Small-step Operational Semantics

Lecture 2 Thursday, January 25, 2018

1 Small-step operational semantics

At this point we have defined the syntax of our simple arithmetic language. We have some informal, intuitive notion of what programs in this language mean. For example, the program $7 + (4 \times 2)$ should equal 15, and the program $\mathsf{foo} := 6 + 1; 2 \times 3 \times \mathsf{foo}$ should equal 42.

We would like now to define formal semantics for this language.

Operational semantics describe how a program would execute on an abstract machine. A *small-step operational semantics* describe how such an execution in terms of successive reductions of an expression, until we reach a number, which represents the result of the computation.

The state of the abstract machine is usually referred to as a configuration, and for our language it must include two pieces of information:

- a store (also called an *environment* or *state*), which assigns integer values to variables. During program execution, we will refer to the store to determine the values associated with variables, and also update the store to reflect assignment of new values to variables.
- the expression left to evaluate.

Thus, the domain of stores is functions from **Var** to **Int** (written **Var** $\to$ **Int**), and the domain of configurations is pairs of expressions and stores.

$$\textbf{Config} = \textbf{Exp} \times \textbf{Store}$$
$$\textbf{Store} = \textbf{Var} \to \textbf{Int}$$

We will denote configurations using angle brackets. For instance, $\langle(\mathsf{foo} + 2) \times (\mathsf{bar} + 1), \sigma\rangle$, where σ is a store and $(\mathsf{foo} + 2) \times (\mathsf{bar} + 1)$ is an expression that uses two variables, foo and bar.

The small-step operational semantics for our language is a relation $\longrightarrow \subseteq$ **Config** $\times$ **Config** that describes how one configuration transitions to a new configuration.[1] That is, the relation $\longrightarrow$ shows us how to evaluate programs, one step at a time. We use infix notation for the relation $\longrightarrow$. That is, given any two configurations $\langle e_1, \sigma_1\rangle$ and $\langle e_2, \sigma_2\rangle$, if $(\langle e_1, \sigma_1\rangle, \langle e_2, \sigma_2\rangle)$ is in the relation $\longrightarrow$, then we write $\langle e_1, \sigma_1\rangle \longrightarrow \langle e_2, \sigma_2\rangle$.

For example, we have $\langle(4 + 2) \times \mathsf{y}, \sigma\rangle \longrightarrow \langle 6 \times \mathsf{y}, \sigma\rangle$. That is, we can evaluate the configuration $\langle(4 + 2) \times \mathsf{y}, \sigma\rangle$ by one step, to get the configuration $\langle 6 \times \mathsf{y}, \sigma\rangle$.

Now defining the semantics of the language boils down to defining the relation $\longrightarrow$ that describes the transitions between machine configurations.

One issue here is that the domain of integers is infinite, and so is the domain of expressions. Therefore, there is an infinite number of possible machine configurations, and an infinite number of possible one-step transitions. We need to use a finite description for the infinite set of transitions.

We can compactly describe the transition function $\longrightarrow$ using inference rules:

$$\textsc{Var}\ \frac{}{\langle x, \sigma\rangle \longrightarrow \langle n, \sigma\rangle}\ \text{where } n = \sigma(x)$$

[1] If you are not familiar and comfortable with the concept of *relations*, please go to the course website and find the readings for background material.

$$\textsc{LAdd}\ \frac{\langle e_1, \sigma\rangle \longrightarrow \langle e_1', \sigma'\rangle}{\langle e_1 + e_2, \sigma\rangle \longrightarrow \langle e_1' + e_2, \sigma'\rangle} \qquad \textsc{RAdd}\ \frac{\langle e_2, \sigma\rangle \longrightarrow \langle e_2', \sigma'\rangle}{\langle n + e_2, \sigma\rangle \longrightarrow \langle n + e_2', \sigma'\rangle}$$

$$\textsc{Add}\ \frac{}{\langle n + m, \sigma\rangle \longrightarrow \langle p, \sigma\rangle}\ \text{where } p \text{ is the sum of } n \text{ and } m$$

$$\textsc{LMul}\ \frac{\langle e_1, \sigma\rangle \longrightarrow \langle e_1', \sigma'\rangle}{\langle e_1 \times e_2, \sigma\rangle \longrightarrow \langle e_1' \times e_2, \sigma'\rangle} \qquad \textsc{RMul}\ \frac{\langle e_2, \sigma\rangle \longrightarrow \langle e_2', \sigma'\rangle}{\langle n \times e_2, \sigma\rangle \longrightarrow \langle n \times e_2', \sigma'\rangle}$$

$$\textsc{Mul}\ \frac{}{\langle n \times m, \sigma\rangle \longrightarrow \langle p, \sigma\rangle}\ \text{where } p \text{ is the product of } n \text{ and } m$$

$$\textsc{Asg1}\ \frac{\langle e_1, \sigma\rangle \longrightarrow \langle e_1', \sigma'\rangle}{\langle x := e_1; e_2, \sigma\rangle \longrightarrow \langle x := e_1'; e_2, \sigma'\rangle} \qquad \textsc{Asg}\ \frac{}{\langle x := n; e_2, \sigma\rangle \longrightarrow \langle e_2, \sigma[x \mapsto n]\rangle}$$

The meaning of an inference rule is that if the fact above the line holds and the side conditions also hold, then the fact below the line holds. The fact(s) above the line are called premises; the fact below the line is called the conclusion. The rules without premises are axioms; and the rules with premises are inductive rules.

Also, we use the notation $\sigma[x \mapsto n]$ for a store that maps the variable x to integer n, and maps every other variable to whatever σ maps it to. More explicitly, if f is the function $\sigma[x \mapsto n]$, then we have

$$f(y) = \begin{cases} n & \text{if } y = x \\ \sigma(y) & \text{otherwise} \end{cases}$$

2 Using the Semantic Rules

Let's see how we can use these rules. Suppose we want to evaluate expression $(\mathsf{foo} + 2) \times (\mathsf{bar} + 1)$ in a store σ where $\sigma(\mathsf{foo}) = 4$ and $\sigma(\mathsf{bar}) = 3$. That is, we want to find the transition for configuration $\langle(\mathsf{foo} + 2) \times (\mathsf{bar} + 1), \sigma\rangle$. For this, we look for a rule with this form of a configuration in the conclusion. By inspecting the rules, we find that the only matching rule is LMul, where $e_1 = \mathsf{foo} + 2$, $e_2 = \mathsf{bar} + 1$, but e_1' is not yet known. We can *instantiate* the rule LMul, replacing the metavariables e_1 and e_2 with appropriate expressions.

$$\textsc{LMul}\ \frac{\langle \mathsf{foo} + 2, \sigma\rangle \longrightarrow \langle e_1', \sigma\rangle}{\langle(\mathsf{foo} + 2) \times (\mathsf{bar} + 1), \sigma\rangle \longrightarrow \langle e_1' \times (\mathsf{bar} + 1), \sigma\rangle}$$

Now we need to show that the premise actually holds and find out what e_1' is. We look for a rule whose conclusion matches $\langle \mathsf{foo} + 2, \sigma\rangle \longrightarrow \langle e_1', \sigma\rangle$. We find that LAdd is the only matching rule:

$$\textsc{LAdd}\ \frac{\langle \mathsf{foo}, \sigma\rangle \longrightarrow \langle e_1'', \sigma\rangle}{\langle \mathsf{foo} + 2, \sigma\rangle \longrightarrow \langle e_1'' + 2, \sigma\rangle}$$

where $e_1' = e_1'' + 2$. We repeat this reasoning for $\langle \mathsf{foo}, \sigma\rangle \longrightarrow \langle e_1'', \sigma\rangle$, and we find that the only applicable rule is the axiom Var:

$$\textsc{Var}\ \frac{}{\langle \mathsf{foo}, \sigma\rangle \longrightarrow \langle 4, \sigma\rangle}$$

because we have $\sigma(\mathsf{foo}) = 4$. Since this is an axiom and has no premises, there is nothing left to prove. Hence, $e'' = 4$ and $e'_1 = 4 + 2$. We can put together the above pieces and build the following proof:

$$\textsc{LMul}\ \frac{\textsc{LAdd}\ \dfrac{\textsc{Var}\ \dfrac{}{\langle \mathsf{foo}, \sigma\rangle \longrightarrow \langle 4, \sigma\rangle}}{\langle \mathsf{foo} + 2, \sigma\rangle \longrightarrow \langle 4 + 2, \sigma\rangle}}{\langle (\mathsf{foo} + 2) \times (\mathsf{bar} + 1), \sigma\rangle \longrightarrow \langle (4 + 2) \times (\mathsf{bar} + 1), \sigma\rangle}$$

This proves that, given our inference rules, the one-step transition $\langle (\mathsf{foo} + 2) \times (\mathsf{bar} + 1), \sigma\rangle \longrightarrow \langle (4 + 2) \times (\mathsf{bar} + 1), \sigma\rangle$ is possible. The above proof structure is called a "proof tree" or "derivation". It is important to keep in mind that proof trees must be finite for the conclusion to be valid.

We can use a similar reasoning to find out the next evaluation step:

$$\textsc{LMul}\ \frac{\textsc{Add}\ \dfrac{}{\langle 4 + 2, \sigma\rangle \longrightarrow \langle 6, \sigma\rangle}}{\langle (4 + 2) \times (\mathsf{bar} + 1), \sigma\rangle \longrightarrow \langle 6 \times (\mathsf{bar} + 1), \sigma\rangle}$$

And we can continue this process. At the end, we can put together all of these transitions, to get a view of the entire computation:

$$\begin{aligned}\langle (\mathsf{foo} + 2) \times (\mathsf{bar} + 1), \sigma\rangle &\longrightarrow \langle (4 + 2) \times (\mathsf{bar} + 1), \sigma\rangle \\ &\longrightarrow \langle 6 \times (\mathsf{bar} + 1), \sigma\rangle \\ &\longrightarrow \langle 6 \times (3 + 1), \sigma\rangle \\ &\longrightarrow \langle 6 \times 4, \sigma\rangle \\ &\longrightarrow \langle 24, \sigma\rangle\end{aligned}$$

The result of the computation is a number, 24. The machine configuration that contains the final result is the point where the evaluation stops; they are called *final configurations*. For our language of expressions, the final configurations are of the form $\langle n, \sigma\rangle$ where n is a number and σ is a store.

We write $\longrightarrow^*$ for the reflexive transitive closure of the relation $\longrightarrow$. That is, if $\langle e, \sigma\rangle \longrightarrow^* \langle e', \sigma'\rangle$, then using zero or more steps, we can evaluate the configuration $\langle e, \sigma\rangle$ to the configuration $\langle e', \sigma'\rangle$. Thus, we can write

$$\langle (\mathsf{foo} + 2) \times (\mathsf{bar} + 1), \sigma\rangle \longrightarrow^* \langle 24, \sigma\rangle.$$

3 Expressing Program Properties

Now that we have defined our small-step operational semantics, we can formally express different properties of programs. For instance:

- **Progress:** For each store σ and expression e that is not an integer, there exists a possible transition for $\langle e, \sigma\rangle$:
$$\forall e \in \textbf{Exp}.\ \forall \sigma \in \textbf{Store}.\ \text{either } e \in \textbf{Int} \text{ or } \exists e', \sigma'.\ \langle e, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$$
- **Termination:** The evaluation of each expression terminates:
$$\forall e \in \textbf{Exp}.\ \forall \sigma_0 \in \textbf{Store}.\ \exists \sigma \in \textbf{Store}.\ \exists n \in \textbf{Int}.\ \langle e, \sigma_0\rangle \longrightarrow^* \langle n, \sigma\rangle$$
- **Deterministic Result:** The evaluation result for any expression is deterministic:
$$\forall e \in \textbf{Exp}.\ \forall \sigma_0, \sigma, \sigma' \in \textbf{Store}.\ \forall n, n' \in \textbf{Int}.$$
$$\text{if } \langle e, \sigma_0\rangle \longrightarrow^* \langle n, \sigma\rangle \text{ and } \langle e, \sigma_0\rangle \longrightarrow^* \langle n', \sigma'\rangle \text{ then } n = n' \text{ and } \sigma = \sigma'.$$

How can we prove such kinds of properties? *Inductive proofs* allow us to prove statements such as the properties above. In the next lecture, We will first introduce inductive sets, introduce inductive proofs, and then show how we can prove progress (the first property above) using inductive techniques.