1 Large-step semantics

So far we have defined the small step evaluation relation \( \rightarrow \subseteq \text{Config} \times \text{Config} \) for our simple language of arithmetic expressions, and used its transitive and reflexive closure \( \rightarrow^* \) to describe the execution of multiple steps of evaluation. In particular, if \( \langle e, \sigma \rangle \) is some start configuration, and \( \langle n, \sigma' \rangle \) is a final configuration, the evaluation \( \langle e, \sigma \rangle \rightarrow^* \langle n, \sigma' \rangle \) shows that by executing expression \( e \) starting with the store \( \sigma \), we get the result \( n \), and the final store \( \sigma' \).

Large-step semantics is an alternative way to specify the operational semantics of a language. Large-step semantics directly give the final result.

We'll use the same configurations as before, but define a large step evaluation relation:

\[
\downarrow \subseteq \text{Config} \times \text{FinalConfig}
\]

where

\[
\text{Config} = \text{Exp} \times \text{Store}
\]

and

\[
\text{Final Config} = \text{Int} \times \text{Store} \subseteq \text{Config}.
\]

We write \( \langle e, \sigma \rangle \downarrow \langle n, \sigma' \rangle \) to mean that \( \langle e, \sigma \rangle \) evaluates in one big step directly to final configuration \( \langle n, \sigma' \rangle \). In other words, the big step semantics takes a configuration to an “answer”. For our language of arithmetic expressions, “answers” are a subset of configurations, but this is not always true in general.

The large step semantics boils down to defining the relation \( \downarrow \). We use inference rules to inductively define the relation \( \downarrow \), similar to how we specified the small-step operational semantics \( \rightarrow \).

\[
\text{INT}_{\text{LRG}} \quad \langle n, \sigma \rangle \downarrow \langle n, \sigma \rangle
\]

\[
\text{VAR}_{\text{LRG}} \quad \langle x, \sigma \rangle \downarrow \langle n, \sigma \rangle \quad \text{where } \sigma(x) = n
\]

\[
\text{ADD}_{\text{LRG}} \quad \langle e_1, \sigma \rangle \downarrow \langle n_1, \sigma'' \rangle \quad \langle e_2, \sigma'' \rangle \downarrow \langle n_2, \sigma' \rangle \quad \langle e_1 + e_2, \sigma \rangle \downarrow \langle n, \sigma' \rangle \quad \text{where } n \text{ is the sum of } n_1 \text{ and } n_2
\]

\[
\text{MUL}_{\text{LRG}} \quad \langle e_1, \sigma \rangle \downarrow \langle n_1, \sigma'' \rangle \quad \langle e_2, \sigma'' \rangle \downarrow \langle n_2, \sigma' \rangle \quad \langle e_1 \times e_2, \sigma \rangle \downarrow \langle n, \sigma' \rangle \quad \text{where } n \text{ is the product of } n_1 \text{ and } n_2
\]

\[
\text{ASG}_{\text{LRG}} \quad \langle e_1, \sigma \rangle \downarrow \langle n_1, \sigma'' \rangle \quad \langle e_2, \sigma''[x \mapsto n_1] \rangle \downarrow \langle n_2, \sigma' \rangle \quad \langle x := e_1; e_2, \sigma \rangle \downarrow \langle n_2, \sigma' \rangle
\]

To see how we use these rules, here is a proof tree that shows that \( \langle \text{foo} := 3; \text{foo} \times \text{bar}, \sigma \rangle \downarrow \langle 21, \sigma' \rangle \) for a store \( \sigma \) such that \( \sigma(\text{bar}) = 7 \), and \( \sigma' = \sigma[\text{foo} \mapsto 3] \).
A closer look to this structure reveals the relation between small step and large-step evaluation: a depth-first traversal of the large-step proof tree yields the sequence of one-step transitions in small-step evaluation.

## 2 Equivalence of semantics

So far, we have specified the semantics of our language of arithmetic expressions in two different ways: small-step operational semantics and large-step operational semantics. Are they expressing the same meaning of arithmetic expressions? Can we show that they express the same thing?

**Theorem** (Equivalence of semantics). For all expressions $e$, stores $\sigma$ and $\sigma'$, and integers $n$, we have:

$$\langle e, \sigma \rangle \Downarrow \langle n, \sigma' \rangle \iff \langle e, \sigma \rangle \rightarrow^* \langle n, \sigma' \rangle.$$

**Proof sketch.**

- **$\Rightarrow$.** We proceed by structural induction on expressions $e$. The property we will prove by induction is:

$$P(e) = \forall \sigma, \sigma' \in \text{Store}, \forall n \in \text{Int}. \langle e, \sigma \rangle \Downarrow \langle n, \sigma' \rangle \implies \langle e, \sigma \rangle \rightarrow^* \langle n, \sigma' \rangle.$$

We have to consider each of the possible axioms and inference rules for constructing an expression.

- **Case** $e \equiv x$.

Here, we are considering the case where the expression $e$ is equal to some variable $x$. Assume that for some $\sigma$, $\sigma'$, and $n$ we have $\langle x, \sigma \rangle \Downarrow \langle n, \sigma' \rangle$. That means that there is some derivation using the axioms and inference rules of the large-step operational semantics, whose conclusion is $\langle x, \sigma \rangle \Downarrow \langle n, \sigma' \rangle$. There is only one rule whose conclusion could look like this, the rule $\text{Var}_{\text{Lrg}}$. That rule requires that $n = \sigma(x)$, and that $\sigma' = \sigma$.

(This reasoning is an example of inversion: using the inference rules in reverse. That is, we know that some conclusion holds—$\langle x, \sigma \rangle \Downarrow \langle n, \sigma' \rangle$—and we examine the inference rules to determine which rule must have been used in the derivation, and thus which premises must be true, and which side conditions satisfied.)

Since $n = \sigma(x)$ we know that $\langle x, \sigma \rangle \rightarrow \langle n, \sigma \rangle$ also holds, by using the small-step axiom $\text{VAR}$. So we can conclude that $\langle x, \sigma \rangle \rightarrow^* \langle n, \sigma \rangle$ holds, which is what we needed to show.

- **Case** $e \equiv n$.

Here, we consider the case where expression $e$ is equal to some integer $n$. Assume that for some $\sigma$, $\sigma'$, and $n'$ we have $\langle n, \sigma \rangle \Downarrow \langle n', \sigma' \rangle$. Like the case above, by inversion, we know that the rule $\text{Int}_{\text{Lrg}}$ was used to conclude that $\langle n, \sigma \rangle \Downarrow \langle n', \sigma' \rangle$, and so $n' = n$ and $\sigma' = \sigma$.

So we need to show that $\langle n, \sigma \rangle \rightarrow^* \langle n, \sigma \rangle$. But this holds trivially because of reflexivity of $\rightarrow^*$.

- **Case** $e \equiv e_1 + e_2$.

This is an inductive case. Expressions $e_1$ and $e_2$ are subexpressions of $e$, and so we can assume that $P(e_1)$ and $P(e_2)$ hold. We need to show that $P(e)$ holds. Let’s write out $P(e_1)$, $P(e_2)$, and $P(e)$ explicitly.

$$P(e_1) = \forall n, \sigma, \sigma' : \langle e_1, \sigma \rangle \Downarrow \langle n, \sigma' \rangle \implies \langle e_1, \sigma \rangle \rightarrow^* \langle n, \sigma' \rangle$$

$$P(e_2) = \forall n, \sigma, \sigma' : \langle e_2, \sigma \rangle \Downarrow \langle n, \sigma' \rangle \implies \langle e_2, \sigma \rangle \rightarrow^* \langle n, \sigma' \rangle$$

$$P(e) = \forall n, \sigma, \sigma' : \langle e_1 + e_2, \sigma \rangle \Downarrow \langle n, \sigma' \rangle \implies \langle e_1 + e_2, \sigma \rangle \rightarrow^* \langle n, \sigma' \rangle$$
Assume that for some \( \sigma, \sigma' \) and \( n \) we have \( \langle e_1 + e_2, \sigma \rangle \Downarrow (n, \sigma') \). We now need to show that \( (e_1 + e_2, \sigma) \rightarrow^* (n, \sigma') \).

We assumed that \( \langle e_1 + e_2, \sigma \rangle \Downarrow (n, \sigma') \). Let's use inversion again: there is some derivation whose conclusion is \( \langle e_1 + e_2, \sigma \rangle \Downarrow (n, \sigma') \). By looking at the large-step semantic rules, we see that only one rule could possibly have a conclusion of this form: the rule ADD\(_{\text{LRG}}\). So that means that the last rule use in the derivation was ADD\(_{\text{LRG}}\). But in order to use the rule ADD\(_{\text{LRG}}\), it must be the case that \( \langle e_1, \sigma \rangle \Downarrow (n_1, \sigma'') \) and \( \langle e_2, \sigma'' \rangle \Downarrow (n_2, \sigma') \) hold for some \( n_1 \) and \( n_2 \) such that \( n = n_1 + n_2 \) (i.e., there is a derivation whose conclusion is \( \langle e_1, \sigma \rangle \Downarrow (n_1, \sigma'') \) and a derivation whose conclusion is \( \langle e_2, \sigma'' \rangle \Downarrow (n_2, \sigma') \)).

Using the inductive hypothesis \( P(e_1) \), since \( \langle e_1, \sigma \rangle \Downarrow (n_1, \sigma'') \), we must have \( \langle e_1, \sigma \rangle \rightarrow^* (n_1, \sigma'') \). Similarly, by \( P(e_2) \), we have \( \langle e_2, \sigma'' \rangle \rightarrow^* (n_2, \sigma') \). By Lemma 1 below, we have

\[
\langle e_1 + e_2, \sigma \rangle \rightarrow^* \langle n_1 + e_2, \sigma'' \rangle
\]

and by another application of Lemma 1 we have

\[
\langle n_1 + e_2, \sigma'' \rangle \rightarrow^* \langle n_1 + n_2, \sigma' \rangle
\]

and by the rule ADD we have

\[
\langle n_1 + n_2, \sigma' \rangle \rightarrow (n, \sigma').
\]

Thus, we have \( \langle e_1 + e_2, \sigma \rangle \rightarrow^* (n, \sigma') \), which proves this case.

+ **Case** \( e \equiv e_1 \times e_2 \). Similar to the case \( e = e_1 + e_2 \) above.
+ **Case** \( e \equiv x := e_1 : e_2 \). Omitted. Try it as an exercise.

\[\leftarrow.\] We proceed by mathematical induction on the number of steps \( (e, \sigma) \rightarrow^* (n, \sigma') \).

+ **Base case.** If \( (e, \sigma) \rightarrow^* (n, \sigma') \) in zero steps, then we must have \( e \equiv n \) and \( \sigma' = \sigma \). Then, \( (n, \sigma) \Downarrow (n, \sigma) \) by the large-step operational semantics rule INT\(_{\text{LRG}}\).

+ **Inductive case.** Assume that \( (e, \sigma) \rightarrow^* (n, \sigma') \) and that (the inductive hypothesis) \( (e', \sigma'') \Downarrow (n, \sigma') \). That is, \( (e', \sigma'') \rightarrow^* (n, \sigma') \) takes \( m \) steps, and we assume that the property holds for it \( (e'', \sigma''' ) \Downarrow (n, \sigma'') \), and we are considering \( (e, \sigma) \rightarrow^* (n, \sigma') \), which takes \( m + 1 \) steps. We need to show that \( (e, \sigma) \Downarrow (n, \sigma') \). This follows immediately from Lemma 2 below.

\[\square\]

**Lemma 1.** If \( (e, \sigma) \rightarrow^* (n, \sigma') \) then for all \( e_1, e_2 \) the following hold.

\[
\begin{align*}
(e + e_2, \sigma) & \rightarrow^* (n + e_2, \sigma') \\
(e \times e_2, \sigma) & \rightarrow^* (n \times e_2, \sigma') \\
(n_1 + e, \sigma) & \rightarrow^* (n_1 + n, \sigma') \\
(n_1 \times e, \sigma) & \rightarrow^* (n_1 \times n, \sigma')
\end{align*}
\]

**Proof.** By (mathematical) induction on the number of evaluation steps in \( \rightarrow^* \). \[\square\]

**Lemma 2.** For all \( e, e', \sigma \), and \( n \), if \( (e, \sigma) \rightarrow (e', \sigma'') \) and \( (e', \sigma'') \Downarrow (n, \sigma') \), then \( (e, \sigma) \Downarrow (n, \sigma') \).