Recall the definition of the syntax of our language:

$$\textsf{int}\quad \frac{}{i \in \textsf{Exp}}$$

$$\textsf{var}\quad \frac{}{x \in \textsf{Exp}}$$

$$\textsf{plus}\quad \frac{e_1 \in \textsf{Exp} \qquad e_2 \in \textsf{Exp}}{e_1 + e_2 \in \textsf{Exp}}$$

$$\textsf{times}\quad \frac{e_1 \in \textsf{Exp} \qquad e_2 \in \textsf{Exp}}{e_1 * e_2 \in \textsf{Exp}}$$

We write $\vdash e \in \textsf{Exp}$ if we can construct a derivation (i.e., a formal proof tree using the rules above) to conclude that e is an expression.

The definition of our abstract machine states:

$$\begin{array}{rcl} s & \in & \textsf{Store} : \textsf{Var} \to \textsf{Int} \\ \textsf{MS} & \in & \textsf{MachineState} ::= (e, s) \end{array}$$

Finally, recall the inference rules that allow us to conclude that (e, s) transitions to (e', s):

$$\textsf{t-var}\quad \frac{}{(x, s) \to (s(x), s)}$$

$$\textsf{t-plus}\quad \frac{}{(i_1 + i_2, s) \to (i, s)}\ (i = i_1 + i_2)$$

$$\textsf{t-times}\quad \frac{}{(i_1 * i_2, s) \to (i, s)}\ (i = i_1 * i_2)$$

$$\textsf{t-lplus}\quad \frac{(e_1, s) \to (e_1', s)}{(e_1 + e_2, s) \to (e_1' + e_2, s)}$$

$$\textsf{t-rplus}\quad \frac{(e_2, s) \to (e_2', s)}{(i_1 + e_2, s) \to (i_1 + e_2', s)}$$

$$\textsf{t-ltimes}\quad \frac{(e_1, s) \to (e_1', s)}{(e_1 * e_2, s) \to (e_1' * e_2, s)}$$

$$\textsf{t-rtimes}\quad \frac{(e_2, s) \to (e_2', s)}{(i_1 * e_2, s) \to (i_1 * e_2', s)}$$

We write $\vdash (e_1, s) \to (e_2, s)$ if we can construct a derivation that (e_1, s) steps to (e_2, s) using the transition rules above.

We define the height of a proof tree $\mathcal{P}$ as follows:

- If $\mathcal{P}$ is an instance of an axiom, then $\mathsf{height}(\mathcal{P}) = 0$.
- If $\mathcal{P}$ is a proof whose root ends with an instance of an inference rule, then the height of $\mathcal{P}$ is 1 + the maximum of the heights of the proofs that make up the sub-trees of the proof, corresponding to the assumptions.

Theorem 1. *If* $\vdash e \in \mathsf{Exp}$ *and* s *is a store, then either* e *is an integer and there is no* e' *such that* $(e, s) \to (e', s)$*, or else there exists a unique* e' *such that* $\vdash (e, s) \to (e', s)$*.*

Proof. We argue the proof by induction on the height h of the proof tree $\mathcal{P}$ that allows us to conclude that $e \in \mathsf{Exp}$.

If the proof tree $\mathcal{P}$ has height 0, then the proof tree must end with an axiom (see the definition of the height above). There are two axioms that allow us to conclude $e \in \mathsf{Exp}$, the int and var rules.

For the int rule, the theorem is immediately satisifed since i is an integer, and by inspection of the conclusions of the rules, there is no transition that allows to move to a new expression when starting from an integer.

For the var rule, we must show there is a unique e' such that $(x, s) \to (e', s)$. By inspection of the transition rules, only the t-var rule can apply:

$$\textsf{t-var}\ \frac{}{(x, s) \to (s(x), s)}$$

and this allows us to conclude that $e' = s(x)$. Since s is a function, there is at most one integer i such that $s(x) = i$.

This concludes the base case where our proof tree has height 0. Now our induction hypothesis is:

IH: For all e_1, such that the proof of $e_1 \in \mathsf{Exp}$ is less than or equal to h, e_1 is either an integer, or else there exists a unique e_1' such that $(e_1, s) \to (e_1', s)$.

Note that we are using strong induction here, since we're assuming the property holds for all proof trees smaller than $h + 1$ (not just those proof trees of height h).

We must show that for a proof of $e \in \mathsf{Exp}$ of height $h + 1$, the theorem holds. Since the proof has height greater than 0, it must end with something besides an axiom. Thus, the proof must end with an instance of the plus rule or the times rule.

First case. Our proof $\mathcal{P}$ that $e \in \mathsf{Exp}$ looks like this:

$$\textsf{plus}\ \frac{\dfrac{\mathcal{P}_1}{e_1 \in \mathsf{Exp}} \qquad \dfrac{\mathcal{P}_2}{e_2 \in \mathsf{Exp}}}{e_1 + e_2 \in \mathsf{Exp}}$$

That is, $e = e_1 + e_2$. Note that the height of $\mathcal{P}_1$ and the height of $\mathcal{P}_2$ must be less than or equal to h, since the height of $\mathcal{P}$ is $h + 1$.

Applying the induction hypothesis to the proof $\mathcal{P}_1$ that $e_1 \in \mathsf{Exp}$, e_1 is either an integer i_1 or else there exists a unique e_1' such that $(e_1, s) \to (e_1', s)$. Similarly, by the induction hypothesis, e_2 is either an integer or else there exists a unique e_2' such that $(e_2, s) \to (e_2', s)$.

If e_1 and e_2 are both integers, then only one transition rule applies, namely t-plus. (t-lplus cannot apply since, by induction, there isn't a transition for e_1 if it's an integer. Similarly, t-rplus cannot apply since, by induction, there isn't a transition rule for e_2 if it's an integer.)

If e_1 is an integer, but e_2 is not, then only one rule applies, namely t-lplus. (Note that t-plus and t-rplus don't match since e_1 isn't an integer.) Since there is at most one e_1' such that $(e_1, s) \to (e_1', s)$, we can conclude that $(e_1 + e_2, s) \to (e_1' + e_2, s)$ and this is the only possible transition.

Second case. Our proof $\mathcal{P}$ that $e \in \mathsf{Exp}$ looks like this:

$$\mathsf{times}\ \frac{\dfrac{\mathcal{P}_1}{e_1 \in \mathsf{Exp}} \qquad \dfrac{\mathcal{P}_2}{e_2 \in \mathsf{Exp}}}{e_1 * e_2 \in \mathsf{Exp}}$$

The analysis proceeds in exactly the same fashion as for plus. □

Our theorem shows that evaluation of expressions is deterministic. That is, the transition relation is a partial function taking machine states to machine states. The machine is finished computing when we've reduced the expression to an integer. That is, we consider machine states of the form (i, s) as terminal states.

So, we might define:

$$\mathsf{eval}(e, s) = i \text{ if } (e, s) \to^* (i, s),$$

where we interpret $\to^*$ as the reflexive, transitive closure of the transition relation. More formally, we might define:

$$\mathsf{Z}\quad \frac{}{\mathsf{eval}(i, s) = i}$$

$$\mathsf{S}\quad \frac{(e, s) \to (e', s) \qquad \mathsf{eval}(e', s) = i}{\mathsf{eval}(e, s) = i}$$

There is yet another way that we could define eval directly, without appealing to the one-step transition rules:

$$\mathsf{EI}\quad \frac{}{(i, s) \Downarrow i}$$

$$\mathsf{EV}\quad \frac{}{(x, s) \Downarrow s(x)}$$

$$\mathsf{E+}\quad \frac{(e_1, s) \Downarrow i_1 \qquad (e_2, s) \Downarrow i_2}{(e_1 + e_2, s) \Downarrow i}\ (i = i_1 + i_2)$$

$$\mathsf{E*}\quad \frac{(e_1, s) \Downarrow i_1 \qquad (e_2, s) \Downarrow i_2}{(e_1 * e_2, s) \Downarrow i}\ (i = i_1 * i_2)$$

The $\Downarrow$ relation is an example of a big-step semantics because the evaluation of a sub-expression happens in one big step. In contrast, the $\rightarrow$ relation is a small-step semantics, which only captures one step in the abstract machine. For example, we can prove $((3 * 4) + 2, s) \Downarrow 14$ like this:

$$\mathsf{E+}\ \frac{\mathsf{E*}\ \dfrac{\mathsf{EI}\ \dfrac{}{(3,s) \Downarrow 3} \qquad \mathsf{EI}\ \dfrac{}{(4,s) \Downarrow 4}\ (12 = 3 * 4)}{(3 * 4, s) \Downarrow 12} \qquad \mathsf{EI}\ \dfrac{}{(2,s) \Downarrow 2}}{((3 * 4) + 2, s) \Downarrow 14}\ (14 = 12 + 2)$$

Big-step semantics are often easier to use in reasoning about the answers that a program might produce, because we don't have to get caught up in all of the intermediate machine states. But big-step semantics have their own problems. In particular, they're not very good for modelling programs that might run forever (e.g., an operating system or server).

Homework

Hand in to Prof. Morrisett in class next Monday (Sep. 8). Type your homework or write very, very neatly.

1. Prove that if $\vdash e \in \mathsf{Exp}$ and s is a store, then there exists a unique i such that $(e, s) \Downarrow i$.
2. Define a transition relation $\Rightarrow$ that forces evaluation to go right-to-left instead of left-to-right.
3. Consider adding the following alternative transition semantics for our language:

   $$\mathsf{t\text{-}var}\quad \frac{}{(x,s) \rightsquigarrow (s(x), s)}$$

   $$\mathsf{t\text{-}plus}\quad \frac{}{(i_1 + i_2, s) \rightsquigarrow (i, s)}\ (i = i_1 + i_2)$$

   $$\mathsf{t\text{-}times}\quad \frac{}{(i_1 * i_2, s) \rightsquigarrow (i, s)}\ (i = i_1 * i_2)$$

   $$\mathsf{t\text{-}lplus}\quad \frac{(e_1, s) \rightsquigarrow (e_1', s)}{(e_1 + e_2, s) \rightsquigarrow (e_1' + e_2, s)}$$

   $$\mathsf{t\text{-}rplus}\quad \frac{(e_2, s) \rightsquigarrow (e_2', s)}{(e_1 + e_2, s) \rightsquigarrow (e_1 + e_2', s)}$$

   $$\mathsf{t\text{-}ltimes}\quad \frac{(e_1, s) \rightsquigarrow (e_1', s)}{(e_1 * e_2, s) \rightsquigarrow (e_1' * e_2, s)}$$

   $$\mathsf{t\text{-}rtimes}\quad \frac{(e_2, s) \rightsquigarrow (e_2', s)}{(e_1 * e_2, s) \rightsquigarrow (e_1 * e_2', s)}$$

   The rules are exactly the same as for $\rightarrow$ except that the t-rplus and t-rtimes rules are slightly different.

   How would our proof that there is at most one machine state that we can step to break? Give a counter example and then show where in the proof we would be making a flawed assumption. (Just cut and paste the text of my proof, put it in quotes, and say something about why it is wrong.)
4. Prove that $\mathsf{eval}(e, s) = i$ if and only if $(e, s) \Downarrow i$.