f_1 X \subseteq f_1(\bigsqcup X)$. Therefore f is continuous. Furthermore, since $f_1(\emptyset) = \emptyset$, then f_1 is strict.

Lemma 2.5 *Let* $(A, \sqsubseteq_A)$, $(B, \sqsubseteq_B)$ *and* $(C, \sqsubseteq_C)$ *be three CPOs and* $f_1 : A \to B$, $f_2 : B \to C$ *two continuous functions. Then* $f_2 \circ f_1 : A \to C$ *is continuous.*

Proof: Since f_1 and f_2 are monotone, then $f_2 \circ f_1$ is also monotone by the previous lemma. Let X be a chain in A. By continuity of f_1 we have

$$\bigsqcup \langle f_1\ x \mid x \in X\rangle = f_1(\bigsqcup X)$$

. Since $\langle f_1\ x \mid x \in X\rangle$ is a chain in B, by continuity of f_2 we have

$$\bigsqcup \langle f_2\ y \mid y \in \langle f_1\ x \mid x \in X\rangle\rangle = f_2(\bigsqcup \langle f_1\ x \mid x \in X\rangle).$$

Therefore

$$\bigsqcup \langle f_2(f_1(x)) \mid x \in X\rangle = f_2(f_1(\bigsqcup X)).$$

□

Theorem 2.6 *Let* $f : D \to D$ *be a continuous function in the CPO* $(D, \sqsubseteq)$ *and let* $\perp$ *be its least upper bound. Then*

$$\texttt{FIX}\ f = \bigsqcup \langle f^n(\perp) \mid n \geq 0\rangle$$

defines the element of D *that is the least fixed point of* f.

Proof: Consider $f^0 = \texttt{id}$ and $f^{n+1} = f \circ f^n$, $\forall n \geq 0$. Since $f^0 \perp = \perp$ and $\perp \sqsubseteq d$, for all $d \in D$, then (by induction on n and given that f is monotone) we have that, for all $d \in D$:

$$f^n \perp \sqsubseteq f^n\ d.$$

Therefore $\langle f^n \perp \mid n \geq 0\rangle$ is a non-empty chain in D, since it follows that $f^n \perp \sqsubseteq f^m \perp$ for $n \leq m$. Since D is a CPO then there exists $\bigsqcup \langle f^n \perp \mid n \geq 0\rangle$, therefore $\texttt{FIX}\ f$ is defined.

We now show that $\texttt{FIX}\ f$ is indeed a fixed point of f, that is $f(\texttt{FIX}\ f) = \texttt{FIX}\ f$.

$$\begin{array}{rcl} f(\texttt{FIX}\ f) & = & f(\bigsqcup \langle f^n \perp \mid n \geq 0\rangle) \\ & = & \bigsqcup \langle f(f^n \perp) \mid n \geq 0\rangle \\ & = & \bigsqcup \langle f^n \perp \mid n \geq 1\rangle \\ & = & \bigsqcup (\langle f^n \perp \mid n \geq 1\rangle \cup \{\perp\})^* \\ & = & \bigsqcup \langle f^n \perp \mid n \geq 0\rangle \\ & = & \texttt{FIX}\ f \end{array}$$

* since $\bigsqcup(X \cup \{\perp\}) = \bigsqcup X$ for all chains X.

The last thing we need to show is that FIX f is the least fixed-point. Consider d another fixed point of f. Since $\perp \sqsubseteq d$ and f is monotone, then $f^n \perp \sqsubseteq f^n\ d,\ n \geq 0$. Since d is a fixed point (that is $f(d) = d$, then $f^n \perp \sqsubseteq d$. Therefore d is an upper bound of the chain $\{f^n(\perp) \mid n \geq 0\}$. Given that FIX f is the least upper bound of that chain it follows that FIX $f \sqsubseteq d$. □

For example let $F : \texttt{State} \to \texttt{State}$ be the following function:

$$(F\ g)\sigma = \begin{cases} g\ \sigma & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases}$$

The minimal element of $\texttt{State} \to \texttt{State}$ is the function $\perp$, such that $\perp\ \sigma =$ not def, for all $\sigma \in \Sigma$. We have:

$$\begin{array}{lcllcl}
F^0 \perp \sigma & = & \text{not def} & & & \\
F^1 \perp \sigma & = & F^1 \perp \sigma & = \begin{cases} \perp\ \sigma & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases} & = & \begin{cases} \text{not def} & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases} \\
F^2 \perp \sigma & = & F(F^1 \perp)\ \sigma & = \begin{cases} (F^1 \perp)\ \sigma & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases} & = & \begin{cases} \text{not def} & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases}
\end{array}$$

Therefore

$$\texttt{FIX}\ F = \begin{cases} \text{not def} & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases}$$

3 Denotational semantics for IMP

We now define a denotational semantics for the IMP language.

Definition 3.1 *The function $\mathcal{A}[\![\cdot]\!] : \texttt{AExp} \to (\texttt{State} \to \mathbb{Z})$, recursively defines the denotational semantics of arithmetic expressions, in the following way:*

$$\begin{array}{rcl@{\qquad}rcl}
\mathcal{A}[\![n]\!]\sigma & = & \mathcal{N}[\![n]\!]^* & \mathcal{A}[\![a_1 + a_2]\!]\sigma & = & \mathcal{A}[\![a_1]\!]\sigma + \mathcal{A}[\![a_2]\!]\sigma \\
\mathcal{A}[\![x]\!]\sigma & = & \sigma(x) & \mathcal{A}[\![a_1 - a_2]\!]\sigma & = & \mathcal{A}[\![a_1]\!]\sigma - \mathcal{A}[\![a_2]\!]\sigma \\
 & & & \mathcal{A}[\![a_1 * a_2]\!]\sigma & = & \mathcal{A}[\![a_1]\!]\sigma * \mathcal{A}[\![a_2]\!]\sigma
\end{array}$$

$(^*)$ *The function $\mathcal{N}[\![]\!] : \texttt{Num} \to \mathbb{Z}$, allows for different representations of numbers. For simplicity, we will assume that $\texttt{Num} = \mathbb{Z}$, therefore $\mathcal{N}[\![]\!]$ is the identity function.*

Definition 3.2 *The function $\mathcal{B}[\![\cdot]\!] : \texttt{BExp} \to (\texttt{State} \to T)$, recursively defines the denotational semantics of boolean expressions, in the following way:*

$$\begin{array}{rcl@{\qquad}rcl}
\mathcal{B}[\![1]\!]\sigma & = & 1 & & & \\
\mathcal{B}[\![0]\!]\sigma & = & 0 & & & \\
\mathcal{B}[\![a_1 = a_2]\!]\sigma & = & \begin{cases} 1 & \text{if } \mathcal{A}[\![a_1]\!]\sigma = \mathcal{A}[\![a_2]\!]\sigma \\ 0 & \text{otherwise} \end{cases} & \mathcal{B}[\![a_1 \leq a_2]\!]\sigma & = & \begin{cases} 1 & \text{if } \mathcal{A}[\![a_1]\!]\sigma \leq \mathcal{A}[\![a_2]\!]\sigma \\ 0 & \text{otherwise} \end{cases} \\
\mathcal{B}[\![\neg b]\!]\sigma & = & \begin{cases} 1 & \text{if } \mathcal{B}[\![b]\!]\sigma = 0 \\ 0 & \text{if } \mathcal{B}[\![b]\!]\sigma = 1 \end{cases} & \mathcal{B}[\![b_1 \wedge b_2]\!]\sigma & = & \begin{cases} 1 & \text{if } \mathcal{B}[\![b_1]\!]\sigma = \mathcal{B}[\![b_2]\!]\sigma = 1 \\ 0 & \text{otherwise} \end{cases}
\end{array}$$

Definition 3.3 *Given* $p : \mathtt{State} \to T$*, and* $g_1, g_2 : \mathtt{State} \to \mathtt{State}$*, the function* $\mathtt{cond} : (\mathtt{State} \to T) \times (\mathtt{State} \to \mathtt{State}) \times (\mathtt{State} \to \mathtt{State}) \to (\mathtt{State} \to \mathtt{State})$ *is defined in the following way:*

$$\mathrm{cond}(p, g_1, g_2)\sigma = \begin{cases} g_1\sigma & \textit{if } p\sigma = 1 \\ g_2\sigma & \textit{if } p\sigma = 0 \end{cases}$$

Definition 3.4 *The function* $\mathcal{S}_{ds}[\![\cdot]\!] : \mathtt{Com} \to (\mathtt{State} \to \mathtt{State})$*, recursively defines the denotational semantics of commands, in the following way:*

$$\begin{array}{rcl} \mathcal{S}_{ds}[\![\mathtt{skip}]\!] & = & \mathrm{id} \\ \mathcal{S}_{ds}[\![x := a]\!]\sigma & = & \sigma[\mathcal{A}[\![a]\!]\sigma/x] \\ \mathcal{S}_{ds}[\![c_1; c_2]\!] & = & \mathcal{S}_{ds}[\![c_2]\!] \circ \mathcal{S}_{ds}[\![c_1]\!] \\ \mathcal{S}_{ds}[\![\mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2]\!] & = & \mathrm{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![c_1]\!], \mathcal{S}_{ds}[\![c_2]\!]) \\ \mathcal{S}_{ds}[\![\mathtt{while}\ b\ \mathtt{do}\ c]\!] & = & \mathtt{FIX}\ F \\ \textit{where}\ \ F\,g & = & \mathrm{cond}(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![c]\!], \mathrm{id}) \end{array}$$

In particular for sequences of commands we have:

$$\begin{array}{rcl} \mathcal{S}_{ds}[\![c_1; c_2]\!]\sigma & = & (\mathcal{S}_{ds}[\![c_2]\!] \circ \mathcal{S}_{ds}[\![c_1]\!])\sigma \\ & = & \begin{cases} \sigma'' & \text{if } \exists \sigma'.\mathcal{S}_{ds}[\![c_1]\!]\sigma = \sigma' \text{ and } \mathcal{S}_{ds}[\![c_2]\!]\sigma' = \sigma'' \\ \text{not def} & \text{if } \mathcal{S}_{ds}[\![c_1]\!]\sigma \text{ not def, or } \mathcal{S}_{ds}[\![c_1]\!]\sigma = \sigma' \text{ but } \mathcal{S}_{ds}[\![c_2]\!]\sigma' \text{ not def.} \end{cases} \end{array}$$

And for conditionals we have:

$$\begin{array}{rcl} \mathcal{S}_{ds}[\![\mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2]\!]\sigma & = & \mathrm{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![c_1]\!], \mathcal{S}_{ds}[\![c_2]\!])\sigma \\ & = & \begin{cases} \sigma' & \text{if } \mathcal{B}[\![b]\!]\sigma = 1 \text{ and } \mathcal{S}_{ds}[\![c_1]\!]\sigma = \sigma' \\ & \text{or } \mathcal{B}[\![b]\!]\sigma = 0 \text{ and } \mathcal{S}_{ds}[\![c_2]\!]\sigma = \sigma' \\ \text{not def} & \text{otherwise.} \end{cases} \end{array}$$

Regarding a command of the form $\mathtt{while}\ b\ \mathtt{do}\ c$, its effect should be the same as the effect of the command $\mathtt{if}\ b\ \mathtt{then}\ (c; \mathtt{while}\ b\ \mathtt{do}\ c)\ \mathtt{else}\ \mathtt{skip}$:

$$\mathcal{S}_{ds}[\![\mathtt{while}\ b\ \mathtt{do}\ c]\!] = \mathrm{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![\mathtt{while}\ b\ \mathtt{do}\ c]\!] \circ \mathcal{S}_{ds}[\![c]\!], \mathrm{id})$$

However, this is not a compositional definition, which is what we expect of a denotational semantics definition. Nevertheless, the recursive definition of $\mathcal{S}_{ds}[\![\mathtt{while}\ b\ \mathtt{do}\ c]\!]$, points to the definition above, as the fixed-point of the function F defined as:

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

For example consider the command:

$$\mathtt{while}\ \neg(x = 0)\ \mathtt{do}\ \mathtt{skip}$$

with the corresponding function:

$$(F\ g)\sigma = \begin{cases} g\sigma & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases}$$

The following function $g_1 : \mathtt{State} \to \mathtt{State}$, is a fixed-point of F:

$$g_1\ \sigma = \begin{cases} \text{not def} & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases}$$

Note that

$$
\begin{aligned}
(F\ g_1)\sigma &= \begin{cases} g_1\sigma & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases} \\
&= \begin{cases} \text{not def} & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases} \\
&= g_1\sigma
\end{aligned}
$$

The function $g_2 : \mathtt{State} \rightarrow \mathtt{State}$, defined as $g_2\sigma = \text{not def}, \quad \forall\sigma \in \mathtt{State}$, is not a fixed-point of F.

The issue here is that there are functions that have several fixed-points, whereas other do not have fixed point at all. For example

$$
(F_1\ g) = \begin{cases} g_1 & \text{if } g = g_2 \\ g_2 & \text{otherwise} \end{cases}.
$$

Considering the different effects of executing a `while b do c` statement, the following conditions must hold for the fixed-point of the functional F. The fixed point of F, denoted by $\mathtt{FIX}\ F$ is a partial function $g_0 : \mathtt{State} \rightarrow \mathtt{State}$, such that:

- g_0 is a fixed-point of F, that is, $F; g_0 = g_0$ and,
- if g is another fixed-point of F ($F; g = g$), then

  $$g_0\ \sigma = \sigma' \quad \Rightarrow \quad g\ \sigma = \sigma',\ \forall\sigma, \sigma' \in \mathtt{State}$$

  If $g_0\ \sigma$ is not defined, then there are no restrictions for $g\ \sigma$.

Recall from the previous section that for continuous functions of CPOs, the least upper bound of the chain $\{F^n(\perp) \mid n \geq 0\}$ is the least fixed point of F, satisfying the conditions above. Therefore, the following results guarantee the completeness of the $\mathcal{S}_{ds}[\![]\!]$ function for commands.

Lemma 3.5 *The pair* $(\mathtt{State} \rightarrow \mathtt{State}, \sqsubseteq)$ *is a CPO.*

Lemma 3.6 *Given* $p : State \rightarrow T$ *and* $g_0 : \mathtt{State} \rightarrow \mathtt{State}$*, let*

$$F\ g = cond(p, g, g_0)$$

. Then F *is a continuous function.*

Lemma 3.7 *Given* $g_0 : \mathtt{State} \rightarrow \mathtt{State}$*, let*

$$F\ g = g \circ g_0$$

. Then F *is a continuous function.*

Proposition 3.8 *The function* $\mathcal{S}_{ds}[\![]\!] : \mathtt{Com} \rightarrow (\mathtt{State} \rightarrow \mathtt{State})$ *is a total function.*

Proof: By induction on $c \in \mathtt{Com}$.

- **[skip]:** The function id is a total function.
- **[atrb]:** The function $\lambda\sigma.\sigma[\mathcal{A}[\![a]\!]\sigma/x]$ is a total function. Note that $\mathcal{A}[\![a]\!]$ is a total function.
- **[seqn]:** Consider the command $c_1; c_2$. By induction hypothesis, both $\mathcal{S}_{ds}[\![c_1]\!]$ and $\mathcal{S}_{ds}[\![c_2]\!]$ are well-defined, therefore its composition $\mathcal{S}_{ds}[\![c_2]\!] \circ \mathcal{S}_{ds}[\![c_1]\!]$ is well-defined.

- **[if]:** Consider the command `if` b `then` c_1 `else` c_2. By induction hypothesis, both $\mathcal{S}_{ds}[\![c_1]\!]$ and $\mathcal{S}_{ds}[\![c_2]\!]$ are well-defined, therefore $\mathrm{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![c_1]\!], \mathcal{S}_{ds}[\![c_2]\!])$ is well-defined.

- **[while]:** Consider the command `while` b `do` c. By induction hypothesis $\mathcal{S}_{ds}[\![c]\!]$ is well-defined. Let F_1, F_2 be defined as follows:

$$\begin{array}{rcl} F_1\ g & = & \mathrm{cond}(\mathcal{B}[\![b]\!], g, \mathrm{id}) \\ F_2\ g & = & g \circ \mathcal{S}_{ds}[\![c]\!]. \end{array}$$

  F_1 and F_2 are both continuous functions by the previous lemmas. Therefore $F\ g = F_1(F_2\ \ g)$ is continuous. Therefore, `FIX` F is defined, which implies that $\mathcal{S}_{ds}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]$ is well-defined.

□

Example 3.9 *Consider the command* $y := 1; \texttt{while}\ \neg(x = 1)\ \texttt{do}\ (y := y * x; x := x - 1))$.

$$\begin{array}{l} \mathcal{S}_{ds}[\![y := 1; \texttt{while}\ \neg(x = 1)\ \texttt{do}\ (y := y * x; x := x - 1))]\!]\sigma_0 \\ = (\texttt{FIX}\ F)\sigma_0[1/y] \end{array}$$

with

$$F\ g\ \sigma \quad = \quad \begin{cases} g(\mathcal{S}_{ds}[\![y := y * x; x := x - 1]\!]\sigma & \textit{if } \mathcal{B}[\![\neg(x = 1)]\!])\sigma = 1 \\ \sigma & \textit{if } \mathcal{B}[\![\neg(x = 1)]\!]\sigma = 0 \end{cases}$$

that is:

$$F\ g\ \sigma \quad = \quad \begin{cases} g(\sigma[\sigma(y) * \sigma(x)/y][\sigma(x) - 1/x]) & \textit{if } \sigma(x) \neq 1 \\ \sigma & \textit{if } \sigma(x) = 1 \end{cases}.$$

We then have:

$$(F^0\ \bot)\ \sigma \quad = \quad \textit{not def}$$

$$(F^1\ \bot)\ \sigma \quad = \quad \begin{cases} \textit{not def} & \textit{if } \sigma(x) \neq 1 \\ \sigma & \textit{if } \sigma(x) = 1 \end{cases}$$

$$(F^2\ \bot)\ \sigma \quad = \quad \begin{cases} \textit{not def} & \textit{if } \sigma(x) \neq 1 \textit{ and } \sigma(x) \neq 2 \\ \sigma[\sigma(y) * 2/y][1/x] & \textit{if } \sigma(x) = 2 \\ \sigma & \textit{if } \sigma(x) = 1 \end{cases}$$

$$(F^n\ \bot)\ \sigma \quad = \quad \begin{cases} \textit{not def} & \textit{if } \sigma(x) > n \textit{ or } \sigma(x) \geq 1 \\ \sigma[\sigma(y) * j * 2/y][1/x] & \textit{if } \sigma(x) = 2 \\ \sigma & \textit{if } \sigma(x) = 1 \end{cases}$$

Therefore

$$(\texttt{FIX}\ F)\ \sigma \quad = \quad \begin{cases} \textit{not def} & \textit{if } \sigma(x) < 1 \\ \sigma[\sigma(y) * n * \cdots * 2 * 1/y][1/x] & \textit{if } \sigma(x) = 2 \\ \sigma & \textit{if } \sigma(x) = 1 \end{cases}$$

3.1 Properties of the denotational semantics

Definition 3.10 *Two commands* c_1, c_2 *are semantically equivalent if and only if* $\mathcal{S}_{ds}[\![c_1]\!] = \mathcal{S}_{ds}[\![c_2]\!]$.

Definition 3.11 *Let* $\mathcal{S}_{st}[\![\cdot]\!] : \mathtt{Com} \to (\mathtt{State} \to \mathtt{State})$ *be the partial function defined in the following way:*

$$\mathcal{S}_{st}[\![c]\!]\sigma = \begin{cases} \sigma' & \text{if } \langle c, \sigma\rangle \Rightarrow^* \sigma' \\ \text{not def} & \text{otherwise} \end{cases}$$

Theorem 3.12 *For every* $c \in \mathtt{Com}$, $\mathcal{S}_{st}[\![c]\!] = \mathcal{S}_{ds}[\![c]\!]$.

Both functions $\mathcal{S}_{st}[\![c]\!], \mathcal{S}_{ds}[\![c]\!]$, are functions in $(State \to \mathtt{State}, \sqsubseteq)$, which is a poset. To show that two elements d_1, d_2 from a poset are equal, we need to show that $d_1 \sqsubseteq d_2$ and $d_2 \sqsubseteq d_1$. Therefore, we need to prove that: $\mathcal{S}_{st}[\![c]\!] \sqsubseteq \mathcal{S}_{ds}[\![c]\!]$ and $\mathcal{S}_{ds}[\![c]\!] \sqsubseteq \mathcal{S}_{st}[\![c]\!]$.

Lemma 3.13 *Let* $c \in \mathtt{Com}$, *then* $\mathcal{S}_{st}[\![c]\!] \sqsubseteq \mathcal{S}_{ds}[\![c]\!]$.

Proof: We will show that for every states σ, σ' we have:

$$\langle c, \sigma\rangle \Rightarrow^* \sigma' \Rightarrow \mathcal{S}_{ds}[\![c]\!]\sigma = \sigma' \quad (1)$$

To that end we will establish the following:

$$\begin{array}{rcl} \langle c, \sigma\rangle \Rightarrow \sigma' & \Rightarrow & \mathcal{S}_{ds}[\![c]\!]\sigma = \sigma' \\ \langle c, \sigma\rangle \Rightarrow \langle c', \sigma'\rangle & \Rightarrow & \mathcal{S}_{ds}[\![c]\!]\sigma = \mathcal{S}_{ds}[\![c']\!]\sigma' \end{array} \quad (2)$$

Assuming that (2) holds, then (1) holds by induction on k, where k is the length of the sequence

$$\langle c, \sigma\rangle \Rightarrow^k \sigma'.$$

We show (2) by induction on the derivation tree of $\langle c, \sigma\rangle \Rightarrow \sigma'$ or $\langle c, \sigma\rangle \Rightarrow \langle c', \sigma'\rangle$.

- **[skip]:** We have $\langle \mathtt{skip}, \sigma\rangle \Rightarrow \sigma$ and $\mathcal{S}_{ds}[\![\mathtt{skip}]\!]\sigma = \mathrm{id}\ \sigma = \sigma$.

- **[assn]:** We have $\langle x := a, \sigma\rangle \Rightarrow \sigma[n/x]$ and $\mathcal{S}_{ds}[\![x := a]\!]\sigma = \sigma[\mathcal{A}[\![a]\!]\sigma/x]$. (By induction, we need to prove that $\langle a, \sigma\rangle \to n$ if and only if $\mathcal{A}[\![a]\!]\sigma = n$). Therefore $\mathcal{S}_{ds}[\![x := a]\!]\sigma = \sigma[n/x]$.

- **[seqn$_1$]:** We have $\langle c_1; c_2, \sigma\rangle \Rightarrow \langle c_1'; c_2, \sigma'\rangle$, which follows from $\langle c_1, \sigma\rangle \Rightarrow \langle c_1', \sigma'\rangle$. By induction hypothesis we have $\mathcal{S}_{ds}[\![c_1]\!]\sigma = \mathcal{S}_{ds}[\![c_1']\!]\sigma'$. Therefore

$$\begin{array}{rcl} \mathcal{S}_{ds}[\![c_1; c_2]\!]\sigma & = & \mathcal{S}_{ds}[\![c_2]\!](\mathcal{S}_{ds}[\![c_1]\!]\sigma) \\ & = & \mathcal{S}_{ds}[\![c_2]\!](\mathcal{S}_{ds}[\![c_1']\!]\sigma') \\ & = & \mathcal{S}_{ds}[\![c_1'; c_2]\!]\sigma' \end{array}$$

- **[seqn$_2$]:** We have $\langle c_1; c_2, \sigma\rangle \Rightarrow \langle c_2, \sigma'\rangle$, which follows from $\langle c_1, \sigma\rangle \Rightarrow \sigma'$. By induction hypothesis we have $\mathcal{S}_{ds}[\![c_1]\!]\sigma = \sigma'$. Therefore

$$\begin{array}{rcl} \mathcal{S}_{ds}[\![c_1; c_2]\!]\sigma & = & \mathcal{S}_{ds}[\![c_2]\!](\mathcal{S}_{ds}[\![c_1]\!]\sigma) \\ & = & \mathcal{S}_{ds}[\![c_2]\!]\sigma' \end{array}$$

- **[if$_1$]:** We have $\mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2\sigma \Rightarrow \langle c_1, \sigma\rangle$, which follows from $\langle b, \sigma\rangle \to 1$. (By induction, we need to prove that $\langle b, \sigma\rangle \to t$ if and only if $\mathcal{B}[\![b]\!]\sigma = t$). Therefore

$$\begin{array}{rcl} \mathcal{S}_{ds}[\![\mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2]\!]\sigma & = & \mathrm{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![c_1]\!], \mathcal{S}_{ds}[\![c_2]\!])\sigma \\ & = & \mathrm{cond}(1, \mathcal{S}_{ds}[\![c_1]\!], \mathcal{S}_{ds}[\![c_2]\!])\sigma \\ & = & \mathcal{S}_{ds}[\![c_1]\!]\sigma \end{array}$$

- **[if_0]:** Analogous to the previous case.
- **[while]:** We have $\mathcal{S}_{ds}[\![\texttt{while b do c}]\!]\sigma \to \mathcal{S}_{ds}[\![\texttt{if b then c;while b do c else skip}]\!]\sigma$. By the definition of $\mathcal{S}_{ds}[\![]\!]$, we have $\mathcal{S}_{ds}[\![\texttt{while b do c}]\!] = \texttt{FIX}\ F$, where $F\ g = \text{cond}(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![c]\!], \text{id})$, therefore:

$$\begin{array}{rcl} \mathcal{S}_{ds}[\![\texttt{while b do c}]\!] & = & \texttt{FIX}\ F \\ & = & F(\texttt{FIX}\ F) \\ & = & \text{cond}(\mathcal{B}[\![b]\!], (\texttt{FIX}\ F) \circ \mathcal{S}_{ds}[\![c]\!], \text{id}) \\ & = & \text{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![\texttt{while b do c}]\!] \circ \mathcal{S}_{ds}[\![c]\!], \text{id}) \\ & = & \text{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![\texttt{c;while b do c}]\!], \mathcal{S}_{ds}[\![\texttt{skip}]\!]) \\ & = & \mathcal{S}_{ds}[\![\texttt{if b then c;while b do c else skip}]\!] \end{array}$$

□

Lemma 3.14 *Let* $c \in \texttt{Com}$, *then* $\mathcal{S}_{ds}[\![c]\!] \sqsubseteq \mathcal{S}_{st}[\![c]\!]$.

Proof: By induction on c.

- **[skip]:** $\mathcal{S}_{ds}[\![\text{skip}]\!]\sigma = \sigma = \mathcal{S}_{st}[\![\text{skip}]\!]\sigma$.
- **[assn]:** We have $\mathcal{S}_{st}[\![x := a]\!]\sigma = \sigma[n/x]$ (with $\mathcal{A}[\![a]\!]\sigma = n$). From $\mathcal{A}[\![a]\!]\sigma = n$ it follows that $\langle a, \sigma\rangle \to n$, therefore $\langle x := a, \sigma\rangle \to \sigma[n/x]$, which implies $\mathcal{S}_{st}[\![\text{skip}]\!]\sigma = \sigma[n/x]$.
- **[seqn]:** By induction hypothesis we have $\mathcal{S}_{ds}[\![c_1]\!] \sqsubseteq \mathcal{S}_{st}[\![c_1]\!]$ and $\mathcal{S}_{ds}[\![c_2]\!] \sqsubseteq \mathcal{S}_{st}[\![c_2]\!]$, therefore:

$$\begin{array}{rcl} \mathcal{S}_{ds}[\![c_1;c_2]\!] & = & \mathcal{S}_{ds}[\![c_2]\!] \circ \mathcal{S}_{ds}[\![c_1]\!] \\ & \sqsubseteq & \mathcal{S}_{st}[\![c_2]\!] \circ \mathcal{S}_{st}[\![c_1]\!] \end{array}$$

  Recall that, if $\langle c_1, \sigma\rangle \Rightarrow^* \sigma'$, then $\langle c_1;c_2, \sigma\rangle \Rightarrow^* \langle c_2, \sigma'\rangle$. Therefore

$$\mathcal{S}_{st}[\![c_2]\!] \circ \mathcal{S}_{st}[\![c_1]\!] \sqsubseteq \mathcal{S}_{st}[\![c_1;c_2]\!]$$

- **[if]:** By induction hypothesis we have $\mathcal{S}_{ds}[\![c_1]\!] \sqsubseteq \mathcal{S}_{st}[\![c_1]\!]$ and $\mathcal{S}_{ds}[\![c_2]\!] \sqsubseteq \mathcal{S}_{st}[\![c_2]\!]$, therefore:

$$\begin{array}{rcl} \mathcal{S}_{ds}[\![\texttt{if b then } c_1 \texttt{ else } c_2]\!] & = & \text{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![c_1]\!], \mathcal{S}_{ds}[\![c_2]\!]) \\ & \sqsubseteq & \text{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{st}[\![c_1]\!], \mathcal{S}_{st}[\![c_2]\!]) \end{array}$$

  Note that

$$\mathcal{S}_{st}[\![\texttt{if b then } c_1 \texttt{ else } c_2]\!]\sigma = \begin{cases} \mathcal{S}_{st}[\![c_1]\!]\sigma & \text{if } \langle b, \sigma\rangle \to 1 \\ \mathcal{S}_{st}[\![c_2]\!]\sigma & \text{if } \langle b, \sigma\rangle \to 0 \end{cases}$$

  Note that $\langle b, \sigma\rangle \to t$ iff $\mathcal{B}[\![b]\!]\sigma = t$, therefore:

$$\text{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![c_1]\!], \mathcal{S}_{ds}[\![c_2]\!]) = \mathcal{S}_{st}[\![\texttt{if b then } c_1 \texttt{ else } c_2]\!].$$

- **[while]:** By induction hypothesis we have $\mathcal{S}_{ds}[\![c]\!] \sqsubseteq \mathcal{S}_{st}[\![c]\!]$, and $\mathcal{S}_{ds}[\![\texttt{while b do c}]\!] = \texttt{FIX}\ F$, where $F\ g = \text{cond}(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![c]\!], \text{id})$. F is continuous, therefore:

$$\begin{array}{rcl} F(\mathcal{S}_{st}[\![\texttt{while b do c}]\!]) & = & \text{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{st}[\![\texttt{while b do c}]\!] \circ \mathcal{S}_{ds}[\![c]\!], \text{id}) \\ & \sqsubseteq & \text{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{st}[\![\texttt{while b do c}]\!] \circ \mathcal{S}_{st}[\![c]\!], \text{id}) \\ & \sqsubseteq & \text{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{st}[\![\texttt{c;while b do c}]\!], \text{id}) \\ & = & \mathcal{S}_{st}[\![\texttt{while b do c}]\!] \end{array}$$

  Note that, if $f\ d \sqsubseteq d$, then $\texttt{FIX}\ f \sqsubseteq d$, therefore $\texttt{FIX}\ F \sqsubseteq \mathcal{S}_{st}[\![\texttt{while b do c}]\!]$.

□

Exercises

1 Given two posets$(A, \sqsubseteq_A)$ and $(B, \sqsubseteq_B)$:

(a) Show that $(A \times B, \sqsubseteq)$ is also a poset, considering:

$$(a, b) \sqsubseteq (a', b') \text{ iff } a \sqsubseteq_A a' \wedge b \sqsubseteq_B b'$$

(b) Show that, if s is the least upper bound of the chain $\langle a_i \rangle$ and if s' is the least upper bound of the chain $\langle b_i \rangle$ then (s, s') is the leas upper bound of the chain $\langle (a_i, b_i) \rangle$ in $(A \times B, \sqsubseteq)$. As a consequence, if $(A, \sqsubseteq_A)$ and $(B, \sqsubseteq_B)$ are CPOs then $(A \times B, \sqsubseteq)$ is also a CPO.

2 Draw the Hasse diagram for the following partial orders:

(a) $(2^A, \subseteq)$, where $A = \{1, 2, 3\}$.

(b) $(\{\mathsf{true}, \mathsf{false}\}_\perp \times A_\perp, \sqsubseteq)$ where $A = \{x, y, z\}$.

(c) $(\{\mathsf{true}, \mathsf{false}\}_\perp \times A_\perp, \sqsubseteq)$

3 Considering the denotational semantics for IMP, determine:

(a) $\mathcal{S}_{ds}[\![\texttt{while } 1 \texttt{ do skip}]\!]$

(b) $\mathcal{S}_{ds}[\![\texttt{while } 1 \texttt{ do } x := x + 1]\!]$

(c) $\mathcal{S}_{ds}[\![\texttt{while } x = 3 \texttt{ do skip}]\!]$

(d) $\mathcal{S}_{ds}[\![x := 3; \texttt{while } 1 \leq x \texttt{ do } x := x - 1]\!]$

(e) $\mathcal{S}_{ds}[\![y := 0; \texttt{while } 1 \leq x \texttt{ do } (y := x + y; x := x - 1]\!]$

4 Following the denotational semantics given in the course determine the functional F associated to the command

$$\texttt{while } \neg(x = 0) \texttt{ do } x := x - 1$$

Determine which of the following partial functions on $\texttt{State} \to \texttt{State}$, are fixed points of F:

$$g_1\ \sigma = \text{not def, for all } \sigma \qquad g_2\ \sigma = \begin{cases} \sigma[0/x] & \text{if } \sigma(x) \geq 0 \\ \text{not def} & \text{if } \sigma(x) < 0 \end{cases}$$

$$g_3\ \sigma = \begin{cases} \sigma[0/x] & \text{if } \sigma(x) \geq 0 \\ \sigma & \text{if } \sigma(x) < 0 \end{cases} \qquad g_4\ \sigma = \sigma[0/x], \text{ for all } \sigma$$

$$g_5\ \sigma = \sigma, \text{ for all } \sigma$$

5 Considering the operational and denotational semantics for IMP, prove that:

(a) For all arithmetic expressions $a \in \texttt{AExp}$ and states $\sigma \in \texttt{State}$, $\langle a, \sigma \rangle \to n$ if and only if $\mathcal{A}[\![a]\!]\sigma = n$.

(b) For all boolean expressions $b \in \texttt{BExp}$ and states $\sigma \in \texttt{State}$ $\langle b, \sigma \rangle \to t$ if and only if $\mathcal{B}[\![b]\!]\sigma = t$.

6 Consider the denotational semantics of IMP. Show the equivalence between the following commands:

(a) $\texttt{skip}; c$ and c;

(b) $c_1; (c_2; c_3)$ and $(c_1; c_2); c_3$;

(c) $\texttt{if } b \texttt{ then } (c_1;\ c \texttt{ else } (c_2;\ c)$ and $(\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2); c$;

(d) $\texttt{while } b \texttt{ do } c_1; \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2$ and $\texttt{while } b \texttt{ do } c_1; c_2$.

7 Consider the function $f : \mathbb{N}_\perp \to \mathbb{N}_\perp$ defined as $f(3) = 1$ and $f(x) = f(x+2) + 5$, for $x \neq 3$.

(a) Determine $F : [\mathbb{N}_\perp \to \mathbb{N}_\perp] \to [\mathbb{N}_\perp \to \mathbb{N}_\perp]$ such that $\mathsf{FIX}\ F = f$

(b) Show that F is continuous.

(c) Calculate $\mathsf{FIX}\ F$.

8 Let f be a continuous function. Show that, if $f\ d \sqsubseteq d$, then $\mathtt{FIX}\ f \sqsubseteq d$.

References

[Nielson and Nielson, 1992] Nielson, H. R. and Nielson, F. (1992). *Semantics with Applications: A Formal Introduction.* John Wiley & Sons, Inc., New York, NY, USA.

[Winskel, 1993] Winskel, G. (1993). *The Formal Semantics of Programming Languages: An Introduction.* MIT Press, Cambridge, MA, USA.