CS 411 Lecture 2 — Small-step Operational Semantics — September 1, 2004

Lecturer: Radu Rugina

Consider the language of arithmetic expressions:

$$\begin{aligned} x, y, z &\in \mathsf{Var} \\ n, m &\in \mathsf{Int} \\ e &\in \mathsf{Expr} \end{aligned}$$

$$e ::= x \;\mid\; n \;\mid\; e_1 + e_2 \;\mid\; e_1 * e_2$$

Operational semantics describes the evaluation of expressions as successive reductions of the expression on an abstract machine. Configurations capture the state of the abstract machine, and consist of the expression that remains to be evaluated and a store:

$$\begin{aligned} \mathsf{Config} &= \mathsf{Expr} \times \mathsf{Store} \\ \mathsf{Store} &= \mathsf{Var} \to \mathsf{Int} \end{aligned}$$

We will denote configurations using angle brackets. For instance, $\langle (x+2)*(y+1),\, s\rangle$, where x is a variable and s a store. We describe the successive reductions in the evaluation using a transition function:

$$\mathsf{OneStep} \subseteq \mathsf{Config} \times \mathsf{Config}$$

For instance, $(\langle (4+2)*(y+1),\, s\rangle, \langle 6*(y+1),\, s\rangle)$ in $\mathsf{OneStep}$. For better readability, we will write "$mc \to mc'$" instead of "$(mc, mc') \in \mathsf{OneStep}$". Hence: $\langle (4+2)*(y+1),\, s\rangle \to \langle 6*(y+1),\, s\rangle$.

Now defining the semantics of the language boils down to defining the one step evaluation relation $\to$ that describes the transition 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 $\mathsf{OneStep}$ using inference rules:

(var)
$$\frac{}{\langle x,\, s\rangle \to \langle n,\, s\rangle} \text{ where } n = s(x)$$

(plus)
$$\frac{}{\langle n+m,\, s\rangle \to \langle p,\, s\rangle} \text{ where } p \text{ is the sum of } n, m$$

(mul)
$$\frac{}{\langle n*m,\, s\rangle \to \langle p,\, s\rangle} \text{ where } p \text{ is the product of } n, m$$

(lplus)

$$\frac{\langle e_1,\, s\rangle \to \langle e_1',\, s\rangle}{\langle e_1 + e_2,\, s\rangle \to \langle e_1' + e_2,\, s\rangle}$$

(rplus)
$$\frac{\langle e_2,\, s\rangle \to \langle e_2',\, s\rangle}{\langle n + e_2,\, s\rangle \to \langle n + e_2',\, s\rangle}$$

(lmul)
$$\frac{\langle e_1,\, s\rangle \to \langle e_1',\, s\rangle}{\langle e_1 * e_2,\, s\rangle \to \langle e_1' * e_2,\, s\rangle}$$

(rmul)
$$\frac{\langle e_2,\, s\rangle \to \langle e_2',\, s\rangle}{\langle n * e_2,\, s\rangle \to \langle n * e_2',\, s\rangle}$$

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 hold. 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.

1 Using the Semantic Rules

Let's see how we can use these rules. Consider that we want to evaluate expression $(x + 2) * (y + 1)$ in a store $s = \{x = 4, y = 3\}$. That is, we want to find the transition for configuration $\langle (x + 2) * (y + 1),\, s\rangle$. 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 $e_1 = x + 2, e_2 = y + 1$, but e_1' is not yet known:

(lmul)
$$\frac{\langle x + 2,\, s\rangle \to \langle e_1',\, s\rangle}{\langle (x + 2) * (y + 1),\, s\rangle \to \langle e_1' * (y + 1),\, s\rangle}$$

Now we need to show that the premise actually holds and find out who e_1' is. We look for a rule whose conclusion matches $\langle x + 2,\, s\rangle \to \langle e_1',\, s\rangle$. We find that (lplus) is the only matching rule:

(lplus)
$$\frac{\langle x,\, s\rangle \to \langle e_1'',\, s\rangle}{\langle x + 2,\, s\rangle \to \langle e_1'' + 2,\, s\rangle}$$

where $e_1' = e_1'' + 2$. We repeat this reasoning for $\langle x,\, s\rangle \to \langle e_1'',\, s\rangle$, and we find that the only applicable rule is the axiom (var):

(var)
$$\frac{}{\langle x,\, s\rangle \to \langle 4,\, s\rangle}$$

because the store is $s = \{x = 4, y = 3\}$. 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:

$$\dfrac{\dfrac{\dfrac{}{\langle x,\, s\rangle \to \langle 4,\, s\rangle}\ (\text{var})}{\langle x + 2,\, s\rangle \to \langle 4 + 2,\, s\rangle}\ (\text{lplus})}{\langle (x + 2) * (y + 1),\, s\rangle \to \langle (4 + 2) * (y + 1),\, s\rangle}\ (\text{lmul})$$

This proves that, given our inference rules, the one-step transition $\langle (x+2)*(y+1),\, s\rangle \to \langle (4+2)*(y+1),\, s\rangle$ 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:

$$\dfrac{\dfrac{}{\langle 4+2,\, s\rangle \to \langle 6,\, s\rangle}\ \text{(plus)}}{\langle (4+2)*(y+1),\, s\rangle \to \langle 6*(y+1),\, s\rangle}\ \text{(lmul)}$$

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:

$$\langle (x+2)*(y+1),\, s\rangle \to \langle (4+2)*(y+1),\, s\rangle \to \langle 6*(y+1),\, s\rangle \to \langle 6*(3+1),\, s\rangle \to \langle 6*4,\, s\rangle \to \langle 24,\, s\rangle$$

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 $\langle n,\, s\rangle$ where n is a number and s is a store.

We describe multiple steps of evaluation using a relation $\to^*$, which is the transitive, reflexive closure of $\to$: $\to^* = \{(mc, mc') \mid \exists n \geq 0 \text{ such that } mc \to^n mc'\}$.

2 Expressing Program Properties

We can now formally express different properties of programs. For instance:

Termination: The evaluation of each expression terminates:

$$\forall e \in \mathsf{Expr}, \forall s \in \mathsf{Store}, \exists n \in \mathsf{Int} : \langle e,\, s\rangle \to^* \langle n,\, s\rangle$$

or:

Deterministic Result: The evaluation result for any expression is deterministic:

$$\forall e \in \mathsf{Expr}, \forall s \in \mathsf{Store}, \forall n, m \in \mathsf{Int} : \langle e,\, s\rangle \to^* \langle n,\, s\rangle \text{ and } \langle e,\, s\rangle \to^* \langle n',\, s\rangle \text{ implies } n = n'$$

or

Progress: For each store s and expression e that is not an integer, there exists a possible transition for $\langle e,\, s\rangle$:

$$\forall e, \forall s : (e \in \mathsf{Int}) \text{ or } (\exists e' : \langle e,\, s\rangle \to \langle e',\, s\rangle)$$

How can we prove such kinds of properties? Let's take the last property above and rephrase is as: for all expressions e, $P(e)$ holds, where:

$$P(e) = \forall s : (e \in \mathsf{Int}) \text{ or } (\exists e' : \langle e,\, s\rangle \to \langle e',\, s\rangle)$$

The idea is to build a proof that follows the inductive structure in the grammar of expressions: $e ::= x \mid n \mid e_1 + e_2 \mid e_1 * e_2$. This is called "structural induction on the expressions e". We must examine each case in the grammar and show that P(e) holds for that case. Since the grammar productions $e = e_1 + e_2$ and $e = e_1 * e_2$ are inductive definitions of expression, they are inductive steps in the proof; the other two cases $e = x$ and $e = n$ are the basis of induction. The proof goes as follows.

- Case $e = x$.

  By the (var) axiom, we can evaluate $\langle x, s\rangle$ in any state: $\langle x, s\rangle \to \langle n, s\rangle$, where $n = s(x)$. So $e' = n$ is a witness that there exists e' such that $\langle x, s\rangle \to \langle e', s\rangle$, and $P(x)$ holds.

- Case $e = n$.

  Then $e \in \mathsf{Int}$, so $P(n)$ trivially holds.

- Case $e = e_1 + e_2$.

  This is an inductive step. We assume that P holds for subexpressions e_1 and e_2 and we want to prove that is holds for e. In other words, $P(e_1)$ and $P(e_2)$ implies $P(e)$. Let's expand these properties. We know that:

  $$P(e_1) = \forall s : (e_1 \in \mathsf{Int}) \text{ or } (\exists e' : \langle e_1, s\rangle \to \langle e', s\rangle) P(e_2) \quad = \forall s : (e_2 \in \mathsf{Int}) \text{ or } (\exists e' : \langle e_2, s\rangle \to \langle e', s\rangle)$$

  and want to show:

  $$P(e) = \forall s : (e \in \mathsf{Int}) \text{ or } (\exists e' : \langle e, s\rangle \to \langle e', s\rangle)$$

  We must inspect several subcases.

  First, if both e_1 and e_2 are integer constants, say $e_1 = n_1$ and $e_2 = n_2$, then by rule (plus) we know that the transition $\langle n_1 + n_2, s\rangle \to \langle m, s\rangle$ is valid, where m is the sum of n_1 and n_2. Hence, $P(e) = P(n_1 + n_2)$ holds (with witness $e' = m$).

  Second, if e_1 is not an integer constant, then by the inductive hypothesis $P(e_1)$ we know that $\langle e_1, s\rangle \to \langle e', s\rangle$ for some e'. We can then use rule (lplus) to conclude $\langle e_1 + e_2, s\rangle \to \langle e' + e_2, s\rangle$, so $P(e) = P(e_1 + e_2)$ holds.

  Third, if e_1 is an integer constant, say $e_1 = n_1$, but e_2 is not, then by the inductive hypothesis $P(e_2)$ we know that $\langle e_2, s\rangle \to \langle e', s\rangle$ for some e'. We can then use rule (lplus) to conclude $\langle n_1 + e_2, s\rangle \to \langle n_1 + e', s\rangle$, so $P(e) = P(n_1 + e_2)$ holds.

- Case $e = e_1 * e_2$.

  The proof in this case is similar to that in the previous case.