Programming Languages and Computation

Week 8: Big Steps, Blocks and Procedures

We first practice working with small-step and big-step semantics, and consider the notion of derivation trees more formally. Optional exercises provide depth on all topics covered in this part of the unit (including induction, derivations, and the choices made in defining semantics) beyond what we will expect in assessment, but that you may find useful as foundations for pathways focused on programming languages and their theory.

Small-Step and Big-Step Derivations

Recall that the inference rules we use to inductively define our semantics relations ($\leftarrow$ and $\Downarrow$) can be used to construct *derivations* for particular facts. We constructed a few small-step derivations for particular transitions in the lecture, using the small-step semantics rules.

We can similarly construct derivations for the big-step semantics relation $\Downarrow$. For example, the following is a derivation for the fact $\langle \mathtt{v} \leftarrow 2\,;\ \mathtt{x} \leftarrow \mathtt{v}\ +\ 3, [] \rangle \Downarrow [\mathtt{v} \mapsto 2; \mathtt{x} \mapsto 5]$.

$$\dfrac{\dfrac{}{\langle \mathtt{v} \leftarrow 2, [] \rangle \Downarrow [\mathtt{v} \mapsto 2]} \qquad \dfrac{}{\langle \mathtt{x} \leftarrow \mathtt{v}\ +\ 3, [\mathtt{v} \mapsto 2] \rangle \Downarrow [\mathtt{v} \mapsto 2; \mathtt{x} \mapsto 5]}}{\langle \mathtt{v} \leftarrow 2\,;\ \mathtt{x} \leftarrow \mathtt{v}\ +\ 3, [] \rangle \Downarrow [\mathtt{v} \mapsto 2; \mathtt{x} \mapsto 5]}$$

* 1. Consider the While programs from Week 7 Question 4. Write big-step derivations for their semantics (when they exist) in the same initial state $\sigma = [\mathtt{n} \mapsto 3; \mathtt{a} \mapsto 5]$.

** 2. Without using procedures or blocks, write a While programme that computes the nth Pingala number $p(n)$—with p as defined below, where n is the value of variable `n` in the initial state.[a] The value of $p(n)$ should be, in the final state, stored in variable `r`. You may, if needed, assume that the initial value of `n` is non-negative. You may also, if needed, use variables other than `n` and `r` without worry.

$$p(n) = \begin{cases} 0 & \text{if } n \leq 0 \\ 1 & \text{if } n = 1 \\ p(n-2) + p(n-1) & \text{otherwise} \end{cases}$$

Hint: First work out the logic of turning the recursive definition into an iterative programme, then write it as a programme. Our language is restricted enough that it's a bit painful to do directly. Feel free to check that your programme is correct on a couple of small examples by unrolling execution traces.

[a] These are commonly known as Fibonacci number. Indian mathematician Pingala first wrote about them about 1000 years before Fibonacci.

** 3. Prove that our small-step semantics is functional.
Namely, show that, for any statement s and any states σ_0, σ and σ', if we have both $\langle s, \sigma_0 \rangle \to^* \langle \texttt{skip}, \sigma \rangle$ and $\langle s, \sigma_0 \rangle \to^* \langle \texttt{skip}, \sigma' \rangle$, then $\sigma = \sigma'$.
You may want to first argue that it is sufficient to show that the transition relation $\to$ itself is functional.

*** 4. Prove that the rules we gave for our big-step semantics are *complete* with respect to our small-step semantics.
Namely, show that, for any statement s and any states σ, σ' such that $\langle s, \sigma \rangle \to^* \langle \texttt{skip}, \sigma' \rangle$, there exists a big-step derivation for $\langle s, \sigma \rangle \Downarrow \sigma'$.

Note: We want to prove that the rules are in line with the definition we gave for $\Downarrow$, so don't use the definition itself in this proof—use the rules.
Hint: The proof is by *strong* mathematical induction on the length of the trace for $\langle s, \sigma \rangle \to^* \langle \texttt{skip}, \sigma' \rangle$. Strong mathematical induction works the same as mathematical induction, but in the inductive cases you get to assume that the induction hypothesis holds for all smaller values of k (instead of assuming it only for the immediate predecessor).
The idea of the proof is to "step" through the small-step trace and "construct" the big-step derivation step-by-step, but this doesn't work as smoothly as we'd like for sequential composition. (Can you see why?) In addition to strong induction, you may want to take a look at the properties you proved last week on the small-step semantics of sequential composition.

Blocks and Procedures

We now explore the While language extended with block structures and procedures.

** 5. Using only blocks, rewrite your solution to Question 2 so that, if the initial state is σ, then the final state is $\sigma[\texttt{r} \mapsto p(\sigma(\texttt{n}))]$, without any other changes (including to `n`).

** 6. *Without using recursion*, rewrite your solution to Question 2 *as a procedure* `pingala` that reads its only parameter from top-level variable `n` and writes its only output to top-level variable `r`, and otherwise leaves the state at point of call unchanged.

*** 7. Do the same as in Question 6 using recursion instead of a `while` loop.

Induction over Derivations: Small- and Big-Step Semantics (Optional)

Go back to a time before you knew about blocks, procedures, and scoped variables.

As noted by one of your colleagues in the lecture, rule ($\mathsf{BWhile}_\top$) looks very recursive. But so did the corresponding small-step rule ($\mathsf{SWhile}_\top$).

When reasoning about properties of the small-step semantics in Sheet 7, you ended up reasoning by mathematical induction over the length of the trace, because reasoning by induction over the statement would have led to a circular proof on `while` statements.

In the big step semantics, we do not have a trace: we have a derivation for $\Downarrow$. Fortunately for us, the relation is defined inductively, and we can reason by induction over the derivation itself! Base cases are the axioms (rules with no premises), and inductive cases get to assume that the induction hypothesis holds on derivations of the premises.

As an example, the following is the start of a proof that $\Downarrow$ is a partial function.

Proof. We want to show that, if $\langle s, \sigma_0 \rangle \Downarrow \sigma$ and $\langle s, \sigma_0 \rangle \Downarrow \sigma'$, then $\sigma = \sigma'$.

By induction over the derivation of $\langle s, \sigma_0 \rangle \Downarrow \sigma$, we show that if there is a derivation of $\langle s, \sigma_0 \rangle \Downarrow \sigma'$ for some σ', then it must be that $\sigma = \sigma'$.

Case (BSkip)**:** if $\langle s, \sigma_0 \rangle \Downarrow \sigma$ by (BSkip), then we have $s = \texttt{skip}$ and $\sigma = \sigma_0$. (BSkip) is the only rule to provide a derivation where the statement is `skip` and we therefore also have $\sigma' = \sigma_0$.

Case (BAss)**:** if $\langle s, \sigma_0 \rangle \Downarrow \sigma$ by (BAss), then we have $s = \texttt{x} \leftarrow a$ for some variable x and some arithmetic expression a, and $\sigma = \sigma_0\left[\texttt{x} \mapsto [\![\texttt{a}]\!]^{\mathscr{A}}(\sigma_0)\right]$. (BAss) is the only rule to provide a derivation where the statement is an assignment, therefore we also have $\sigma' = \sigma_0\left[\texttt{x} \mapsto [\![\texttt{a}]\!]^{\mathscr{A}}(\sigma_0)\right]$. We conclude by observing that $[\![\texttt{a}]\!]^{\mathscr{A}}(\cdot)$ is a function.

Case (BSeq)**:** if $\langle s, \sigma_0 \rangle \Downarrow \sigma$ by (BSeq), then we have $s = s_1; s_2$ for some statements s_1 and s_2, and we have σ_1 such that $\langle s_1, \sigma_0 \rangle \Downarrow \sigma_1$ and $\langle s_2, \sigma_1 \rangle \Downarrow \sigma$. (BSeq) is the only rule to provide a derivation where the statement is a sequential composition, therefore we also have σ_1' such that $\langle s_1, \sigma_0 \rangle \Downarrow \sigma_1'$ and $\langle s_2, \sigma_1' \rangle \Downarrow \sigma'$.

By induction hypothesis (on the derivation of $\langle s_1, \sigma_0 \rangle \Downarrow \sigma_1$), we have $\sigma_1 = \sigma_1'$, and also therefore (by induction hypothesis on the derivation of $\langle s_2, \sigma_1 \rangle \Downarrow \sigma$), we have $\sigma = \sigma'$.

*** 8. Finish the proof.

*** 9. Prove that the rules we gave in the lecture for big-step semantics are *sound* with respect to our small-step semantics.
Namely, show that, for any statement s, and any states σ and σ', if $\langle s, \sigma \rangle \Downarrow \sigma'$, then $\langle s, \sigma \rangle \rightarrow^* \langle \texttt{skip}, \sigma' \rangle$.
This proof will be by induction over the derivation for $\langle s, \sigma \rangle \Downarrow \sigma'$, and the idea of the proof is to construct a trace corresponding to the derivation. Once again, you may want to rely on results proved last week.