CS611 Lecture 09 **Denotational Semantics of While Loops** 17 September 2003

Scribe: Benyah Shaparenko Lecturer: Radu Rugina

1 Introduction

Recall that denotational semantics consists of purely mathematical functions that describe what programs do. For the language IMP, these are functions in $\Sigma \to \mathbb{Z}$ for expressions, and functions in $\Sigma \to \Sigma_\perp$ for commands. In the last lecture, we wrote the denotational semantics for all of IMP except for the **while** loop, so this lecture will cover the denotational semantics of the **while** loop.

2 Denotational Semantics of While Loops

The denotational semantics has the form $\mathcal{C}[\![w]\!] : \Sigma \to \Sigma_\perp$ where $w \equiv$ **while** b **do** c. The denotational semantics of the **while** loop maps a state to either another state or to $\perp$. Think of the equivalent command to **while** b **do** c that we previously used, namely, **if** b **then** $c; w$ **else skip**. Then we can write out an equation for $\mathcal{C}[\![w]\!]$ as follows (with $\sigma \in \Sigma$):

$$\begin{aligned} \mathcal{C}[\![w]\!] &= \{(\sigma,\sigma) \mid \neg\mathcal{B}[\![b]\!]\sigma\} && (1)\\ &\cup \{(\sigma,\perp) \mid \mathcal{B}[\![b]\!]\sigma \wedge \mathcal{C}[\![c]\!]\sigma = \perp\} && (2)\\ &\cup \{(\sigma, \mathcal{C}[\![w]\!](\mathcal{C}[\![c]\!]\sigma) \mid \mathcal{B}[\![b]\!]\sigma \wedge \mathcal{C}[\![c]\!]\sigma \neq \perp\} && (3) \end{aligned}$$

Notice that the above equation is not a definition for the **while** loop, just a recursive equation describing $\mathcal{C}[\![w]\!]$. This means that we aren't satisfied with the above equation, but need to solve this equation for the actual denotational semantics of the **while** loop. To begin, translate the equation into inductively defined rules, namely,

$$\frac{}{(\sigma,\sigma)}\ \neg\mathcal{B}[\![b]\!]\sigma \qquad (4)$$

$$\frac{}{(\sigma,\perp)}\ \mathcal{B}[\![b]\!]\sigma \wedge \mathcal{C}[\![c]\!]\sigma = \perp \qquad (5)$$

$$\frac{(\mathcal{C}[\![c]\!]\sigma, \sigma')}{(\sigma,\sigma')}\ \mathcal{B}[\![b]\!]\sigma \wedge \mathcal{C}[\![c]\!]\sigma \neq \perp \qquad (6)$$

Let this set of rules be R. This is where the $\hat{R}$ operator comes in. The denotation of w is an inductively defined set, so one can build this set using the $\hat{R}$ operator as follows:

$$\mathcal{C}[\![w]\!] = \bigcup_{n\in\mathbb{N}} \hat{R}^n(\phi)$$

where $\hat{R}(X) = \{y \mid X' \subseteq X.$ X' / y$\}$, as was defined before. In the inference rules above, we start with ϕ, and inductively apply the rule operator $\hat{R}$. In the first step,

$$\hat{R}(\phi) = (1) \cup (2)$$

In the second step, we get the new sets to add to $\hat{R}(\phi)$ by applying the first (4) and second (5) axioms, respectively, to the third (6) rule. Doing this yields:

$$\begin{aligned} \hat{R}^2(\phi) &= (1)\cup(2)\\ &\cup \{(\sigma, \mathcal{C}[\![c]\!]\sigma) \mid \mathcal{B}[\![b]\!]\sigma \wedge \mathcal{C}[\![c]\!]\sigma \neq \perp \wedge \neg\mathcal{B}[\![b]\!](\mathcal{C}[\![c]\!]\sigma)\}\\ &\cup \{(\sigma,\perp) \mid \mathcal{B}[\![b]\!]\sigma \wedge \mathcal{C}[\![c]\!]\sigma \neq \perp \wedge \mathcal{B}[\![b]\!](\mathcal{C}[\![c]\!]\sigma) \wedge \mathcal{C}[\![c]\!](\mathcal{C}[\![c]\!]\sigma) = \perp\} \end{aligned}$$

Intuitively, think of each iteration $\hat{R}^i(\phi)$ as approximating the final denotational semantics. The final infinite union is the denotational semantics for the **while** loop. Also, notice that each step above is only a partial function. We can alternatively make it a total function, by explicitly saying that all states that don't have a value mapped to them map to $\perp$. Formally,

$$\begin{array}{rcl} \phi & \rightarrow & \lambda\sigma.\perp = \perp_{\Sigma\rightarrow\Sigma_\perp} \\ \hat{R}(\phi) & \rightarrow & \{(\sigma,\sigma) \mid \neg\mathcal{B}[\![b]\!]\sigma\} \cup \\ & & \{(\sigma,\perp) \mid \mathcal{B}[\![b]\!]\sigma\} \end{array}$$

This means that for every state σ where no side conditions are met, then we must map σ to $\perp$.

We can also see from the above applications $\hat{R}$ that for i, $\hat{R}^i(\phi)$ is the approximation of the denotational semantics after we execute the **while** loop *at most* i times; if the loop runs for more than i iterations, $\hat{R}^i(\phi)$ approximates the final state by $\perp$. Another way of thinking of this is that when we move from $\hat{R}^n(\phi)$ to $\hat{R}^{n+1}(\phi)$, we are gaining information about iteration $n+1$ of the **while** loop.

Note: The second axiom (5) can be removed since we defined that every case not explicitly stated will "fall through" to $\perp$. So, it is unnecessary to explicitly say that something evaluates to $\perp$. The reason that we write explicitly that (5) evaluates to $\perp$ is so the rules follow more naturally the pattern of execution of the **while** loop.

3 The Higher Order Function Γ

Instead of using $\hat{R}$, we will actually use Γ, a higher order function similar in behavior to $\hat{R}$, and defined as:

$$\begin{array}{rcl} \Gamma & : & (\Sigma\rightarrow\Sigma_\perp)\rightarrow(\Sigma\rightarrow\Sigma_\perp) \\ \Gamma(\gamma) & = & \{(\sigma,\sigma) \mid \neg\mathcal{B}[\![b]\!]\sigma\} \\ & \cup & \{(\sigma,\perp) \mid \mathcal{B}[\![b]\!]\sigma \wedge \mathcal{C}[\![c]\!]\sigma = \perp\} \\ & \cup & \{(\sigma,\gamma(\mathcal{C}[\![c]\!]\sigma) \mid \mathcal{B}[\![b]\!]\sigma \wedge \mathcal{C}[\![c]\!]\sigma \neq \perp\} \end{array}$$

Essentially, we have the following correspondence between $\hat{R}$ and Γ:

$$\begin{array}{rcl} \hat{R}(\phi) & \rightarrow & \Gamma(\perp_{\Sigma\rightarrow\Sigma_\perp}) \\ \hat{R}^2(\phi) & \rightarrow & \Gamma^2(\perp_{\Sigma\rightarrow\Sigma_\perp}) \\ & \text{etc.} & \end{array}$$

Recall that an inductively defined set is defined as $\bigcup_{n\in\mathbb{N}} \hat{R}^n(\phi)$, which is the least fixed point of $\hat{R}$. We want to follows this definition and say that the denotational semantics of the while loop is:

$$\mathcal{C}[\![w]\!] = \bigsqcup_{n\in\mathbb{N}} \Gamma^n(\perp_{\Sigma\rightarrow\Sigma_\perp})$$

where $\bigsqcup$ is a union-like operator for functions. Alternatively, we can define the loop semantics as:

$$\mathcal{C}[\![w]\!] = \mathit{fix}(\Gamma)$$

where $\mathit{fix} : (D \rightarrow D) \rightarrow D$ and $\mathit{fix}(f)$ is the least fixed point of f. Note: $d = \mathit{fix}(f)$ means that $f(d) = d$, which implies $(\forall d'.\ f(d') = d' \Rightarrow d \sqsubseteq d')$. In our example, the domain D is $\Sigma \rightarrow \Sigma_\perp$.

Recall that we had $\hat{R}^n(\phi) \subseteq \hat{R}^{n+1}(\phi)$. Following this relation, we should have a relation of the form: $\Gamma^n(\perp) \subseteq \Gamma^{n+1}(\perp)$, where $\Gamma^n(\perp)$ and $\Gamma^{n+1}(\perp)$ are functions, and $\sqsubseteq$ is an operator that compares functions.

At this point, we wonder how we define the operator $\bigsqcup$ and the relation $\sqsubseteq$ over function spaces. For this, we will discuss the theory of partial orders, and, in particular, the ordering over function spaces.

4 Partial Orders

4.1 Definition of Partial Orders

Partial orders are also known as posets. A partial order is a pair $(S, \sqsubseteq)$ of a set S and relation $\sqsubseteq$ s.t. $\sqsubseteq$ is:

1. reflexive: $x \sqsubseteq x$
2. antisymmetric: $x \sqsubseteq y,\ y \sqsubseteq x \Rightarrow x = y$
3. transitive: $x \sqsubseteq y,\ y \sqsubseteq z \Rightarrow x \sqsubseteq z$

This structure is called a partial order because it is possible that there are incomparable elements, i.e., x and y for which neither $x \sqsubseteq y$ nor $y \sqsubseteq x$.

4.2 Some Examples of Partial Orders

- $(\mathbb{N}, \leq)$ or $(\mathbb{Z}, \leq)$, which are numbers and usual $\leq$
- $(\mathbb{N}, =)$ is a partial order. In general, for any set s, $(S, =)$ is a partial order. This is called a *discrete* or *flat* order.
- $(\mathcal{P}(S), \subseteq)$, $(\mathcal{P}(S), \supseteq)$, the powerset of S and inclusion relation.
- if $(S, \sqsubseteq)$ is a poset, then $(S, \sqsupseteq)$ is also a poset where $x \sqsupseteq y$ iff $y \sqsubseteq x$.
- $(\mathbb{N}, |)$, where $a \sqsubseteq b$ iff a divides b $(\exists k \in \mathbb{N}.\ b = ka)$.
- Suppose $(B, \sqsubseteq_B)$ is a poset. Then define $\sqsubseteq$ over $A \to B$:
  $f \sqsubseteq g$ iff $\forall x \in A.\ f(x) \sqsubseteq_B g(x)$.
  This is called pointwise ordering. $(A \to B, \sqsubseteq)$ is a poset.