Semantics and Types - Assignment 6

Mikkel Willén
bmq419

March 19, 2024

Task 7.2

a)

$$\mathcal{C}[\![c_0; (c_1; c_2)]\!] = \mathcal{C}[\![(c_0; c_1); c_2]\!]$$

We have that

$$\lfloor \sigma'_l \rfloor = \mathcal{C}[\![c_0]\!]\sigma \ \star \ \lambda\sigma_1.\mathcal{C}[\![(c_1; c_2)]\!]\sigma_1$$

and

$$\lfloor \sigma'_r \rfloor = \mathcal{C}[\![(c_0; c_1)]\!]\sigma \ \star \ \lambda\sigma_a.\mathcal{C}[\![c_2]\!]\sigma_a$$

where $\lfloor \sigma'_l \rfloor$ is the left hand side and $\lfloor \sigma'_r \rfloor$ is the right hand side. Then expanding the inner most **sequence**-command, we get

$$\lfloor \sigma'_l \rfloor = \mathcal{C}[\![c_0]\!]\sigma \ \star \ \lambda\sigma_1.(\mathcal{C}[\![c_1]\!]\sigma_1 \ \star \ \lambda\sigma_2.\mathcal{C}[\![c_2]\!]\sigma_2)$$

and

$$\lfloor \sigma'_r \rfloor = (\mathcal{C}[\![c_0]\!]\sigma \ \star \ \lambda\sigma_b.\mathcal{C}[\![c_1]\!]\sigma_b) \ \star \ \lambda\sigma_a.\mathcal{C}[\![c_2]\!]\sigma_a$$

For the left hand side, we have that $\mathcal{C}[\![c_0]\!]\sigma = \lfloor \sigma'' \rfloor$, $\mathcal{C}[\![c_1]\!]\sigma_1 = \lfloor \sigma''' \rfloor$ and $\mathcal{C}[\![c_2]\!]\sigma_2 = \lfloor \sigma' \rfloor$. By definition of $\star$, we have that $\sigma'' = \sigma_1$ and $\sigma''' = \sigma_2$ and that $\lfloor \sigma'_l \rfloor = \sigma'$.

For the right hand side, we have that $\mathcal{C}[\![c_0]\!]\sigma = \lfloor \sigma'' \rfloor$, $\mathcal{C}[\![c_1]\!]\sigma_b = \lfloor \sigma''' \rfloor$ and $\mathcal{C}[\![c_2]\!]\sigma_a = \lfloor \sigma' \rfloor$. By definition of $\star$, we have that $\sigma'' = \sigma_b$ and $\sigma''' = \sigma_a$ and that $\lfloor \sigma'_r \rfloor = \sigma'$. We then have that $\lfloor \sigma'_l \rfloor = \lfloor \sigma'_r \rfloor$, which is what we wanted to show.

b)

$$\mathcal{C}[\![(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1); c_2]\!] = \mathcal{C}[\![\textbf{if } b \textbf{ then } (c_0, c_2) \textbf{ else } (c_1; c_2)]\!]$$

We have that

$$\lfloor \sigma'_l \rfloor = \mathcal{C}[\![cond(\mathcal{B}[\![b]\!]\sigma, \mathcal{C}[\![c_0]\!]\sigma, \mathcal{C}[\![c_1]\!]\sigma)]\!]\sigma \star \lambda\sigma_1.C[\![c_2]\!]\sigma_1$$

and

$$\lfloor \sigma'_r \rfloor = cond(\mathcal{B}[\![b]\!]\sigma, \mathcal{C}[\![(c_0; c_2)]\!]\sigma, \mathcal{C}[\![(c_1, c_2)]\!]\sigma)$$

We split it up into to cases, since we know that $\mathcal{B}[\![b]\!]\sigma$ must have one of two truth values.

Suppose $\mathcal{B}[\![b]\!]\sigma = \textbf{true}$
Then we have

$$\begin{aligned}\lfloor \sigma'_l \rfloor &= \mathcal{C}[\![cond(\textbf{true}, \mathcal{C}[\![c_0]\!]\sigma, \mathcal{C}[\![c_1]\!]\sigma)]\!]\sigma \star \lambda\sigma_1.C[\![c_2]\!]\sigma_1 \\ &= \mathcal{C}[\![c_0]\!]\sigma \star \lambda\sigma_1.C[\![c_2]\!]\sigma_1\end{aligned}$$

and

$$\begin{aligned}\lfloor\sigma_r'\rfloor &= cond(\textbf{true}, \mathcal{C}[\![(c_0;c_2)]\!]\sigma, \mathcal{C}[\![(c_1,c_2)]\!]\sigma)\\ &= \mathcal{C}[\![(c_0;c_2)]\!]\sigma\\ &= \mathcal{C}[\![c_0]\!]\sigma \star \lambda\sigma_1.C[\![c_2]\!]\sigma_1\end{aligned}$$

and we see that $\lfloor\sigma_l'\rfloor = \lfloor\sigma_r'\rfloor$

Suppose $\mathcal{B}[\![b]\!]\sigma = \textbf{false}$
Then we have

$$\begin{aligned}\lfloor\sigma_l'\rfloor &= \mathcal{C}[\![cond(\textbf{false}, \mathcal{C}[\![c_0]\!]\sigma, \mathcal{C}[\![c_1]\!]\sigma)]\!]\sigma \star \lambda\sigma_1.C[\![c_2]\!]\sigma_1\\ &= \mathcal{C}[\![c_1]\!]\sigma \star \lambda\sigma_1.C[\![c_2]\!]\sigma_1\end{aligned}$$

and

$$\begin{aligned}\lfloor\sigma_r'\rfloor &= cond(\textbf{false}, \mathcal{C}[\![(c_0;c_2)]\!]\sigma, \mathcal{C}[\![(c_1;c_2)]\!]\sigma)\\ &= \mathcal{C}[\![(c_1;c_2)]\!]\sigma.\\ &= \mathcal{C}[\![c_1]\!]\sigma \star \lambda\sigma_1.C[\![c_2]\!]\sigma_1\end{aligned}$$

and we see that $\lfloor\sigma_l'\rfloor = \lfloor\sigma_r'\rfloor$

c)

$$\mathcal{C}[\![\textbf{while } b \textbf{ do } c_0]\!] = \mathcal{C}[\![\textbf{if } b \textbf{ then } (c_0;\ \textbf{while } b \textbf{ do } c_0) \textbf{ else skip}]\!]$$

We have that

$$\lfloor\sigma_l'\rfloor = fix(\lambda\phi.\lambda\sigma.cond(\mathcal{B}[\![b]\!]\sigma, \mathcal{C}[\![c_0]\!]\sigma \ \star\ \phi, \eta\sigma))$$

$$\lfloor\sigma_r'\rfloor = cond(\mathcal{B}[\![b]\!]\sigma, \mathcal{C}[\![(c_0; fix(\lambda\phi.\lambda\sigma.cond(\mathcal{B}[\![b]\!]\sigma, \mathcal{C}[\![c_0]\!]\sigma \ \star\ \phi, \eta\sigma)))]\!]\sigma, \mathcal{C}[\![\textbf{skip}]\!]\sigma)$$

Suppose $\mathcal{B}[\![b]\!]\sigma = \textbf{false}$
Then we have

$$\begin{aligned}\lfloor\sigma_l'\rfloor &= fix(\lambda\phi.\lambda\sigma.cond(\textbf{false}, \mathcal{C}[\![c_0]\!]\sigma \ \star\ \phi, \eta\sigma))\\ &= fix(\lambda\phi.\lambda\sigma.\eta\sigma)\\ &= \eta\sigma\end{aligned}$$

$$\begin{aligned}\lfloor\sigma_r'\rfloor &= cond(\textbf{false}, \mathcal{C}[\![(c_0; fix(\lambda\phi.\lambda\sigma.cond(\textbf{false}, \mathcal{C}[\![c_0]\!]\sigma \ \star\ \phi, \eta\sigma)))]\!]\sigma, \mathcal{C}[\![\textbf{skip}]\!]\sigma)\\ &= \mathcal{C}[\![\textbf{skip}]\!]\sigma\\ &= \eta\sigma\end{aligned}$$

here we see that $\lfloor\sigma_l'\rfloor = \lfloor\sigma_r'\rfloor$, since both sides end in the unchanged σ.

Suppose $\mathcal{B}[\![b]\!]\sigma = \textbf{true}$
Then we have

$$\begin{aligned}\lfloor\sigma_l'\rfloor &= fix(\lambda\phi.\lambda\sigma.cond(\textbf{true}, \mathcal{C}[\![c_0]\!]\sigma \ \star\ \phi, \eta\sigma))\\ &= fix(\lambda\phi.\lambda\sigma.\mathcal{C}[\![c_o]\!]\sigma \ \star\ \phi)\\ &= fix(\lambda\phi.\lambda\sigma.\mathcal{C}[\![c_o]\!]\sigma \ \star\ fix(\lambda\phi.\lambda\sigma_1.cond(\mathcal{B}[\![b]\!]\sigma_1, \mathcal{C}[\![c_0]\!]\sigma_1 \ \star\ \phi, \eta\sigma_1)))\end{aligned}$$

where we have that $\mathcal{C}[\![c_0]\!]\sigma = \lfloor\sigma''\rfloor$, $\sigma'' = \sigma_1$, $fix(\lambda\phi.\lambda\sigma_1.cond(\mathcal{B}[\![b]\!]\sigma_1, \mathcal{C}[\![c_0]\!]\sigma_1 \ \star\ \phi, \eta\sigma_1))) = \lfloor\sigma'\rfloor$ and $\lfloor\sigma_l'\rfloor = \sigma'$.

$$\begin{aligned}\lfloor\sigma_r'\rfloor &= cond(\textbf{true}, \mathcal{C}[\![(c_0; fix(\lambda\phi.\lambda\sigma.cond(\mathcal{B}[\![b]\!]\sigma, \mathcal{C}[\![c_0]\!]\sigma \ \star\ \phi, \eta\sigma)))]\!]\sigma, \mathcal{C}[\![\textbf{skip}]\!]\sigma)\\ &= \mathcal{C}[\![(c_0; fix(\lambda\phi.\lambda\sigma.cond(\mathcal{B}[\![b]\!]\sigma, \mathcal{C}[\![c_0]\!]\sigma \ \star\ \phi, \eta\sigma)))]\!]\sigma\\ &= \lambda\sigma.(\mathcal{C}[\![c_0]\!]\sigma \ \star\ fix(\lambda\phi.\lambda\sigma_a.cond(\mathcal{B}[\![b]\!]\sigma_a, \mathcal{C}[\![c_0]\!]\sigma_1 \ \star\ \phi, \eta\sigma_a)))\end{aligned}$$

where we have that $\mathcal{C}[\![c_0]\!]\sigma = \lfloor\sigma''\rfloor$ and $\sigma'' = \sigma_a$, $fix(\lambda\phi.\lambda\sigma_a.cond(\mathcal{B}[\![b]\!]\sigma_a, \mathcal{C}[\![c_0]\!]\sigma_1 \ \star\ \phi, \eta\sigma_a))) = \sigma'$ and $\lfloor\sigma_r'\rfloor = \sigma'$ and we have that $\sigma_1 = \sigma_a$, which means that $\lfloor\sigma_l'\rfloor = \lfloor\sigma_r'\rfloor$.

Task 7.3

We extend the language with

$$c :== \dots \mid \textbf{repeat } c_0 \textbf{ until } b$$

The big-step rules for defining the formal semantics of **repeat**-loops are:

$$\text{EC-R\textsc{epeat}T} : \frac{\langle c_0, \sigma\rangle \downarrow \sigma' \quad \langle b, \sigma'\rangle \downarrow \textbf{true}}{\langle \textbf{repeat } c_0 \textbf{ until } b, \sigma\rangle \downarrow \sigma'}$$

$$\text{EC-R\textsc{epeat}F} : \frac{\langle c_0, \sigma\rangle \downarrow \sigma'' \quad \langle b, \sigma''\rangle \downarrow \textbf{false} \quad \langle \textbf{repeat } c_0 \textbf{ until } b, \sigma''\rangle \downarrow \sigma'}{\langle \textbf{repeat } c_0 \textbf{ until } b, \sigma\rangle \downarrow \sigma'}$$

We extend the denotional semantics of commands with:

$$\mathcal{C}[\![\textbf{repeat } c_0 \textbf{ until } b]\!] = \mathcal{C}[\![c_0]\!]\sigma \star fix(\underbrace{\lambda\phi\lambda\sigma_1.cond(\mathcal{B}[\![b]\!]\sigma, \eta\sigma_1, \mathcal{C}[\![c_0]\!]\sigma_1 \star \phi)}_{\Phi:[\Sigma\to\Sigma_\bot]\to[\Sigma\to\Sigma_\bot]})$$

Lemma 7.3 *If* $\langle c, \sigma\rangle \downarrow \sigma'$, *then* $\mathcal{C}[\![c]\!]\sigma = \lfloor\sigma'\rfloor$.

$$\text{Case } \mathcal{E} = \text{EC-R\textsc{epeat}T}\frac{\langle c_0, \sigma\rangle \downarrow \sigma' \quad \langle b, \sigma'\rangle \downarrow \textbf{true}}{\langle \textbf{repeat } c_0 \textbf{ until } b, \sigma\rangle \downarrow \sigma'}$$

We have $c = \textbf{repeat } c_0 \textbf{ until } b$. By IH on $\mathcal{E}_0$, we get $\mathcal{C}[\![c_0]\!]\sigma = \lfloor\sigma'\rfloor$, and by theorem 7.2($\Leftarrow$) on $\mathcal{E}_1$, we get $\mathcal{B}[\![b]\!]\sigma = \textbf{true}$. Thus

$$\begin{aligned}
\mathcal{C}[\![c]\!]\sigma &= \mathcal{C}[\![c_0]\!]\sigma \star fix(\lambda\phi\lambda\sigma_1.cond(\mathcal{B}[\![b]\!]\sigma, \eta\sigma_1, \mathcal{C}[\![c_0]\!]\sigma_1 \star \phi)) \\
&= \lfloor\sigma'\rfloor \star fix(\lambda\phi\lambda\sigma_1.cond(\textbf{true}, \eta\sigma_1, \mathcal{C}[\![c_0]\!]\sigma_1 \star \phi)) \\
&= \eta\sigma' \\
&= \lfloor\sigma'\rfloor
\end{aligned}$$

as required, where $\sigma' = \sigma_1$.

$$\text{Case } \mathcal{E} = \text{EC-R\textsc{epeat}F}\frac{\langle c_0, \sigma\rangle \downarrow \sigma'' \quad \langle b, \sigma''\rangle \downarrow \textbf{false} \quad \langle \textbf{repeat } c_0 \textbf{ until } b, \sigma''\rangle \downarrow \sigma'}{\langle \textbf{repeat } c_0 \textbf{ until } b, \sigma\rangle \downarrow \sigma'}$$

We have $c = \textbf{repeat } c_0 \textbf{ until } b$. By IH on $\mathcal{E}_0$, we get $\mathcal{C}[\![c_0]\!]\sigma = \lfloor\sigma'\rfloor$, by theorem 7.2($\Leftarrow$) on $\mathcal{E}_1$, we get $\mathcal{B}[\![b]\!]\sigma = \textbf{false}$ and by IH on $\mathcal{E}_2$ we get $\mathcal{C}[\![c]\!]\sigma'' = \lfloor\sigma'\rfloor$. Thus

$$\begin{aligned}
\mathcal{C}[\![c]\!]\sigma &= \mathcal{C}[\![c_0]\!]\sigma \star fix(\lambda\phi\lambda\sigma_1.cond(\mathcal{B}[\![b]\!]\sigma, \eta\sigma_1, \mathcal{C}[\![c_0]\!]\sigma_1 \star \phi)) \\
&= \lfloor\sigma'''\rfloor \star fix(\lambda\phi\lambda\sigma_1.cond(\textbf{false}, \eta\sigma_1, \mathcal{C}[\![c_0]\!]\sigma_1 \star \phi)) \\
&= \lfloor\sigma'''\rfloor \star \mathcal{C}[\![c_0]\!]\sigma_1 \star \phi \\
&= \lfloor\sigma''\rfloor \star \phi \\
&= \phi\sigma'' \\
&= \mathcal{C}[\![c]\!]\sigma'' \\
&= \lfloor\sigma'\rfloor
\end{aligned}$$

as required, where $\sigma' = \sigma_1$.

Lemma 7.4 *If* $\mathcal{C}[\![c]\!]\sigma = \lfloor\sigma'\rfloor$, *then* $\langle c, \sigma\rangle \downarrow \sigma'$.

$$\text{Case } c = \textbf{repeat } c_0 \textbf{ until } b$$

We can show this by an inner fixpoint induction, where most of it is analogous to the **while** case. For the inductive step, we have

$$\mathcal{C}[\![c_0]\!]\sigma'' \star \lambda\sigma'''.cond(\mathcal{B}[\![b]\!]\sigma''', \eta\sigma''', \mathcal{C}[\![c_0]\!]\sigma''' \star \phi) = \lfloor\sigma'\rfloor \Rightarrow \langle c, \sigma''\rangle \downarrow \sigma'$$

If we assume the LHS of the implication holds, then there are two subcases.

$$\text{Case } \mathcal{B}[\![b]\!]\sigma''' = \mathbf{true}$$

By outer IH on c_0 we get a derivation $\mathcal{E}_0$ of $\langle c_0, \sigma''\rangle \downarrow \sigma'''$, and by Theorem 7.2($\Rightarrow$) on b we get a derivation $\mathcal{E}_1$ of $\langle b, \sigma'''\rangle \downarrow \mathbf{true}$. Putting $\mathcal{E}_0$ and $\mathcal{E}_1$ together with EC-RepeatT we get the required derivation of $\langle c, \sigma''\rangle \downarrow \sigma'$ with $\sigma' = \sigma'''$.

$$\text{Case } \mathcal{B}[\![b]\!]\sigma''' = \mathbf{false}$$

By Theorem 7.2($\Rightarrow$) on b we get a derivation $\mathcal{E}_1$ of $\langle b, \sigma'''\rangle \downarrow \mathbf{false}$. Now by the LHS equation, we get that $\mathcal{C}[\![c_0]\!]\sigma \star \phi = \lfloor \sigma' \rfloor$, which can only happen, if (1) $\mathcal{C}[\![c_0]\!]\sigma'' = \lfloor \sigma''' \rfloor$ for some σ''' and (2) $\phi\sigma''' = \lfloor \sigma' \rfloor$. By outer IH on c_0 we get a derivation $\mathcal{E}_0$ of $\langle c_0, \sigma''\rangle \downarrow \sigma'''$, and by the fixpoint IH, and from (2), we get an $\mathcal{E}_2$ of $\langle c, \sigma'''\rangle \downarrow \sigma'$. Putting $\mathcal{E}_0$, $\mathcal{E}_1$ and $\mathcal{E}_2$ together with EC-RepeatF we get the required derivation of $\langle c, \sigma''\rangle \downarrow \sigma'$.

Task 3

We consider an extenstion of IMP with guarded conditionals:

$$c ::= \ldots \mid \mathbf{if}\ b_1 \to c_1, \ldots, b_k \to c_k\ \mathbf{fi}\ (k \geq 1)$$

The semantics of guarded conditionals is given by the following big-step rule:

$$\text{EC-GC} : \frac{\langle b_i, \sigma\rangle \downarrow \mathbf{true} \quad \langle c_i, \sigma\rangle \downarrow \sigma'}{\langle \mathbf{if}\ b_1 \to c_1, \ldots, b_k \to c_k\ \mathbf{fi}, \sigma\rangle \downarrow \sigma'}\ (1 \leq i \leq k)$$

a)

The Hoare-logic rule is given as:

$$\text{H-GC} : \frac{\{A \wedge b_1\}c_1\{B\} \quad \ldots \quad \{A \wedge b_k\}c_k\{B\}}{\{A \vee \neg(b_1 \vee \ldots \vee b_k)\}\ \mathbf{if}\ b_1 \to c_1, \ldots, b_k \to c_k\ \mathbf{fi}\ \{B\}}$$

Theorem 3.7 *If* $\overset{\mathcal{H}}{\vdash \{A\}c\{B\}}$, *then* $\vDash \{A\}c\{B\}$.

$$\text{Case } \mathcal{H} = \text{H-GC}\frac{\overset{\mathcal{H}_1}{\{A \wedge b_1\}c_1\{B\}} \quad \ldots \quad \overset{\mathcal{H}_k}{\{A \wedge b_k\}c_k\{B\}}}{\{A \vee \neg(b_1 \vee \ldots \vee b_k)\}\ \mathbf{if}\ b_1 \to c_1, \ldots, b_k \to c_k\ \mathbf{fi}\ \{B\}}$$

so $c = (\mathbf{if}\ b_1 \to c_1, \ldots, b_k \to c_k\ \mathbf{fi})$. We have two cases. The first, where one or more $b_i \in [1 \ldots k]$ evalute to true, or the second, where none of them do. In the first case, we have:

$$\mathcal{E} = \text{EC-GC}\frac{\overset{\mathcal{E}_{i,0}}{\langle b_i, \sigma\rangle \downarrow \mathbf{true}} \quad \overset{\mathcal{E}_{i,1}}{\langle c_i, \sigma\rangle \downarrow \sigma'}}{\langle \mathbf{if}\ b_1 \to c_1, \ldots, b_k \to c_k\ \mathbf{fi}, \sigma\rangle \downarrow \sigma'}\ (1 \leq i \leq k)$$

By Lemma 3.1 ($\Leftarrow$) on $\mathcal{E}_{i,0}$ we have $\sigma \vDash b$, and we have $\sigma \vDash A \wedge b$, and thus by IH on $\mathcal{H}_i$ with $\mathcal{E}_{i,1}$, we get $\sigma' \vDash B$.

In the other case, we have that all $b_i \in [1 \ldots k]$ evalute to false, and thus by Lemma 3.1 ($\Rightarrow$) on all $\mathcal{E}_{i,0}$ for $i \in [1 \ldots k]$ we have that $\sigma \nvDash b_i$, so $\sigma \vDash \neg b_1 \wedge \ldots \wedge \neg b_k$ or $\sigma \vDash \neg(b_1 \vee \ldots \vee b_k)$, so it initially holds. But, since if all guards evalute to false, it does not terminate, we can't say anything about the end state.

b)

We give a denotational semantic for guarded conditionals:

$$\mathcal{C}[\![\mathbf{if}\ b_1 \to c_1, \ldots, b_k \to c_k\ \mathbf{fi}]\!] = \lambda\sigma.cond(\mathcal{B}[\![b_1]\!], \mathcal{C}[\![c_1]\!]\sigma, cond(\ldots, cond(\mathcal{B}[\![b_k]\!], \mathcal{C}[\![c_k]\!]\sigma, err)\ldots)$$

Where $\ldots$ are all the cases between 1 and k (and closing parenthesis in the end), and err is some error, signaling that the program does not terminate.

Lemma 7.3 *If* $\langle c, \sigma \rangle \downarrow \sigma'$*, then* $\mathcal{C}[\![c]\!]\sigma = \lfloor \sigma' \rfloor$. We have two cases. The first, where one or more $b_i \in [1 \dots k]$ evalute to true, or the second, where none of them do. In the first case, we have:

$$\text{Case } \mathcal{E} = \text{EC-GC} \frac{\overset{\mathcal{E}_{i,0}}{\langle b_i, \sigma \rangle \downarrow \mathbf{true}} \quad \overset{\mathcal{E}_{i,1}}{\langle c_i, \sigma \rangle \downarrow \sigma'}}{\langle \mathbf{if}\ b_1 \to c_1, \dots, b_k \to c_k\ \mathbf{fi}, \sigma \rangle \downarrow \sigma'}\ (1 \leq i \leq k)$$

We have $c = (\mathbf{if}\ b_1 \to c_1, \dots, b_k \to c_k\ \mathbf{fi})$, and so by Theorem 7.2 ($\Leftarrow$) on $\mathcal{E}_{i,0}$ we get $\mathcal{B}[\![b_i]\!]\sigma = \mathbf{true}$, and by IH on $\mathcal{E}_{i,1}$, we get $\mathcal{C}[\![c_i]\!]\sigma = \lfloor \sigma' \rfloor$. Thus $\mathcal{C}[\![c]\!]\sigma = \lambda\sigma.\mathit{cond}(\mathcal{B}[\![b_1]\!], \mathcal{C}[\![c_1]\!]\sigma, \mathit{cond}(\dots, \mathit{cond}(\mathcal{B}[\![b_k]\!], \mathcal{C}[\![c_k]\!]\sigma, \mathit{err}) \dots) = \mathit{cond}(\mathbf{true}, \mathcal{C}[\![c_i,]\!]\sigma, \dots) = \mathcal{C}[\![c_i]\!]\sigma = \lfloor \sigma' \rfloor$ as required.

In the case, where no b_i for $i \in [1 \dots k]$ evaluate to true, we might have a problem. If the program does not terminate, we cannot create a derivation of execution. If we just move out of the guarded conditional case, we have that $\sigma' = \sigma$, and then $\mathcal{C}[\![c]\!]\sigma = \mathcal{C}[\![\mathbf{skip}]\!]\sigma = \eta\sigma = \lfloor \sigma \rfloor = \lfloor \sigma' \rfloor$, and we are done.

Lemma 7.4 *If* $\mathcal{C}[\![c]\!]\sigma = \lfloor \sigma' \rfloor$*, then* $\langle c, \sigma \rangle \downarrow \sigma'$.

$$\text{Case } c = \mathbf{if}\ b_1 \to c_1, \dots, b_k \to c_k\ \mathbf{fi}$$

We have $\lfloor \sigma' \rfloor = \mathcal{C}[\![\mathbf{if}\ b_1 \to c_1, \dots, b_k \to c_k\ \mathbf{fi}]\!]\sigma = \lambda\sigma.\mathit{cond}(\mathcal{B}[\![b_1]\!], \mathcal{C}[\![c_1]\!]\sigma, \mathit{cond}(\dots, \mathit{cond}(\mathcal{B}[\![b_k]\!], \mathcal{C}[\![c_k]\!]\sigma, \mathit{err}) \dots)$. We now have to cases, the first, where one or more $b_i \in [1 \dots k]$ evaluate to true, or the second, where none of them do. In the first case, we have, that some b_i for $i \in [1 \dots k]$ evaluates to **true**, and thus if we suppose $\mathcal{B}[\![b_i]\!]\sigma = \mathit{true}$ (and that no b_j evaluates to **true** for some $j < i$), then by Theorem 7.2 ($\Rightarrow$) we get a derivation $\mathcal{E}_{i,0}$ of $\langle b_i, \sigma \rangle \downarrow \mathbf{true}$. Then we also have $\lfloor \sigma' \rfloor = \mathit{cond}(\mathbf{true}, \mathcal{C}[\![c_i,]\!]\sigma, \dots) = \mathcal{C}[\![c_i]\!]\sigma$ and thus by IH on c_i we get an $\mathcal{E}_{i,1}$ of $\langle c_i, \sigma \rangle \downarrow \sigma$ and by EC-GC on $\mathcal{E}_{i,0}$ and $\mathcal{E}_{i,1}$ we get the required derivation of $\langle \mathbf{if}\ b_1 \to c_1, \dots, b_k \to c_k\ \mathbf{fi}, \sigma \rangle \downarrow \sigma'$.

In the case, where no b_i for $i \in [1 \dots k]$ evaluate to true, we might have a problem. If the program does not terminate, we cannot create a derivation of execution. If we just move out of the guarded conditional case, we have that $\sigma = \sigma'$ and thus we can use EC-Skip to get the required derivation.