1 Induction principle for big-step operational semantics

In this section we describe structural induction for the big-step semantics of the While-language. Every hypothesis of this induction principle is a one-to-one mapping from a corresponding derivation rule in the operational semantics. Computers can be and are used to automatically create induction principles from inductively defined datatypes and inference systems (like the operational semantics).

$$\frac{\begin{array}{l}\langle S,\ s\rangle \to s' \\ \forall s.\ P\ \mathbf{skip}\ s\ s \\ \forall \mathbf{x}\ a\ s.\ P\ (\mathbf{x} := a)\ s\ (s[\mathbf{x} \mapsto \mathcal{A}[a]_s]) \\ \forall S_1\ S_2\ s\ s'\ s''.\ \langle S_1,\ s\rangle \to s'' \longrightarrow P\ S_1\ s\ s'' \longrightarrow \\ \qquad \langle S_2,\ s''\rangle \to s' \longrightarrow P\ S_2\ s''\ s' \longrightarrow P\ (S_1;\ S_2)\ s\ s' \\ \forall b\ S_1\ S_2\ s\ s'.\ \langle S_1,\ s\rangle \to s' \longrightarrow P\ S_1\ s\ s' \longrightarrow \mathcal{B}[b]_s = \mathbf{tt} \longrightarrow \\ \qquad P\ (\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2)\ s\ s' \\ \forall b\ S_1\ S_2\ s\ s'.\ \langle S_2,\ s\rangle \to s' \longrightarrow P\ S_2\ s\ s' \longrightarrow \mathcal{B}[b]_s = \mathbf{ff} \longrightarrow \\ \qquad P\ (\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2)\ s\ s' \\ \forall b\ S\ s\ s'\ s''.\ \langle S,\ s\rangle \to s'' \longrightarrow P\ S\ s\ s'' \longrightarrow \mathcal{B}[b]_s = \mathbf{tt} \longrightarrow \\ \qquad \langle \mathbf{while}\ b\ \mathbf{do}\ S,\ s''\rangle \to s' \longrightarrow P\ (\mathbf{while}\ b\ \mathbf{do}\ S)\ s''\ s' \longrightarrow \\ \qquad P\ (\mathbf{while}\ b\ \mathbf{do}\ S)\ s\ s' \\ \forall b\ S\ s.\ \mathcal{B}[b]_s = \mathbf{ff} \longrightarrow P\ (\mathbf{while}\ b\ \mathbf{do}\ S)\ s\ s\end{array}}{P\ S\ s\ s'}$$

2 Determinism for big-step operational semantics

Prove that whenever $\langle S,\ s\rangle \to s'$ and $\langle S,\ s\rangle \to s''$ then $s' = s''$.

For this exercise there are two things to keep in mind. The first is that we need to do induction on the derivation of big-step semantics rather than the program itself. If we do not do this the case for the $[\mathsf{while}^{\mathsf{tt}}_{\mathsf{ns}}]$ rule will break as that rule requires the entire **while**-deiravation to appear in the hypothesis of the rule rather than the structurally smaller body of the loop that the induction hypothesis for program induction provides. The second thing is to keep in mind that the state s'' is still universally quantified in the induction hypothesis – it is not the same s'' as in our goal as that is the state that the entire program terminates in, and not the state for the individual sub components of the program. To clarify this point, I will rewrite the lemma that we are trying to prove in the following way.

Prove that if $\langle S,\ s\rangle \to s'$ then for all states s'', whenever $\langle S,\ s\rangle \to s''$ then $s' = s''$.

The reason that this is usually glossed over is that at the top level everything is universally quantified (all of S, s, s', and s'') but since we will be doing induction on $\langle S,\ s\rangle \to s'$ the names $S\ s\ s'$ need to be fixed. Do note, however, that each individual step of the induction principle universally quantifies its sub-components.

Also note, in the $[\mathsf{skip}_{\mathsf{ns}}]$-case for example, that the universally quantified s from the rule, and the universally quantified s'' from the statement that we

are trying to prove immediately appear in our assumptions. This is due to the standard way that universal quantifiers and implications work; for example, $\forall a\ b\ c, P \to Q \to R$ means that for all a, b, and c, assuming P *and* Q, prove R.

If this is all confusing (and I'm sure it is) I want you to look at the indcution principle for the big-step operational semantics and see exactly how each case matches up to the text in the proof bellow. Moreover, I have included the definition of P for this particular proof and the instantiations for every single case in green background color. This is not a part of the proof, but it may help you to see exactly what is going on.

And without further ado, here is the proof.

Proof by induction on $\langle S,\ s\rangle \to s'$

$$P \triangleq \lambda S\ s\ s'.\ \forall s''.\ \langle S,\ s\rangle \to s'' \longrightarrow s' = s''$$

Case [$\mathsf{skip}_{\mathsf{ns}}$]

$$\forall s''.\ \langle \mathbf{skip},\ s\rangle \to s'' \longrightarrow s = s''$$

Assume A: $\langle \mathbf{skip},\ s\rangle \to s''$

From A and [$\mathsf{skip}_{\mathsf{ns}}$] we have that $s = s''$.

Note that when this case is instantiated in the induction principle, s' is instantiated with s and hence the goal we have to prove is $s = s''$ and not $s' = s''$.

Case [$\mathsf{ass}_{\mathsf{ns}}$]

$$\forall \mathtt{x}\ a.\ \forall s''.\ \langle \mathtt{x} := a,\ s\rangle \to s'' \longrightarrow s[\mathtt{x} \mapsto \mathcal{A}[a]_s] = s''$$

Assume A: $\langle \mathtt{x} := a,\ s\rangle \to s''$

From A and [$\mathsf{ass}_{\mathsf{ns}}$] we have that $s[x \mapsto \mathcal{A}[a]_s] = s''$.

Case [$\mathsf{comp}_{\mathsf{ns}}$]

$$\begin{aligned}\forall S_1\ S_2\ s\ s'\ s'''.\ \langle S_1,\ s\rangle \to s''' \longrightarrow (\forall s''.\ \langle S_1,\ s\rangle \to s'' \longrightarrow s''' = s'') \longrightarrow\\ \langle S_2,\ s'''\rangle \to s' \longrightarrow (\forall s''.\ \langle S_2,\ s'''\rangle \to s'' \longrightarrow s' = s'') \longrightarrow\\ \forall s''.\ \langle S_1;\ S_2,\ s\rangle \to s'' \longrightarrow s' = s''\end{aligned}$$

Assume A: $\langle S_1,\ s\rangle \to s'''$ and B: $\langle S_2,\ s'''\rangle \to s'$ and C: $\langle S_1;\ S_2,\ s\rangle \to s''$
Assume IHA: for all s'', if $\langle S_1,\ s\rangle \to s''$ then $s''' = s''$
Assume IHB: for all s'', if $\langle S_2,\ s'''\rangle \to s''$ then $s' = s''$

Note that we have two induction hypotheses here and note that the universally quantified s'' is fundamentally important. It is *not* the same s'' as in B! Nor are the s''s in IHA and IHB equal to each other – they are both locally universally quantified to IHA and IHB respectively. It's like having two functions in your favourite program that happen to have the same name for one of their arguments – that does not mean that these arguments must have the same value when the function is called, or even be of the same type.
Also note that IHA is instantiated with S_1, s and s''' while IHB is instantiated with S_2, s''' and s'. If this seems strange to you, look at the [$\mathsf{comp}_{\mathsf{ns}}$]-case for the big-step semantics induction principle. We do exactly the same thing there.
Finally, note that we actually do not need A and B for this proof step. They will be included for the remaining steps as well, but marked as (not needed).

From C and [$\mathsf{comp}_{\mathsf{ns}}$] we know that there exists a s'''' such that
D: $\langle S_1,\ s\rangle \to s''''$ and E: $\langle S_2,\ s''''\rangle \to s''$.
From D and IHA we know that $s''' = s''''$ and hence with
E and IHB that $s' = s''$.

Recall that IHA says that for all s'', if $\langle S_1,\ s\rangle \to s''$ then $s''' = s''$. This means that IHA holds for all possible instantiations of s'', in particular it holds for s'''' which is what we were interested in for this particular case.

Case [$\mathsf{if}^{\mathsf{tt}}_{\mathsf{ns}}$]

$$\begin{aligned}\forall b\ S_1\ S_2\ s\ s'.\ \langle S_1,\ s\rangle \to s' \longrightarrow (\forall s''.\ \langle S_1,\ s\rangle \to s'' \longrightarrow s' = s'') \longrightarrow\\ \mathcal{B}[b]_s = \mathbf{tt} \longrightarrow\\ \forall s''.\ \langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2,\ s\rangle \to s'' \longrightarrow s' = s''\end{aligned}$$

Assume $\langle S_1,\ s\rangle \to s'$ (not needed)
Assume A: $\mathcal{B}[b]_s = \mathbf{tt}$ and B: $\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2,\ s\rangle \to s''$
Assume IH: for all s'', if $\langle S_1,\ s\rangle \to s''$ then $s' = s''$

From A, B, and [$\mathsf{if}^{\mathsf{tt}}_{\mathsf{ns}}$] we have that $\langle S_1,\ s\rangle \to s''$ and with IH that $s' = s''$.

Case [$\mathsf{if}^{\mathsf{ff}}_{\mathsf{ns}}$]

$$\begin{aligned}\forall b\ S_1\ S_2\ s\ s'.\ \langle S_2,\ s\rangle \to s' \longrightarrow (\forall s''.\ \langle S_2,\ s\rangle \to s'' \longrightarrow s' = s'') \longrightarrow\\ \mathcal{B}[b]_s = \mathbf{ff} \longrightarrow\\ \forall s''.\ \langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2,\ s\rangle \to s'' \longrightarrow s' = s''\end{aligned}$$

Assume $\langle S_2,\ s\rangle \to s'$ (not needed)
Assume A: $\mathcal{B}[b]_s = \mathbf{ff}$ and B: $\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2,\ s\rangle \to s''$
Assume IH: for all s'', if $\langle S_2,\ s\rangle \to s''$ then $s' = s''$

From A, B and [$\mathsf{if}^{\mathsf{ff}}_{\mathsf{ns}}$] we have that $\langle S_2,\ s\rangle \to s''$ and with IH that $s' = s''$.

Case [$\mathsf{while}^{\mathsf{tt}}_{\mathsf{ns}}$]

$$\forall S\ s\ s'\ s'''.\ \langle S,\ s\rangle \to s''' \longrightarrow (\forall s''.\ \langle S,\ s\rangle \to s'' \longrightarrow s''' = s'') \longrightarrow \\ \mathcal{B}[b]_s = \mathbf{tt} \longrightarrow \langle \mathbf{while}\ b\ \mathbf{do}\ S,\ s'''\rangle \to s' \longrightarrow \\ (\forall s''.\ \langle \mathbf{while}\ b\ \mathbf{do}\ S,\ s'''\rangle \to s'' \longrightarrow s' = s'') \longrightarrow \\ \forall s''.\ \langle \mathbf{while}\ b\ \mathbf{do}\ S,\ s\rangle \to s'' \longrightarrow s' = s''$$

Assume $\langle S,\ s\rangle \to s'''$ (not needed)
Assume $\langle \mathbf{while}\ b\ \mathbf{do}\ S,\ s'''\rangle \to s'$ (not needed)
Assume A: $\mathcal{B}[b]_s = \mathbf{tt}$ and B: $\langle \mathbf{while}\ b\ \mathbf{do}\ S,\ s\rangle \to s''$
Assume IHA: for all s'', if $\langle S,\ s\rangle \to s''$ then $s''' = s''$
Assume IHB: for all s'', if $\langle \mathbf{while}\ b\ \mathbf{do}\ S,\ s'''\rangle \to s''$ then $s' = s''$

> Note that the while loop appears in IHB, with other states than in the goal, but it turns out that this is exactly the induction hypothesis we need (but maybe not the one we deserve). Otherwise, this step is identical to the one for $[\mathsf{comp}_{\mathsf{ns}}]$.

From A, B and $[\mathsf{while}^{\mathsf{tt}}_{\mathsf{ns}}]$ we know that there exists a s'''' such that
C: $\langle S,\ s\rangle \to s''''$ and D: $\langle \mathbf{while}\ b\ \mathbf{do}\ S,\ s''''\rangle \to s''$.
From C and IHA we know that $s''' = s''''$ and hence with
D and IHB that $s' = s''$.

Case $[\mathsf{while}^{\mathsf{ff}}_{\mathsf{ns}}]$

$$\forall b\ S\ s.\ \mathcal{B}[b]_s = \mathbf{ff} \longrightarrow \forall s''.\ \langle \mathbf{while}\ b\ \mathbf{do}\ S,\ s\rangle \to s'' \longrightarrow s = s''$$

Assume A: $\mathcal{B}[b]_s = \mathbf{ff}$ and B: $\langle \mathbf{while}\ b\ \mathbf{do}\ S,\ s\rangle \to s''$

From A, B and $[\mathsf{while}^{\mathsf{ff}}_{\mathsf{ns}}]$ we have that $s = s''$

3 Lemma 2.19

In this section we are going to prove Lemma 2.19 from the book. This proof requires induction on the length of the derivation sequence rather then induction on the inference tree.

If $\langle S_1;\ S_2,\ s\rangle \Rightarrow^k s''$ then there exists a state s' and natural numbers k_1 and k_2 such that $\langle S_1,\ s\rangle \Rightarrow^{k_1} s'$ and $\langle S_2,\ s'\rangle \Rightarrow^{k_2} s''$ and $k = k_1 + k_2$.

> Note how I strengthen the induction principle in the inductive case as much as possible by maintaining the quantifier for all variables other than k.

Proof by induction on k

Case (k = 0)
Assume $\langle S_1;\ S_2,\ s\rangle \Rightarrow^0 s''$
This case holds vacuosly as no derivation sequence exists in zero steps.

Case (k = k' + 1)

Assume A: $\langle S_1;\ S_2,\ s\rangle \Rightarrow^{k'+1} s''$
Assume IH: For all S_1, S_2, s and s'', if $\langle S_1;\ S_2,\ s\rangle \Rightarrow^{k'} s''$ then there exists an s', k_1, and k_2 such that $\langle S_1,\ s\rangle \Rightarrow^{k_1} s'$ and $\langle S_2,\ s'\rangle \Rightarrow^{k_2} s''$ and $k' = k_1 + k_2$.
From A we know that there must exist a γ such that
B: $\langle S_1;\ S_2,\ s\rangle \Rightarrow \gamma$ and C: $\gamma \Rightarrow^{k'} s''$.
The only two rules that can create the derivation sequence
$\langle S_1;\ S_2,\ s\rangle \Rightarrow \gamma$ are $[\mathsf{comp}^1_{\mathsf{sos}}]$ and $[\mathsf{comp}^2_{\mathsf{sos}}]$.

Case $[\mathsf{comp}^1_{\mathsf{sos}}]$
Assume D: $\langle S_1,\ s\rangle \Rightarrow \langle S_1',\ s'''\rangle$ and E: $\gamma = \langle S_1';\ S_2,\ s'''\rangle$
From E, C and IH we know that there must exist an s', a k_1 and a k_2 such that F: $\langle S_1',\ s'''\rangle \Rightarrow^{k_1} s'$ and G: $\langle S_2,\ s'\rangle \Rightarrow^{k_2} s''$ and H: $k' = k_1 + k_2$.

From D and F we have that $\langle S_1,\ s\rangle \Rightarrow^{k_1+1} s'$, we have G,
and finally, from H we have that $k = (k_1 + 1) + k_2$.

Case $[\mathsf{comp}^2_{\mathsf{sos}}]$
Assume D: $\langle S_1,\ s\rangle \Rightarrow\ s'''$ and E: $\gamma = \langle S_2,\ s'''\rangle$.

From D we have that $\langle S_1,\ s\rangle \Rightarrow^1 s'''$
From E and C we have that $\langle S_2,\ s'''\rangle \Rightarrow^{k'} s''$
And finally we know that $k = k' + 1$