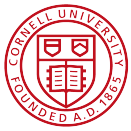
CS 4110 – Programming Languages and Logics
Lecture #8: Denotational Semantics Examples



Last time we defined the denotational semantics of IMP:

$$
\begin{aligned}
\mathcal{A}[\![n]\!] &= \{(\sigma, n)\} \\
\mathcal{A}[\![x]\!] &= \{(\sigma, \sigma(x))\} \\
\mathcal{A}[\![a_1 + a_2]\!] &= \{(\sigma, n) \mid (\sigma, n_1) \in \mathcal{A}[\![a_1]\!] \wedge (\sigma, n_2) \in \mathcal{A}[\![a_2]\!] \wedge n = n_1 + n_2\}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{B}[\![\textbf{true}]\!] &= \{(\sigma, \textbf{true})\} \\
\mathcal{B}[\![\textbf{false}]\!] &= \{(\sigma, \textbf{false})\} \\
\mathcal{B}[\![a_1 < a_2]\!] &= \{(\sigma, \textbf{true}) \mid (\sigma, n_1) \in \mathcal{A}[\![a_1]\!] \wedge (\sigma, n_2) \in \mathcal{A}[\![a_2]\!] \wedge n_1 < n_2\} \cup \\
&\quad \{(\sigma, \textbf{false}) \mid (\sigma, n_1) \in \mathcal{A}[\![a_1]\!] \wedge (\sigma, n_2) \in \mathcal{A}[\![a_2]\!] \wedge n_1 \geq n_2\} \\
\mathcal{C}[\![\textbf{skip}]\!] &= \{(\sigma, \sigma)\} \\
\mathcal{C}[\![x := a]\!] &= \{(\sigma, \sigma[x \mapsto n]) \mid (\sigma, n) \in \mathcal{A}[\![a]\!]\} \\
\mathcal{C}[\![c_1; c_2]\!] &= \{(\sigma, \sigma') \mid \exists \sigma''.\ ((\sigma, \sigma'') \in \mathcal{C}[\![c_1]\!] \wedge (\sigma'', \sigma') \in \mathcal{C}[\![c_2]\!])\} \\
\mathcal{C}[\![\textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2]\!] &= \{(\sigma, \sigma') \mid (\sigma, \textbf{true}) \in \mathcal{B}[\![b]\!] \wedge (\sigma, \sigma') \in \mathcal{C}[\![c_1]\!]\} \cup \\
&\quad \{(\sigma, \sigma') \mid (\sigma, \textbf{false}) \in \mathcal{B}[\![b]\!] \wedge (\sigma, \sigma') \in \mathcal{C}[\![c_2]\!]\} \\
\mathcal{C}[\![\textbf{while } b \textbf{ do } c]\!] &= \mathit{fix}(F) \\
\text{where } F(f) &= \{(\sigma, \sigma) \mid (\sigma, \textbf{false}) \in \mathcal{B}[\![b]\!]\} \cup \\
&\quad \{(\sigma, \sigma') \mid (\sigma, \textbf{true}) \in \mathcal{B}[\![b]\!] \wedge \exists \sigma', \sigma''.\ (\sigma, \sigma') \in \mathcal{C}[\![c]\!] \wedge (\sigma', \sigma'') \in f\}
\end{aligned}
$$

In this lecture we'll prove Kleene's fixpoint theorem, which shows that the fixed point used to define the semantics of **while** commands exists, and work through examples of reasoning using the denotational semantics.

1 Kleene's Fixpoint Theorem

Definition (Scott Continuity)**.** A function F from U to U is said to be *Scott-continuous* if for every chain $X_1 \subseteq X_2 \subseteq \ldots$ we have $F(\bigcup_i X_i) = \bigcup_i F(X_i)$.

It is not hard to show that if F is Scott continuous, then it is also monotonic—that is, $X \subseteq Y$ implies $F(X) \subseteq F(Y)$. The proof of this fact is left as an exercise.

Theorem (Kleene Fixpoint)**.** *Let F be a Scott-continuous function. The least fixed point of F is $\bigcup_i F^i(\emptyset)$.*

Proof. Let $X = \bigcup_i F^i(\emptyset)$.

First, we will prove that X is a fixed point of F—that is, $F(X) = X$. We calculate as follows:

$$\begin{array}{rcll} F(X) & = & F(\bigcup_i F^i(\emptyset)) & \text{By definition of } X \\ & = & \bigcup_i F(F^i(\emptyset)) & \text{By Scott continuity} \\ & = & \bigcup_i F^{i+1}(\emptyset) & \\ & = & \emptyset \cup \bigcup_i F^{i+1}(\emptyset) & \\ & = & F^0(\emptyset) \cup \bigcup_i F^{i+1}(\emptyset) & \\ & = & \bigcup_i F^i(\emptyset) & \\ & = & X & \end{array}$$

Second, we will show that X is the least fixed point of F. Suppose that Y is some other arbitrary fixed point of F.

By induction, we can easily show that $F^i(\emptyset) \subseteq Y$ for all i. For the base case, i is 0 and we trivially have $F^0(\emptyset) = \emptyset \subseteq Y$. For the inductive case, we assume that $F^i(\emptyset) \subseteq Y$ and prove that $F^{i+1}(\emptyset) \subseteq Y$. By our inductive hypothesis and the fact that F is monotone, we have that $F(F^i(\emptyset)) \subseteq F(Y)$. As Y is a fixed point we also have $F(Y) = Y$ and so $F^{i+1}(\emptyset) \subseteq Y$.

Then, since every element of the chain

$$F^0(\emptyset) \subseteq F^1(\emptyset) \subseteq \ldots$$

is a subset of Y immediately we have that their union, $X = \bigcup_i F^i(\emptyset) \subseteq Y$. Hence, X is the least (with respect to $\subseteq$) fixed point of F. □

2 Reasoning

One of the key advantages of using denotational semantics compared to operational semantics is that proofs of equivalence can be carried out directly by simply calculating the denotations of programs and then arguing that they are identical. This is in contrast to operational techniques, where one must reason explicitly about low-level transitions and derivations involving ad hoc abstract machines.

As an example, to show that **skip**; c and c; **skip** are equivalent, we can calcuate as follows,

$$\begin{array}{rcl} \mathcal{C}[\![\textbf{skip}; c]\!] & = & \{(\sigma, \sigma'') \mid \exists \sigma'.(\sigma, \sigma') \in \mathcal{C}[\![\textbf{skip}]\!] \wedge (\sigma', \sigma'') \in \mathcal{C}[\![c]\!]\} \\ & = & \{(\sigma, \sigma'') \mid (\sigma, \sigma'') \in \mathcal{C}[\![c]\!]\} \\ & = & \{(\sigma, \sigma'') \mid \exists \sigma'.(\sigma, \sigma') \in \mathcal{C}[\![c]\!] \wedge (\sigma', \sigma'') \in \mathcal{C}[\![\textbf{skip}]\!]\} \\ & = & \mathcal{C}[\![c; \textbf{skip}]\!] \end{array}$$

using standard facts about partial functions, relations, and sets as convenient in the proof itself.

Next, consider the command $\mathcal{C}[\![\textbf{while false do } c]\!]$. By the definition of the denotational semantics, this is equal to $\mathit{fix}(F)$ where

$$\begin{array}{rcl} F(f) & = & \{(\sigma, \sigma) \mid (\sigma, \textbf{false}) \in \mathcal{B}[\![b]\!]\} \cup \\ & = & \{(\sigma, \sigma'') \mid (\sigma, \textbf{true}) \in \mathcal{B}[\![b]\!] \wedge (\sigma, \sigma') \in \mathcal{C}[\![c]\!] \wedge (\sigma', \sigma'') \in f\} \end{array}$$

By the Kleene fixpoint theorem we have that $\mathit{fix}(F) = \bigcup_i F^i(\emptyset)$. It is straightforward to show by induction that since the guard is **false**, for all i we have $F^i(\emptyset) = \{(\sigma, \sigma)\}$. It follows that $\mathit{fix}(F) = \{(\sigma, \sigma)\}$, which is just $\mathcal{C}[\![\textbf{skip}]\!]$.

As an exercise, calcuate $\mathcal{C}[\![\textbf{while true do skip}]\!]$ using the same technique.