We define big-step evaluation relations for commands and expressions. For arithmetic expressions, we need a relation $\langle a, s\rangle \Downarrow n$, and for boolean expressions a relation $\langle b, s\rangle \Downarrow t$, where $t \in \{\mathbf{true}, \mathbf{false}\}$. These evaluation relations are similar to what we've seen before.

For commands, the big-step evaluation of a configuration $\langle c, s\rangle$ yields a final store s'. Hence, the evaluation relation for commands is of the form $\langle c, s\rangle \Downarrow s'$. The rules that define this relation are as follows.

$$\frac{}{\langle \mathbf{skip}, s\rangle \Downarrow s} \text{(skip)}$$

$$\frac{\langle a, s\rangle \Downarrow n}{\langle x := a, s\rangle \Downarrow s[x{-}\rangle n]} \text{(assign)}$$

$$\frac{\langle c1, s\rangle \Downarrow s' \langle c2, s'\rangle \Downarrow s''}{\langle c1; c2, s\rangle \Downarrow s''} \text{(seq)}$$

$$\frac{\langle b, s\rangle \Downarrow \mathbf{true} \langle c1, s\rangle \Downarrow s'}{\langle \mathbf{if}\ b\ \mathbf{then}\ c1\ \mathbf{else}\ c2, s\rangle \Downarrow s'} \text{(tif)}$$

$$\frac{\langle b, s\rangle \Downarrow \mathbf{false} \langle c2, s\rangle \Downarrow s'}{\langle \mathbf{if}\ b\ \mathbf{then}\ c1\ \mathbf{else}\ c2, s\rangle \Downarrow s'} \text{(fif)}$$

$$\frac{\langle b, s\rangle \Downarrow \mathbf{false}}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, s\rangle \Downarrow s} \text{(fwhile)}$$

$$\frac{\langle b, s\rangle \Downarrow \mathbf{true} \langle c, s\rangle \Downarrow s' \quad \langle \mathbf{while}\ b\ \mathbf{do}\ c, s'\rangle \Downarrow s''}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, s\rangle \Downarrow s''} \text{(twhile)}$$

It's interesting to see that the rule for **while** loops does not rely on using an **if** command (as we needed in the case of small-step semantics). Why does this rule work?

The small-step semantics suggest that the loop construct

$$\mathbf{while}\ b\ \mathbf{do}\ c$$

should be equivalent to

$$\mathbf{if}\ b\ \mathbf{then}\ (c; \mathbf{while}\ b\ \mathbf{do}\ c)\ \mathbf{else}\ \mathbf{skip}$$

Can we show that this indeed the case that the language is defined using the above large-step evaluation?

First, we need to to be more precise about what "equivalent commands" mean. Our formal model allows us to define this concept using large-step evaluations as follows (one can write a similar definition using $\rightarrow^*$ in small-step semantics)..

[Equivalence] Two commands c, c' are equivalent (written $c \sim c'$) if, for any states s and s', we have: $\langle c, s\rangle \Downarrow s' \iff \langle c', s\rangle \Downarrow s'$.

So we'd like to prove:

$$\textbf{while } b \textbf{ do } c \sim \textbf{if } b \textbf{ then } (c; \textbf{while } b \textbf{ do } c) \textbf{ else skip}$$

Let W be an abbreviation for **while** b **do** c. We want to show: $\langle W, s\rangle \Downarrow s'$ iff $\langle \textbf{if } b \textbf{ then } (c; W) \textbf{ else skip}, s\rangle \Downarrow s'$, for all s, s'. For this, we must show that both implications $\Rightarrow$ and $\Leftarrow$ hold. We'll show only direction *Rightarrow*; the other is similar.

Assume that s and s' are stores such that $\langle W, s\rangle \Downarrow s'$. It means that there is some derivation that proves for this fact. Inspecting the evaluation rules, we see that there are two possible rules whose conclusions match this fact. We analyze each of them in turn.

One rule is the (fwhile) rule, in which case the proof tree for $\langle W, s\rangle \Downarrow s'$ looks as follows:

$$\dfrac{\dfrac{\ldots(1)}{\langle b, s\rangle \Downarrow \textbf{false}}}{\langle \textbf{while } b \textbf{ do } c, s\rangle \Downarrow s} \text{ (fwhile)}$$

We can use subtree (1) to derive a proof tree showing that the evaluation of **if** b **then** $(c; W)$ **else skip** yields the same final state s:

$$\dfrac{\dfrac{\ldots(1)}{\langle b, s\rangle \Downarrow \textbf{false}}}{\langle \textbf{if } b \textbf{ then } (c; W) \textbf{ else skip}, s\rangle \Downarrow s} \text{ (fif)}$$

The other alternative is that $\langle W, s\rangle \Downarrow s$ has been derived using the (twhile) rule. In this case, the proof tree has the following form:

$$\dfrac{\dfrac{\ldots(2)}{\langle b, s\rangle \Downarrow \textbf{true}} \quad \dfrac{\ldots(3)}{\langle c, s\rangle \Downarrow s'} \quad \dfrac{\ldots(4)}{\langle \textbf{while } b \textbf{ do } c, s'\rangle \Downarrow s''}}{\langle \textbf{while } b \textbf{ do } c, s\rangle \Downarrow s''} \text{ (twhile)}$$

We can use subderivations (2), (3), and (4) to show that **if** b **then** $(c; W)$ **else skip** evaluates to the same final state s'':

$$\dfrac{\dfrac{\ldots(2)}{\langle b, s\rangle \Downarrow \textbf{true}} \quad \dfrac{\dfrac{\ldots(3)}{\langle c, s\rangle \Downarrow s'} \quad \dfrac{\ldots(4)}{\langle \textbf{while } b \textbf{ do } c, s'\rangle \Downarrow s''}}{\langle c; \textbf{while } b \textbf{ do } c, s\rangle \Downarrow s''}}{\langle \textbf{if } b \textbf{ then } (c; W) \textbf{ else skip}, s\rangle \Downarrow s''} \text{ (tif)}$$

Hence, we showed that in each of the two possible cases, the **if** command evaluates to the same final state as the **while** command.