G54FOP: Lecture 17 & 18

Denotational Semantics and Domain Theory III & IV

Henrik Nilsson

University of Nottingham, UK

These Two Lectures

- Revisit attempt to define denotational semantics for small imperative language
- Discussion of the reasons for it being inadequate
- Fixed point semantics
- Basic domain theory
- The Least Fixed Point Theorem

Recap: Imperative Language (1)

Syntax of expressions:

$e \rightarrow$		*expressions:*
	x	*variable*
|	n	*constant number,* $n \in \mathbb{N}$
|	**true**	*constant true*
|	**false**	*constant false*
|	**not** e	*logical negation*
|	e **&&** e	*logical conjunction*
$\ldots$		

Recap: Imperative Language (2)

$e \rightarrow$		*expressions:*
$\ldots$		
|	e **+** e	*addition*
|	e **-** e	*subtraction*
|	e **=** e	*numeric equality test*
|	e **<** e	*numeric less than test*

Recap: Imperative Language (3)

Syntax of commands:

$c \rightarrow$		*commands:*
$\mid$	`skip`	*no operation*
$\mid$	x `:=` e	*assignment*
$\mid$	c `;` c	*sequence*
$\mid$	`if` e `then` c `else` c	*conditional*
$\mid$	`while` e `do` c	*iteration*

Rcp: Denotational Semantics for IL (1)

We take the ***semantic domain*** to be $\mathbb{N}$ for simplicity.
A ***store*** maps a variable name to its value:

$$\begin{aligned} \Sigma &= x \rightarrow \mathbb{N} \\ \sigma &: \Sigma \end{aligned}$$

We need two ***semantic functions***, one for expressions (no side effects), one for commands:

$$\begin{aligned} \mathrm{E}[\![\cdot]\!] &: e \rightarrow (\Sigma \rightarrow \mathbb{N}) \\ \mathrm{C}[\![\cdot]\!] &: c \rightarrow (\Sigma \rightarrow \Sigma) \quad \text{[Not correct yet!]} \end{aligned}$$

(Note: $e \rightarrow (\Sigma \rightarrow \mathbb{N}) = e \rightarrow \Sigma \rightarrow \mathbb{N}$ etc.)

Rcp: Denotational Semantics for IL (2)

$\mathrm{E}[\![\cdot]\!]$: some typical cases:

$$\begin{aligned} \mathrm{E}[\![x]\!]\ \sigma &= \sigma\ x \\ \mathrm{E}[\![n]\!]\ \sigma &= n \\ \mathrm{E}[\![\texttt{true}]\!]\ \sigma &= 1 \\ \mathrm{E}[\![\texttt{false}]\!]\ \sigma &= 0 \\ \mathrm{E}[\![\texttt{not}\ e]\!]\ \sigma &= \begin{cases} 1, & \text{if } \mathrm{E}[\![e]\!]\ \sigma = 0 \\ 0, & \text{otherwise} \end{cases} \\ \mathrm{E}[\![e_1\ \texttt{+}\ e_2]\!]\ \sigma &= \mathrm{E}[\![e_1]\!]\ \sigma\ +\ \mathrm{E}[\![e_1]\!]\ \sigma \end{aligned}$$

Rcp: Denotational Semantics for IL (3)

First attempt:

$$\begin{aligned} \mathrm{C}[\![\texttt{skip}]\!]\ \sigma &= \sigma \\ \mathrm{C}[\![x\ \texttt{:=}\ e]\!]\ \sigma &= [x \mapsto \mathrm{E}[\![e]\!]\ \sigma]\sigma \\ \mathrm{C}[\![c_1\ \texttt{;}\ c_2]\!]\ \sigma &= \mathrm{C}[\![c_2]\!]\ (\mathrm{C}[\![c_1]\!]\ \sigma) \end{aligned}$$

$$\mathrm{C}[\![\texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2]\!]\ \sigma = \begin{cases} \mathrm{C}[\![c_1]\!]\ \sigma, & \text{if } \mathrm{E}[\![e]\!]\ \sigma = 1 \\ \mathrm{C}[\![c_2]\!]\ \sigma, & \text{otherwise} \end{cases}$$

$$\mathrm{C}[\![\texttt{while}\ e\ \texttt{do}\ c]\!]\ \sigma = \mathrm{C}[\![\texttt{if}\ e\ \texttt{then}\ (c\ \texttt{;}\ \texttt{while}\ e\ \texttt{do}\ c)\ \texttt{else}\ \texttt{skip}]\!]\ \sigma$$

Rcp: Denotational Semantics for IL (4)

Intuition: Semantics of a command is a function mapping state (store) as it is ***prior*** to executing the command to resulting state ***after*** the command has been executed; i.e., a ***state transformer*** $(\Sigma \rightarrow \Sigma)$.

Any problem? Yes:

$\mathrm{C}[\![\texttt{while}\ e\ \texttt{do}\ c]\!]\ \sigma =$
$\quad \mathrm{C}[\![\texttt{if}\ e\ \texttt{then}\ (c\ \texttt{;}\ \texttt{while}\ e\ \texttt{do}\ c)\ \texttt{else}\ \texttt{skip}]\!]\ \sigma$

is ***not compositional*** and does not define a unique solution.
(However, it ***is*** a semantic equation that should hold.)

The Problem (1)

To see no unique solution, consider for example:

$$c_1 = \texttt{while x /= 1 do x := x - 2}$$

$$\mathrm{C}[\![c_1]\!]\ \sigma = \begin{cases} \mathrm{C}[\![c_1]\!]\ ([\texttt{x} \mapsto \sigma\ \texttt{x} - 2]\sigma), & \text{if } \sigma\ \texttt{x} \neq 1 \\ \sigma, & \text{otherwise} \end{cases} \quad \text{(A)}$$

Equation (A) is satisfied by $\mathrm{C}[\![c_1]\!] = f_{c_1}$ where:

$$f_{c_1}\ \sigma = \begin{cases} [\texttt{x} \mapsto 1]\sigma, & \text{if } \mathrm{odd}(\sigma\ \texttt{x}) \\ \sigma', & \text{if } \mathrm{even}(\sigma\ \texttt{x}),\ \sigma' \text{ arbitrary!} \end{cases} \quad \text{(S)}$$

The Problem (2)

Verify this (was homework).

Case $\sigma\ \texttt{x} = 1$:

$$\begin{aligned} \text{LHS (A)} &= \mathrm{C}[\![c_1]\!]\ \sigma \\ &= \{\ \mathrm{C}[\![c_1]\!] = f_{c_1}\ \} \\ &\quad f_{c_1}\ \sigma \\ &= \{\ \text{By (S)}, \mathrm{odd}(\sigma\ \texttt{x})\ \} \\ &\quad [\texttt{x} \mapsto 1]\sigma \\ &= \sigma \\ &= \text{RHS (A)} \end{aligned}$$

The Problem (3)

Case $\mathrm{odd}(\sigma\ \texttt{x}),\ \sigma\ \texttt{x} > 1$:
Note that then also $\mathrm{odd}(\sigma\ \texttt{x} - 2)$.

$$\begin{aligned} \text{LHS (A)} &= \mathrm{C}[\![c_1]\!]\ \sigma \\ &= f_{c_1}\ \sigma \\ &= \{\ \text{By (S)}, \mathrm{odd}(\sigma\ \texttt{x})\ \} \\ &\quad [\texttt{x} \mapsto 1]\sigma \\ &= [\texttt{x} \mapsto 1]([\texttt{x} \mapsto \sigma\ \texttt{x} - 2]\sigma) \\ &= \{\ \mathrm{odd}(\sigma\ \texttt{x} - 2), \text{By (S)}\ \} \\ &\quad f_{c_1}\ ([\texttt{x} \mapsto \sigma\ \texttt{x} - 2]\sigma) \\ &= \ldots \end{aligned}$$

The Problem (4)

$$
\begin{aligned}
&= \ldots \\
&= \mathcal{C}[\![c_1]\!]\ ([\mathbf{x} \mapsto \sigma\ \mathbf{x} - 2]\sigma) \\
&= \{\ \sigma\ \mathbf{x} \neq 1\ \} \\
&\quad \text{RHS (A)}
\end{aligned}
$$

The Problem (5)

Case $\text{even}(\sigma\ \mathbf{x}),\ \sigma\ \mathbf{x} > 1$:

$$
\begin{aligned}
\text{LHS (A)} &= \mathcal{C}[\![c_1]\!]\ \sigma \\
&= f_{c_1}\ \sigma \\
&= \{\ \text{By (S)},\ \text{even}(\sigma\ \mathbf{x})\ \} \\
&\quad \sigma' \\
&= \{\ \text{even}(\sigma\ \mathbf{x} - 2),\ \text{By (S)}\ \} \\
&= f_{c_1}\ ([\mathbf{x} \mapsto \sigma\ \mathbf{x} - 2]\sigma) \\
&= \mathcal{C}[\![c_1]\!]\ ([\mathbf{x} \mapsto \sigma\ \mathbf{x} - 2]\sigma) \\
&= \{\ \sigma\ \mathbf{x} \neq 1\ \} \\
&\quad \text{RHS (A)}
\end{aligned}
$$

Solution: Fixed Point Semantics (1)

How can we proceed?

Clue: $f_{c_1} = \mathcal{C}[\![c_1]\!]$ occurs in both the LHS and RHS of (A). The desired semantic function is the ***fixed point*** of the equation!

New attempt:

$$
\mathcal{C}[\![\texttt{while}\ e\ \texttt{do}\ c]\!] = \\
\text{fix}_{\Sigma \to \Sigma}\left(\lambda f.\lambda\sigma.\begin{cases}\sigma, & \text{if } \mathcal{E}[\![e]\!]\ \sigma = 0 \\ f\ (\mathcal{C}[\![c]\!]\ \sigma), & \text{otherwise}\end{cases}\right)
$$

Solution: Fixed Point Semantics (2)

(Might be easier to see if we allow a ***recursive*** formulation where the fixed point is implicit:

$$
\mathcal{C}[\![\texttt{while}\ e\ \texttt{do}\ c]\!] = f
$$

where

$$
f\ \sigma = \begin{cases}\sigma, & \text{if } \mathcal{E}[\![e]\!]\ \sigma = 0 \\ f\ (\mathcal{C}[\![c]\!]\ \sigma), & \text{otherwise}\end{cases}
$$

However, we stick to an explicit fixed point formulation to make the semantics clear.)

Existence and Uniqueness? (1)

Our definition of $\mathcal{C}[\![\cdot]\!]$ is now ***compositional***!

But:

- Does this fixed point exist?
- Is it unique if it does exist?

We should be suspicious! Consider e.g.:

$$\mathcal{C}[\![\texttt{while true do (x := x + 1)}]\!]\ \{\texttt{x} \mapsto 0\}$$

What could the final value of **x** possibly be?

10? 1000? ∞?

Existence and Uniqueness? (2)

More generally, consider the following "recursive" definitions:

$$\begin{aligned} f_1\ n &= (f_1\ n) + 1 \qquad (1) \\ f_2\ n &= f_2\ n \qquad (2) \end{aligned}$$

- ***No*** $f_1 \in \mathbb{N} \to \mathbb{N}$ satisfies (1).
- ***All*** $f_2 \in \mathbb{N} \to \mathbb{N}$ satisfies (2).

So, if we are considering functions defined on ***sets***, fixed points need not exist, and, if they do, they need not be unique!

Denotation for Non-termination (1)

Idea:

- Let $\bot$ ("bottom") denote non-termination (divergence) or error.
- For a set A such that $\bot \notin A$, let $A_\bot = A \cup \{\bot\}$.
- For a function $f : A \to B_\bot$, let

$$f_{\bot\!\!\!\bot}\ x = \begin{cases} \bot, & \text{if } x = \bot \\ f\ x, & \text{otherwise} \end{cases}$$

(Called "source lifting"; note: $f_{\bot\!\!\!\bot} : A_\bot \to B_\bot$)

Denotation for Non-termination (2)

- A function f is ***strict*** iff $f\ \bot = \bot$.
- Source lifting yields a strict function. Intuitively, it ensures propagation of errors.

We can now find a function satisfying (1):

$$\begin{aligned} f_1 &: \mathbb{N}_\bot \to \mathbb{N}_\bot \\ f_1\ x &= \bot \end{aligned}$$

f_1 satisfies (1) because $+$ is strict (i.e., in this case, $\bot + 1 = \bot$).

Denotation for Non-termination (3)

- Thus, by considering a ***mathematically richer structure*** than plain sets, we could find a solution to at least one fixed point equation that did not have a solution in plain set theory.
- This is a key idea of ***Domain Theory***.
- However, even if we move to such a richer setting, we still don't know:
  - Does a fixed point equation ***always*** have a solution?
  - Are solutions ***unique*** if they exists?

Semantics for Commands Revisited

But first, let us refine the meaning of commands:

$$\mathrm{C}[\![\cdot]\!] : c \to (\Sigma \to \Sigma_\perp)$$

Now we can find a meaning for e.g. an infinite loop:

$$\mathrm{C}[\![\texttt{while true do skip}]\!] \;=\; \lambda\sigma.\,\perp$$

But we have to refine the meaning of sequencing:

$$\mathrm{C}[\![c_1\ \texttt{;}\ c_2]\!]\ \sigma \;=\; (\mathrm{C}[\![c_2]\!]_{\perp\!\!\perp})\ (\mathrm{C}[\![c_1]\!]\ \sigma)$$

Domains and Continuous Functions (1)

- A ***domain*** D is a set with
  - a ***partial order*** $\sqsubseteq$
  - a ***least element*** $\perp$

  such that every ***chain*** of elements $x_i \in D$, $x_0 \sqsubseteq x_1 \sqsubseteq \ldots$, has a ***limit*** in D, i.e., a ***least upper bound***, denoted $\bigsqcup_{i=0}^{\infty} x_i$.
- $\sqsubseteq$ is an ***information ordering***: read $x \sqsubseteq y$ as "x is less informative than y".

Domains and Continuous Functions (2)

- If D satisfies all conditions for being a domain, except that it lacks a smallest element, then it is called a ***predomain***.
- A function f is said to be ***continuous*** if it ***preserves limits*** of chains:

$$f\left(\bigsqcup_{i=0}^{\infty} x_i\right) \;=\; \bigsqcup_{i=0}^{\infty} f\ x_i$$

where x_i is a chain.

Domains and Continuous Functions (3)

- Any ***function space*** from a (pre)domain to a domain is a domain with least element $\lambda x.\ \bot$; i.e., the everywhere undefined function.

The Least Fixed Point Theorem

If D is a domain and $f : D \to D$ is a continuous function, then

$$x = \bigsqcup_{n=0}^{\infty} f^n \bot$$

is the least fixed point of f; i.e., $f\ x = x$, and for all y such that $f\ y = y$, it is the case that $x \sqsubseteq y$.

The Meaning of fix_D

Thus we take the meaning of fix_D to be as given by the Least Fixed Point Theorem.

As long as D is a domain and $f : D \to D$ is a continuous function, then the fixed point x

$$x = \mathrm{fix}_D\ f$$

exists and is unique.

Exercise (1)

Consider the following definition of the factorial function:

$$\begin{aligned} f &: (\mathbb{N} \to \mathbb{N}_\bot) \to (\mathbb{N} \to \mathbb{N}_\bot) \\ f &= \lambda g.\lambda n.\mathrm{if}\ n = 0\ \mathrm{then}\ 1\ \mathrm{else}\ n \times g\ (n-1) \\ fac &= \mathrm{fix}_{\mathbb{N}\to\mathbb{N}_\bot}\ f \end{aligned}$$

Note: $\mathbb{N}$ is a predomain and $\mathbb{N}_\bot$ is a domain. Thus $\mathbb{N} \to \mathbb{N}_\bot$ is a domain.

Calculate $f^n \bot$ for $n = 0, 1, 2, 3$.

Exercise (2)

Note how f^n becomes a better and better approximation of the factorial function as n increases.

Thus each successive approximation is more ***informative*** than the previous one (information ordering).

Thus it seems plausible that the series converges to the factorial function.

And in fact, because f is continuous, it does.

Semantics of `while` Revisited (1)

- It can be shown that Σ

$$\Sigma = x \to \mathbb{N}$$

  is a predomain.
- Thus $\Sigma_\bot$ and $\Sigma \to \Sigma_\bot$ are both domains.
- Furthermore, it can be shown that all functions g

$$g \in (\Sigma \to \Sigma_\bot) \to (\Sigma \to \Sigma_\bot)$$

  are continuous.

Semantics of `while` Revisited (2)

Thus, we can define:

$$\mathrm{C}[\![\texttt{while}\ e\ \texttt{do}\ c]\!] = \mathrm{fix}_{\Sigma\to\Sigma_\bot}\ g$$

where

$$g : (\Sigma \to \Sigma_\bot) \to (\Sigma \to \Sigma_\bot)$$

$$g = \lambda f.\lambda\sigma. \begin{cases} \sigma, & \text{if } \mathrm{E}[\![e]\!]\ \sigma = 0 \\ f_{\bot\!\!\!\bot}\ (\mathrm{C}[\![c]\!]\ \sigma), & \text{otherwise} \end{cases}$$

in the knowledge that the fixed point $\mathrm{fix}_{\Sigma\to\Sigma_\bot}\ g$ exists and is the smallest fixed point of g.

Exercises

- Calculate $g^n\ \bot$ for g from the previous slide for a few n from 0 and upwards until you have convinced yourself that you get a better and better approximation of the semantic function for a `while`-loop (i.e., that each successive approximation can handle one more iteration).

Exercises

- Suppose we wish to add a C/Java-like post increment operator to the expression fragment of our language:

  x++

  The value of the expression is the current value of the variable, but as a side effect the variable is also incremented by one.

  How would the semantic definitions have to be restructured to accommodate this addition? In particular, what is a suitable type for the semantic function $\mathrm{E}[\![\cdot]\!]$?