Semantics and Types - Exam 2024

Mikkel Willén
bmq419
Eksamensnummer: 7

April 3, 2024

Task 1

Task 1.1

a)

Lemma 2 *Let p be such that all l_j declared in p have $j < i$, and suppose $[\![c]\!]$ @ $p \rightsquigarrow^i_{i'} p'$ (by $\mathcal{C}$), and $p_0 \sqsupseteq p'$. If $\langle c, \sigma\rangle \downarrow \sigma''$ (by $\mathcal{E}$), and $p_0 \vdash \langle p, \sigma''\rangle \downarrow \sigma'$ (by $\mathcal{P}$), then $p_0 \vdash \langle p', \sigma\rangle \downarrow \sigma'$ (by some $\mathcal{P}'$).*

Proof. By induction on the derivation $\mathcal{E}$. We show the cases for EC-Skip, EC-Seq, EC-IfF, EC-WhileF and EC-WhileT.

$$\text{Case } \mathcal{E} = \text{EC-Skip}\frac{}{\langle \textbf{skip}, \sigma\rangle \downarrow \sigma}$$

so we have $c = \textbf{skip}$ and $\sigma'' = \sigma$. In this case the derivation must look as follows:

$$\mathcal{C} = \text{C-Skip}\frac{}{[\![\textbf{skip}]\!] \text{ @ } p \rightsquigarrow^i_{i'} p}$$

and thus $p' = p$. Since $\mathcal{P}$ is then a derivation of $p_0 \vdash \langle p, \sigma\rangle \downarrow \sigma'$, we can take $\mathcal{P}' = \mathcal{P}$ directly.

$$\text{Case } \mathcal{E} = \text{EC-Seq}\frac{\overset{\mathcal{E}_0}{\langle c_0, \sigma\rangle \downarrow \sigma'''} \quad \overset{\mathcal{E}_1}{\langle c_1, \sigma'''\rangle \downarrow \sigma''}}{\langle (c_0; c_1), \sigma\rangle \downarrow \sigma''}$$

so we have $c = (c_0; c_1)$ and we must have

$$\mathcal{C} = \text{C-Seq}\frac{\overset{\mathcal{C}_0}{[\![c_1]\!] \text{ @ } p \rightsquigarrow^i_{i''} p''} \quad \overset{\mathcal{C}_1}{[\![c_0]\!] \text{ @ } p'' \rightsquigarrow^{i''}_{i'} p'}}{[\![c_0; c_1]\!] \text{ @ } p \rightsquigarrow^i_{i'} p'}$$

Let $\mathcal{P}_1$ of $p_0 \vdash \langle p'', \sigma\rangle \downarrow \sigma'''$ and $\mathcal{P}_0$ of $p_1 \vdash \langle p, \sigma'''\rangle \downarrow \sigma'$. By IH on $\mathcal{E}_1$ with $\mathcal{C}_0$ and $\mathcal{P}_0$ we get a derivation $\mathcal{P}'_0$ of $p_1 \vdash \langle p'', \sigma'''\rangle \downarrow \sigma'$, and by IH on $\mathcal{E}_0$ with $\mathcal{C}_1$ and $\mathcal{P}_1$ we get a derivation $\mathcal{P}'_1$ of $p_0 \vdash \langle p', \sigma''\rangle \downarrow \sigma'''$. Thus we obtain $\mathcal{P}'$ by first prepending c_1 to p, where we get p'' and then prepending c_0 to p'' to get p', where $p_0 \sqsupseteq p'' \sqsupseteq p'$.

$$\text{Case } \mathcal{E} = \text{EC-IfF}\frac{\overset{\mathcal{E}_0}{\langle b, \sigma\rangle \downarrow \textbf{false}} \quad \overset{\mathcal{E}_1}{\langle c_1, \sigma\rangle \downarrow \sigma''}}{\langle \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma\rangle \downarrow \sigma''}$$

so we have $c = \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1$ and we must have

$$\mathcal{C} = \text{C-If}\frac{\overset{\mathcal{C}_0}{[\![c_0]\!] \text{ @ } l_{i+1} : p \rightsquigarrow^{i+2}_{i''} p''} \quad \overset{\mathcal{C}_1}{[\![c_1]\!] \text{ @ } \textbf{goto } l_{i+1}; l_i : p'' \rightsquigarrow^{i''}_{i'} p'''}}{[\![\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1]\!] \text{ @ } p \rightsquigarrow^i_{i'} \textbf{if } b \textbf{ then goto } l_i; p'''}$$

so $p' = \textbf{if } b \textbf{ then goto } l_i; p'''$. Let $\mathcal{P}_1$ of $p_0 \vdash \langle p_1, \sigma''\rangle \downarrow \sigma'$ with $p_1 = \textbf{goto } l_{i+1}; l_i : p''$ where $p_0 \sqsupseteq p'' \sqsupseteq l_{i+1} : p$. By IH on $\mathcal{E}_1$ with $\mathcal{C}_1$ and $\mathcal{P}_1$ we get a derivation $\mathcal{P}'_1$ of $p_0 \vdash \langle p''', \sigma\rangle \downarrow \sigma'$, and thus, we can take

$$\mathcal{P}' = \text{P-IfGotoF}\frac{\overset{\mathcal{E}_0}{\langle b, \sigma\rangle \downarrow \textbf{false}} \quad \overset{\mathcal{P}'_1}{p_0 \vdash \langle p''', \sigma\rangle \downarrow \sigma'}}{p_0 \vdash \langle \textbf{if } b \textbf{ then goto } l_i; p''', \sigma\rangle \downarrow \sigma'}$$

$$\text{Case } \mathcal{E} = \text{EC-WhileF}\dfrac{\overset{\mathcal{E}_0}{\langle b,\sigma\rangle \downarrow \mathbf{false}}}{\langle \mathbf{while}\ b\ \mathbf{do}\ c_0\rangle \downarrow \sigma}$$

so $c = \mathbf{while}\ b\ \mathbf{do}\ c_0$ and we must have

$$\mathcal{C} = \text{C-While}\dfrac{\overset{\mathcal{C}_0}{[\![c_0]\!] \mathbin{@} \mathbf{goto}\ l_i; l_{i+1} : p \rightsquigarrow^{i+2}_{i'} p''}}{[\![\mathbf{while}\ b\ \mathbf{do}\ c_0 \mathbin{@} p \rightsquigarrow^{i}_{i'} l_i : \mathbf{if}\ \neg b\ \mathbf{then}\ \mathbf{goto}\ l_{i+1}; p''}$$

so $p' = l_i : \mathbf{if}\ \neg b\ \mathbf{then}\ \mathbf{goto}\ l_{i+1}; p''$ and we have $\sigma'' = \sigma$, since c_0 is never executed. Since $\mathcal{P}$ is then a derivation of $p_0 \vdash \langle p, \sigma\rangle \downarrow \sigma'$ we have $p'' = p$ and we can take $\mathcal{P}' = \mathcal{P}$ directly.

$$\text{Case } \mathcal{E} = \text{EC-WhileT}\dfrac{\overset{\mathcal{E}_0}{\langle b,\sigma\rangle \downarrow \mathbf{true}} \quad \overset{\mathcal{E}_1}{\langle c_0,\sigma\rangle \downarrow \sigma''} \quad \overset{\mathcal{E}_2}{\langle \mathbf{while}\ b\ \mathbf{do}\ c_0, \sigma''\rangle \downarrow \sigma'}}{\langle \mathbf{while}\ b\ \mathbf{do}\ c_0, \sigma\rangle \downarrow \sigma'}$$

so $c = \mathbf{while}\ b\ \mathbf{do}\ c_0$ and we must have

$$\mathcal{C} = \text{C-While}\dfrac{\overset{\mathcal{C}_0}{[\![c_0]\!] \mathbin{@} \mathbf{goto}\ l_i; l_{i+1} : p \rightsquigarrow^{i+2}_{i'} p''}}{[\![\mathbf{while}\ b\ \mathbf{do}\ c_0]\!] \mathbin{@} p \rightsquigarrow^{i}_{i'} l_i : \mathbf{if}\ \neg b\ \mathbf{then}\ \mathbf{goto}\ l_{i+1}; p''}$$

so $p' = l_i : \mathbf{if}\ \neg b\ \mathbf{then}\ \mathbf{goto}\ l_{i+1}; p''$. Let $\mathcal{P}_1$ be $p_0 \vdash \langle p_1, \sigma''\rangle \downarrow \sigma'$ with $p_1 = \mathbf{goto}\ l_i; l_{i+1} : p$ where $\mathbf{goto}\ l_i; l_{i+1} : p \sqsupseteq l_{i+1} : p$. By IH on $\mathcal{E}_1$ with $\mathcal{C}_1$ and $\mathcal{P}_1$ we get a derivation $\mathcal{P}_1'$ of $p_0 \vdash \langle p'', \sigma\rangle \downarrow \sigma'$, and thus, we can take

$$\mathcal{P}' = \text{P-IfGotoF}\dfrac{\text{EC-NegT}\dfrac{\overset{\mathcal{E}_0}{\langle b,\sigma\rangle \downarrow \mathbf{true}}}{\langle \neg b,\sigma\rangle \downarrow \mathbf{false}} \quad \overset{\mathcal{P}_1'}{p_0 \vdash \langle p'', \sigma\rangle \downarrow \sigma'}}{p_0 \vdash \langle \mathbf{if}\ \neg b\ \mathbf{then}\ \mathbf{goto}\ l_{i+1}; p'', \sigma\rangle \downarrow \sigma'}$$

b)

Theorem 1. *Suppose* $[\![c]\!]^{\text{top}} \rightsquigarrow p$*. If* $\langle c, \sigma\rangle \downarrow \sigma'$ *then* $p \vdash \langle p, \sigma\rangle \downarrow \sigma'$*.*

Proof. By induction on the translation derivation.

$$\text{Case } \mathcal{C} = \text{C-Skip}\dfrac{}{[\![\mathbf{skip}]\!] \mathbin{@} p \rightsquigarrow^{i}_{i} p}$$

so $c = \mathbf{skip}$ and we must have

$$\mathcal{E} = \text{EC-Skip}\dfrac{}{\langle \mathbf{skip}, \sigma\rangle \downarrow \sigma}$$

so $\sigma = \sigma$, so the case is vacuously true.

$$\text{Case } \mathcal{C} = \text{C-Assign}\dfrac{}{[\![X := a]\!] \mathbin{@} p \rightsquigarrow^{i}_{i} X := a; p}$$

so $c = (X := a)$, and $p = (X := a; p)$ and we must have

$$\mathcal{E} = \text{EC-Assign}\dfrac{\overset{\mathcal{E}_0}{\langle a,\sigma\rangle \downarrow n}}{\langle X := a, \sigma\rangle \downarrow \sigma[X \mapsto n]}$$

so $\sigma' = \sigma[X \mapsto n]$. We observe that $X := a; p \sqsupseteq p$ and thus we have that $X := a; p \vdash \langle X := a; p\rangle \downarrow \sigma[X \mapsto n]$ and

$$\mathcal{P} = \text{P-Assign}\dfrac{\overset{\mathcal{E}_0}{\langle a,\sigma\rangle \downarrow n} \quad \overset{\mathcal{P}_0}{p_0 \vdash \langle p_1, \sigma[X \mapsto n]\rangle \downarrow \sigma'}}{p_0 \vdash \langle X := a; p_1, \sigma\rangle \downarrow \sigma'}$$

$$\text{Case } \mathcal{C} = \text{C-Seq}\dfrac{\overset{\mathcal{C}_0}{[\![c_1]\!] \mathbin{@} p \rightsquigarrow^{i}_{i''} p''} \quad \overset{\mathcal{C}_1}{[\![c_0]\!] \mathbin{@} p'' \rightsquigarrow^{i''}_{i'} p'}}{[\![c_0; c_1]\!] \mathbin{@} p \rightsquigarrow^{i}_{i'} p'}$$

so $c = (c_0; c_1)$ and we must have

$$\mathcal{E} = \text{EC-S{\scriptsize EQ}}\frac{\overset{\mathcal{E}_0}{\langle c_0, \sigma\rangle \downarrow \sigma''} \quad \overset{\mathcal{E}_1}{\langle c_1, \sigma''\rangle \downarrow \sigma'}}{\langle c_0; c_1, \sigma\rangle \downarrow \sigma'}$$

By IH on $\mathcal{C}_0$ with $\mathcal{E}_1$ we get $p'' \vdash \langle c_0, \sigma''\rangle \downarrow \sigma'$, and by IH on $\mathcal{C}_1$ with $\mathcal{E}_0$ we get $p' \vdash \langle c_1, \sigma\rangle \downarrow \sigma''$, and then since $p \sqsupseteq p'' \sqsupseteq p'$ and thus $p \vdash \langle c_0; c_1, \sigma\rangle \downarrow \sigma'$.

$$\text{Case } \mathcal{C} = \text{C-I{\scriptsize F}}\frac{\overset{\mathcal{C}_0}{[\![c_0]\!] \mathbin{@} l_{i+1} : p \leadsto^{i+2}_{i''} p''} \quad \overset{\mathcal{C}_1}{[\![c_1]\!] \mathbin{@} \mathbf{goto}\ l_{i+1}; l_i : p'' \leadsto^{i''}_{i'} p'''}}{[\![\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1]\!] \mathbin{@} p \leadsto^{i}_{i'} \mathbf{if}\ b\ \mathbf{then}\ \mathbf{goto}\ l_i; p'''}$$

so $c = \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1$, so since **if**-commands can have two rules, $\mathcal{E}$ can take one of the following forms:

$$\text{Subcase } \mathcal{E} = \text{EC-I{\scriptsize F}T}\frac{\overset{\mathcal{E}_0}{\langle b, \sigma\rangle \downarrow \mathbf{true}} \quad \overset{\mathcal{E}_1}{\langle c_0, \sigma\rangle \downarrow \sigma''}}{\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle \downarrow \sigma''}$$

By IH on $\mathcal{C}_0$ with $\mathcal{E}_1$ we get $p'' \vdash \langle c_0, \sigma\rangle \downarrow \sigma'$ and thus $p \vdash \langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle \downarrow \sigma'$ since

$$p \sqsupseteq p''' \sqsupseteq l_i : p'' \sqsupseteq p''$$

and

$$P_0 = \text{P-L{\scriptsize AB}}\frac{p_0 \vdash \langle p, \sigma''\rangle \downarrow \sigma'}{p_0 \vdash \langle l_i : p, \sigma''\rangle \downarrow \sigma'}$$

$$\text{Subcase } \mathcal{E} = \text{EC-I{\scriptsize F}F}\frac{\overset{\mathcal{E}_0}{\langle b, \sigma\rangle \downarrow \mathbf{false}} \quad \overset{\mathcal{E}_1}{\langle c_1, \sigma\rangle \downarrow \sigma'}}{\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle \downarrow \sigma'}$$

By IH on $\mathcal{C}_1$ with $\mathcal{E}_1$ we get $p''' \vdash \langle c_1, \sigma\rangle \downarrow \sigma'$ for $p'' = p$, and thus $p \vdash \langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle \downarrow \sigma'$, since $p \sqsupseteq p'''$.

$$\text{Case } \mathcal{C} = \text{C-W{\scriptsize HILE}}\frac{\overset{\mathcal{C}_0}{[\![c_0]\!] \mathbin{@} \mathbf{goto}\ l_i; l_{i+1} : p \leadsto^{i+2}_{i'} p''}}{[\![\mathbf{while}\ b\ \mathbf{do}\ c_0]\!] \mathbin{@} p \leadsto^{i}_{i'} l_i : \mathbf{if}\ \neg b\ \mathbf{then}\ \mathbf{goto}\ l_{i+1}; p''}$$

so $c = \mathbf{while}\ b\ \mathbf{do}\ c_0$ and since there are two rules for **while**-commands, $\mathcal{E}$ can take one of the following forms:

$$\text{Subcase } \mathcal{E} = \text{EC-W{\scriptsize HILE}F}\frac{\overset{\mathcal{E}_0}{\langle b, \sigma\rangle \downarrow \mathbf{false}}}{\langle \mathbf{while}\ b\ \mathbf{do}\ c_0\rangle \downarrow \sigma}$$

so $\sigma = \sigma$, so the case is vacuously true.

$$\text{Subcase } \mathcal{E} = \text{EC-W{\scriptsize HILE}T}\frac{\overset{\mathcal{E}_0}{\langle b, \sigma\rangle \downarrow \mathbf{true}} \quad \overset{\mathcal{E}_1}{\langle c_0, \sigma\rangle \downarrow \sigma''} \quad \overset{\mathcal{E}_2}{\langle \mathbf{while}\ b\ \mathbf{do}\ c_0, \sigma''\rangle \downarrow \sigma'}}{\langle \mathbf{while}\ b\ \mathbf{do}\ c_0, \sigma\rangle \downarrow \sigma'}$$

By IH on $\mathcal{C}_0$ with $\mathcal{E}_1$ we get $p'' \vdash \langle c_0, \sigma\rangle \downarrow \sigma'$. We have that the execution of $\mathcal{E}_2$ is already prepended to the program p in $\mathcal{C}_0$, which gives some intermediate program p''', and thus $p \vdash \langle \mathbf{while}\ b\ \mathbf{do}\ c_0, \sigma\rangle \downarrow \sigma'$, since $p \sqsupseteq p'' \sqsupseteq p'''$.

Task 2

Task 2.1

We show $\Delta / POST \vdash \{PRE\} S$.

$\{x = n\}$
$\{0 + x \times (x+1)/2 = n \times (n+1)/2\} \quad \dagger_1$
$s := 0;$
$\{s + x \times (x+1)/2 = n \times (n+1)/2\}$

2 *loop* :
$\{s + x \times (x+1)/2 = n \times (n+1)/2\}$
$\{(s+x) + (x-1)((x-1)+1)/2 = n \times (n+1)/2\}$ $\dagger_2$
3 $s := s + x$
$\{s + (x-1)((x-1)+1)/2 = n \times (n+1)/2\}$
4 `if` $x = 1$ `then goto` *done*;
$\{s + (x-1)((x-1)+1)/2 = n \times (n+1)/2\}$
5 $x := x - 1$;
$\{s + x \times (x+1)/2\}$
6 `goto` *loop*;
7 *done* :
$\{s = n \times (n+1)/2\}$
$\{s + s = n \times (n+1)\}$ $\dagger_3$
8 $s := s + s$;
9 `halt`
$\{s = n \times (n+1)\}$

We will now prove each of the semantic reasonings marked with a †.

$\dagger_1 : \overbrace{x = n}^{(1)} \Rightarrow \overbrace{0 + x \times (x+1)/2 = n \times (n+1)/2}^{(a)}$
Here we get (a) directly from (1) since $x = n$, and thus $0 + x \times (x+1)/2 = x \times (x+1)/2 = n \times (n+1)/2$.

$\dagger_2 : \overbrace{s + x \times (x+1)/2 = n \times (n+1)/2}^{(1)} \Rightarrow \overbrace{(s+x) + (x-1)((x-1)+1)/2 = n \times (n+1)/2}^{(a)}$
Here we get (a) directly from (1) since $(s+x)(x-1)((x-1)+1)/2 = (s+x)(x-1)((x-1)+1)/2 = (s+x)(x-1)((x-1)+1)/2 = (s+x)(x-1)(x)/2 = s + x \times (x+1)/2 = n \times (n+1)$.

$\dagger_3 : \overbrace{s = n \times (n+1)/2}^{(1)} \Rightarrow \overbrace{s + s = n \times (n+1)}^{(a)}$
Here we get (a) directly from (1) since $s + s = 2s$, and thus if we multiply both sides of (1) with 2 we get (a).

We will now show how to convert to a formal derivation, starting from line 9 and working backwards, with $PRE = \{x = n\}$ and $POST = \{s = n \times (n+1)\}$.
Line 9: **halt** has the postcondition of $POST$, so by rule W-Halt, we get $\Delta/POST \Vdash \{POST\}\mathbf{halt}$.
Line 8: For $s := s + s$, by rule W-Assign we need a precondition A such that $A \rightsquigarrow s = n \times (n+1)/2$. By the assignment axiom, the precondition that ensures this is $\Delta(done)$, and thus $\Delta/POST \Vdash \{\Delta(done)\}s := s + s$.
Line 7: For the label *done*, by rule W-Lab we get $\Delta/POST \Vdash \{\Delta(done)\}done : s := s + s$.
Line 6: The precondition of **goto** *loop* is the invariant $\Delta(loop)$, and so by rule W-Goto, we get $\Delta/POST \Vdash \{\Delta(loop)\}\mathbf{goto}\ loop$.
Line 5: By rule W-Assign, we need $s + (x-1)((x-1)+1)/2 = n(n+1)/2$, which simplifies to $\Delta(loop)$, and thus $\Delta/POST \Vdash \{\Delta(loop)\}x := x - 1; \mathbf{goto}\ loop;$.
Line 4: By rule W-IfGoto, we must ewnsure that both branches of the conditional preserve the invariant. The **true** branch requires the invariant at *done*, which is $\Delta(done)$. The **false** branch continues to line 5, which requires the invariant $\Delta(loop)$, and thus $\Delta/POST \Vdash \{\Delta(loop) \wedge (x \neq 1 \Rightarrow \Delta(loop) \wedge (x = 1 \Rightarrow \Delta(done))\}\mathbf{if}\ x = 1\ \mathbf{then\ goto}\ done;$
Line 1-3: To maintain the invariant, we need the state before the addition to satisfy $s + x \times (x+1)/2 = n \times (n+1)/2$, which simplifies to $s = 0$. This is precisely the action performed at line 1, and thus by W-Assign, we get $\Delta/POST \Vdash \{PRE\}s := 0$

Task 2.2

a)

Lemma 4. (Label lookup) *If* $\Delta/B \Vdash \{A_0\}p_0$ *and* $p_0 \vdash l \Downarrow p$, then $\Delta/B \vdash \{A\}p$.

Proof. By induction on judgement for label lookup.

$$\text{Case } \mathcal{L} = \textsc{L-Labs}\,\frac{}{l : p_1 \vdash l \Downarrow p_1}$$

Here $p_0 = l : p_1$ and $p = p_1$. Since $\Delta/B \Vdash \{A_0\}p_0$, the assertion $\Delta(l)$ must hold, and thus $\Delta/B \vdash \{\Delta(l)\}p_0$ by rule V-Weak with $A = \Delta(l)$ and $A' = A_0$.

$$\text{Case } \mathcal{L} = \text{L-LabD}\frac{\overset{\mathcal{L}_0}{p_1 \vdash l \Downarrow p}}{l' : p_1 \vdash l \Downarrow p}\ (l' \neq l)$$

Here $p_0 = l' : p_1$. By IH on $\mathcal{L}_0$ with $\Delta/B \Vdash \{A_0\}p_1$, we get $\Delta/B \vdash \{\Delta(l)\}p$. Since $l' \neq l$ the annotation Δ for l in p_1 is the same as in p_0, and thus $\Delta/B \vdash \{\Delta(l)\}p$ holds.

$$\text{Case } \mathcal{L} = \text{L-Labs}\frac{\overset{\mathcal{L}_0}{p_1 \vdash l \Downarrow p}}{s; p_1 \vdash l \Downarrow p}$$

Here $p_0 = s; p_1$. By IH on $\mathcal{L}_0$ with $\Delta/B \Vdash \{A_0\}p_1$, we get $\Delta/B \vdash \{\Delta(l)\}p$. Since the simple command s does not affect the label lookup, the annotation for l ramains uncanged, and $\Delta/B \vdash \{\Delta(l)\}p$ holds.

b)

Lemma 5. (Soundness for fragments) *If $\Delta/B \Vdash \{A_0\}p_0$, and $\Delta/B \vdash \{A\}p$, then $p_0 \vDash \{A\}p\{B\}$.*

Proof. Let $\mathcal{W}_0$ be the derivation of $\Delta/B \Vdash \{A_0\}p_0$, and $\mathcal{V}$ of $\Delta/B \vdash \{A\}p$. To show $p_0 \vDash \{A\}p\{B\}$, let σ and σ' be given, with $\sigma \vDash A$ and $p_0 \vdash \langle p, \sigma\rangle \downarrow \sigma'$ by some $\mathcal{P}$; we must show $\sigma' \vDash B$. The proof is by induction on $\mathcal{P}$.

$$\text{Case } \mathcal{P} = \text{P-Halt}\frac{}{p_0 \vdash \langle \textbf{halt}, \sigma\rangle \downarrow \sigma}$$

Here $P = \textbf{halt}$ and $\sigma' = \sigma$, so $\mathcal{W}$ must have the following shape

$$\mathcal{W} = \text{W-Halt}\frac{}{\Delta/B \Vdash \{B\}\textbf{halt}}$$

Since $\sigma' = \sigma$ and $\sigma \vDash B$ the case is vacuously true.

$$\text{Case } \mathcal{P} = \text{P-Lab}\frac{\overset{\mathcal{P}_0}{p_0 \vdash \langle p_1, \sigma\rangle \downarrow \sigma'}}{p_0 \vdash \langle l : p_1, \sigma\rangle \downarrow \sigma'}$$

Here $p = l : p_1$, so $\mathcal{W}$ must have the following shape

$$\mathcal{W} = \text{W-Lab}\frac{\overset{\mathcal{W}_0}{\Delta/B \vdash \{\Delta(l)\}p_1}}{\Delta/B \Vdash \{\Delta(l)\}l : p_1}$$

By IH on $\mathcal{P}_0$ with $\mathcal{W}_0$ we get $\sigma' \vDash B$, and since the presence of the label does not affect the execution or the state, preserving soundness, this case is true.

$$\text{Case } \mathcal{P} = \text{P-IfGotoT}\frac{\overset{\mathcal{P}_0}{\langle b, \sigma\rangle \downarrow \textbf{true}} \quad \overset{\mathcal{P}_1}{p_0 \vdash l \Downarrow p_2} \quad \overset{\mathcal{P}_2}{p_0 \vdash \langle p_2, \sigma\rangle \downarrow \sigma'}}{p_0 \vdash \langle \textbf{if } b \textbf{ then goto } l; p_1, \sigma\rangle \downarrow \sigma'}$$

Here $p = \textbf{if } b \textbf{ then goto } l; p_1$, so $\mathcal{W}$ must have the following shape

$$\mathcal{W} = \text{W-IfGoto}\frac{\overset{\mathcal{W}_0}{\Delta/B \Vdash \{A_1\}p_1}}{\Delta/B \Vdash \{(b \Rightarrow \Delta(l)) \wedge (\neg b \Rightarrow A_1)\}\ \textbf{if } b \textbf{ then goto } l; p_1}$$

Since $b = \textbf{true}$ execution jumpts to p_2. By IH on $\mathcal{P}_2$ with $\mathcal{W}_0$ we get $\sigma' \vDash B$, and since we have $\sigma \vDash \{(b \Rightarrow \Delta(l)) \wedge (\neg b \Rightarrow A_1)\}$, the final state must satisfy the postcondition B.

$$\text{Case } \mathcal{P} = \text{P-IfGotoF}\frac{\overset{\mathcal{P}_0}{\langle b, \sigma\rangle \downarrow \textbf{true}} \quad \overset{\mathcal{P}_1}{p_0 \vdash \langle p_1, \sigma\rangle \downarrow \sigma'}}{p_0 \vdash \langle \textbf{if } b \textbf{ then goto } l; p_1, \sigma\rangle \downarrow \sigma'}$$

Here $p = \textbf{if } b \textbf{ then goto } l; p_1$, so $\mathcal{W}$ must have the following shape

$$\mathcal{W} = \text{W-IfGoto}\frac{\overset{\mathcal{W}_0}{\Delta/B \Vdash \{A_1\}p_1}}{\Delta/B \Vdash \{(b \Rightarrow \Delta(l)) \wedge (\neg b \Rightarrow A_1)\}\ \textbf{if } b \textbf{ then goto } l; p_1}$$

Since $b = \textbf{false}$ execution jumps to p_1. By IH on $\mathcal{P}_1$ with $\mathcal{W}_0$ we get $\sigma' \vDash B$, and since we have $\sigma \vDash \{(b \Rightarrow \Delta(l)) \wedge (\neg b \Rightarrow A_1)\}$, the final state must satisfy the postcondition B.

Task 3

Task 3.1

a)

Theorem 4.2 *If* $t \overset{\mathcal{E}}{\Downarrow} c$*, then* $t \to^* c$.

Proof. By induction on the big-step derivation.

$$\text{Case } \mathcal{E} = \text{E-NIL}\frac{}{[] \Downarrow []}$$

Here $t = []$ and $c = []$. This case is vacuously true, since $[]$ is already on canonical form.

$$\text{Case } \mathcal{E} = \text{E-CONS}\frac{\overset{\mathcal{E}_0}{t_1 \Downarrow c_1} \quad \overset{\mathcal{E}_1}{t_2 \Downarrow c_2}}{t_1 :: t_2 \Downarrow c_1 :: c_2}$$

Here $t = t_1 :: t_2$ and $c = c_1 :: c_2$. By IH on $\mathcal{E}_0$ we get $\mathcal{SS}_0$ of $t_1 \to^* c_1$, and we can then, by step-wise use of S-CONS1, get

$$t_1 :: t_2 \to^* c_1 :: t_2$$

By IH on $\mathcal{E}_1$ we get $\mathcal{SS}_1$ of $t_2 \to^* c_2$ and we can then, by step-wise use of S-CONS2, get

$$c_1 :: t_2 \to^* c_1 :: c_2$$

Concatenating these two derivations together with Lemma 4.1, we get the desired

$$t_1 :: t_2 \to^* c_1 :: c_2$$

$$\text{Case } \mathcal{E} = \text{E-FOLDN}\frac{\overset{\mathcal{E}_0}{t_0 \Downarrow []} \quad \overset{\mathcal{E}_n}{t_n \Downarrow c}}{\textbf{fold}(t_n, x.y.t_c)(t_0) \Downarrow c}$$

Here $t = \textbf{fold}(t_n, x.y.t_c)(t_0)$ and $c = c$. By IH on $\mathcal{E}_0$ we get $\mathcal{SS}_0$ of $t_0 \Downarrow []$, and we can then, by step-wise use of S-FOLD1, get

$$\textbf{fold}(t_n, x.y.t_c)(t_0) \to^* \textbf{fold}(t_n, x.y.t_c)([])$$

Now, by a simple use of S-FOLDN, we get

$$\textbf{fold}(t_n, x.y.t_c)([]) \to t_n$$

By IH on $\mathcal{E}_1$ we get $\mathcal{SS}_1$ of $t_n \Downarrow c$, and concatenating the first derivation with $\mathcal{SS}_1$ by Lemma 4.1, we get

$$\textbf{fold}(t_n, x.y.t_c)(t_0) \to^* c$$

$$\text{Case } \mathcal{E} = \text{E-FOLDC}\frac{\overset{\mathcal{E}_0}{t_0 \Downarrow c_1 :: c_2} \quad \overset{\mathcal{E}_1}{\textbf{fold}(t_n, x.y.t_c)(c_2) \Downarrow c'} \quad \overset{\mathcal{E}_2}{t_c[c_1/x][c'/y] \Downarrow c}}{\textbf{fold}(t_n, x.y.t_c)(t_0) \Downarrow c}$$

Here $t = \textbf{fold}(t_n, x.y.t_c)(t_0)$ and $c = c$. By IH on $\mathcal{E}_0$ we get $\mathcal{SS}_0$ of $t \to^* c_1 :: c_2$, and we can then, by step-wise use of S-CONS2 get

$$t_0 \to^* c_1 :: c_2$$

By IH on $\mathcal{E}_1$ we get $\mathcal{SS}_1$ of $\textbf{fold}(t_n, x.y.t_c)(c_2) \to^* c'$, and then by IH on $\mathcal{E}_2$ we get $\mathcal{SS}_2$ of $t_c[c_1/x][c'/y] \to c$, and we can, by a step-wise use S-FOLDC get

$$\textbf{fold}(t_n, x.y.t_c)(c_1 :: c_2) \to^* \textbf{let } y \Leftarrow \textbf{fold}(t_n, x.y.t_c)(c_2) \textbf{ in } t_c[c_1/x]$$

and we can then use E-LET to get

$$\textbf{let } y \Leftarrow \textbf{fold}(t_n, x.y.t_c)(c_2) \textbf{ in } t_c[c_1/x] \to^* c$$

Concatenating these three derivations together with Lemma 4.1, we get the required

$$\textbf{fold}(t_n, x.y.t_c)(t_0) \to^* c$$

b)

Lemma 4.4 *If* $t \overset{\mathcal{S}}{\to} t'$ *and* $t' \overset{\mathcal{E}'}{\downarrow} c$, *then* $t \overset{\mathcal{E}}{\downarrow} c$.

Proof. By induction on the first derivation.

$$\text{Case } \mathcal{S} = \text{S-C}\textsc{ons}1\,\frac{\overset{\mathcal{S}_0}{t_1 \to t_1'}}{t_1 :: t_2 \to t_1' :: t_2}$$

Here $t = (t_1 :: t_2)$, and then $\mathcal{E}'$ must have the following shape

$$\mathcal{E}' = \text{E-C}\textsc{ons}\,\frac{\overset{\mathcal{E}_0'}{t_1' \downarrow c_1} \quad \overset{\mathcal{E}_1'}{t_2 \downarrow c_2}}{t_1' :: t_2 \to c_1 :: c_2}$$

with $t' = (t_1' :: t_2)$. By IH on $\mathcal{S}_0$ with $\mathcal{E}_0'$ we get $\mathcal{E}_0$ of $t_1 \downarrow c_1$ and we can construct $\mathcal{E}$ as

$$\mathcal{E} = \text{E-C}\textsc{ons}\,\frac{\overset{\mathcal{E}_0}{t_1 \downarrow c_1} \quad \overset{\mathcal{E}_1'}{t_2 \downarrow c_2}}{t_1 :: t_2 \to c_1 :: c_2}$$

$$\text{Case } \mathcal{S} = \text{S-C}\textsc{ons}2\,\frac{\overset{\mathcal{S}_0}{t_2 \to t_2'}}{t_1 :: t_2 \to t_1 :: t_2'}$$

Here $t = (t_1 :: t_2)$, and then $\mathcal{E}'$ must have the following shape

$$\mathcal{E}' = \text{E-C}\textsc{ons}\,\frac{\overset{\mathcal{E}_0'}{t_1 \downarrow c_1} \quad \overset{\mathcal{E}_1'}{t_2' \downarrow c_2}}{t_1 :: t_2' \to c_1 :: c_2}$$

with $t' = (t_1 :: t_2')$. By IH on $\mathcal{S}_0$ with $\mathcal{E}_0'$ we get $\mathcal{E}_0$ of $t_2 \downarrow c_2$ and we can construct $\mathcal{E}$ as

$$\mathcal{E} = \text{E-C}\textsc{ons}\,\frac{\overset{\mathcal{E}_0'}{t_1 \downarrow c_1} \quad \overset{\mathcal{E}_1}{t_2 \downarrow c_2}}{t_1 :: t_2 \to c_1 :: c_2}$$

$$\text{Case } \mathcal{S} = \text{S-F}\textsc{old}1\,\frac{\overset{\mathcal{S}_0}{t_0 \to t_0'}}{\mathbf{fold}(t_n, x.y.t_c)(t_0) \to \mathbf{fold}(t_n, x.y.t_c)([])}$$

Here $t = (\mathbf{fold}(t_n, x.y.t_c)(t_0))$, which means we have two possibilities for $\mathcal{E}'$ depending on which of the two big-step rules are used:

$$\text{Subcase } \mathcal{E}' = \text{E-F}\textsc{old}\text{N}\,\frac{\overset{\mathcal{E}_0'}{t_0' \downarrow []} \quad \overset{\mathcal{E}_n'}{t_n \downarrow c}}{\mathbf{fold}(t_n, x.y.t_c)(t_0') \downarrow c}$$

with $t' = (\mathbf{fold}(t_n, x.y.t_c)(t_0'))$. By IH on $\mathcal{S}_0$ with $\mathcal{E}_0'$ we get $\mathcal{E}_0$ of $t_0 \downarrow []$, and we can construct $\mathcal{E}$ as

$$\mathcal{E} = \text{E-F}\textsc{old}\text{N}\,\frac{\overset{\mathcal{E}_0}{t_0 \downarrow []} \quad \overset{\mathcal{E}_n'}{t_n \downarrow c}}{\mathbf{fold}(t_n, x.y.t_c)(t_0) \downarrow c}$$

$$\text{Subcase } \mathcal{E}' = \text{E-F}\textsc{old}\text{C}\,\frac{\overset{\mathcal{E}_0'}{t_0' \downarrow c_1 :: c_2} \quad \overset{\mathcal{E}_1'}{\mathbf{fold}(t_n, x.y.t_c)(c_2) \downarrow c'} \quad \overset{\mathcal{E}_2'}{t_c[c_1/x][c'/y] \downarrow c}}{\mathbf{fold}(t_n, x.y.t_c)(t_0') \downarrow c}$$

with $t' = (\mathbf{fold}(t_n, x.y.t_c)(t_0'))$. By IH on $\mathcal{S}_0$ with $\mathcal{E}_0'$ we get $\mathcal{E}_0$ of $t_0 \downarrow c_1 :: c_2$, and we can construct $\mathcal{E}$ as

$$\mathcal{E} = \text{E-F}\textsc{old}\text{C}\,\frac{\overset{\mathcal{E}_0}{t_0 \downarrow c_1 :: c_2} \quad \overset{\mathcal{E}_1'}{\mathbf{fold}(t_n, x.y.t_c)(c_2) \downarrow c'} \quad \overset{\mathcal{E}_2'}{t_c[c_1/x][c'/y] \downarrow c}}{\mathbf{fold}(t_n, x.y.t_c)(t_0) \downarrow c}$$

Task 3.2

a)

Lemma 4.11 *If $t \to t'$ (by $\mathcal{S}$) and $[] \vdash t : \tau$ (by $\mathcal{T}$), then also $[] \vdash t' : \tau$ (by some $\mathcal{T}'$).*

Proof. By induction on the derivation $\mathcal{S}$.

$$\text{Case } \mathcal{S} = \text{S-Cons1}\frac{\overset{\mathcal{S}_0}{t_1 \to t_1'}}{t_1 :: t_2 \to t_1' :: t_2}$$

Here $t = (t_1 :: t_2)$, and then $\mathcal{T}$ must have the following shape

$$\mathcal{T} = \text{T-Cons}\frac{\overset{\mathcal{T}_0}{[] \vdash t_1 : \tau_0} \quad \overset{\mathcal{T}_1}{[] \vdash t_2 : \textbf{list } (\tau_0)}}{[] \vdash t_1 :: t_2 : \textbf{list } (\tau_0)}$$

By IH on $\mathcal{S}_0$ with $\mathcal{T}_0$ we get a derivation $\mathcal{T}_0'$ of $[] \vdash t_1' : \tau_0$, and we can construct $\mathcal{T}'$ as

$$\mathcal{T}' = \text{T-Cons}\frac{\overset{\mathcal{T}_0'}{[] \vdash t_1' : \tau_0} \quad \overset{\mathcal{T}_1}{[] \vdash t_2 : \textbf{list } (\tau_0)}}{[] \vdash t_1' :: t_2 : \textbf{list } (\tau_0)}$$

$$\text{Case } \mathcal{S} = \text{S-Cons2}\frac{\overset{\mathcal{S}_0}{t_1 \to t_1'}}{t_1 :: t_2 \to t_1' :: t_2}$$

Here $t = (t_1 :: t_2)$, and then $\mathcal{T}$ must have the following shape

$$\mathcal{T} = \text{T-Cons}\frac{\overset{\mathcal{T}_0}{[] \vdash t_1 : \tau_0} \quad \overset{\mathcal{T}_1}{[] \vdash t_2 : \textbf{list } (\tau_0)}}{[] \vdash t_1 :: t_2 : \textbf{list } (\tau_0)}$$

By IH on $\mathcal{S}_1$ with $\mathcal{T}_1$ we get a derivation $\mathcal{T}_1'$ of $[] \vdash t_2' : \textbf{list } (\tau_0)$, and we can construct $\mathcal{T}'$ as

$$\mathcal{T}' = \text{T-Cons}\frac{\overset{\mathcal{T}_0}{[] \vdash t_1 : \tau_0} \quad \overset{\mathcal{T}_1'}{[] \vdash t_2' : \textbf{list } (\tau_0)}}{[] \vdash t_1 :: t_2' : \textbf{list } (\tau_0)}$$

$$\text{Case } \mathcal{S} = \text{S-Fold1}\frac{\overset{\mathcal{S}_0}{t_0 \to t_0'}}{\textbf{fold}(t_n, x.y.t_c)(t_0) \to \textbf{fold}(t_n, x.y.t_c)([])}$$

Here $t = \textbf{fold}(t_n, x.y.t_c)(t_0)$. and then $\mathcal{T}$ must have the following shape

$$\mathcal{T} = \text{T-Fold}\frac{\overset{\mathcal{T}_0}{[] \vdash t_n : \tau} \quad \overset{\mathcal{T}_1}{[x \mapsto \tau_0][y \mapsto \tau] \vdash t_c : \tau} \quad \overset{\mathcal{T}_2}{[] \vdash t_0 : \textbf{list } \tau_0}}{[] \vdash \textbf{fold}(t_n, x.y.t_c)(t_0) : \tau}$$

By IH on $\mathcal{S}_0$ with $\mathcal{T}_2$ we get a derivation $\mathcal{T}_2'$ of $[] \vdash t_0' : \textbf{list } \tau_0$, and we can construct $\mathcal{T}'$ as

$$\mathcal{T} = \text{T-Fold}\frac{\overset{\mathcal{T}_0}{[] \vdash t_n : \tau} \quad \overset{\mathcal{T}_1}{[x \mapsto \tau_0][y \mapsto \tau] \vdash t_c : \tau} \quad \overset{\mathcal{T}_2'}{[] \vdash t_0' : \textbf{list } \tau_0}}{[] \vdash \textbf{fold}(t_n, x.y.t_c)(t_0') : \tau}$$

$$\text{Case } \mathcal{S} = \text{S-FoldN}\frac{}{\textbf{fold}(t_n, x.y.t_c)([]) \to t_n}$$

Here $t = \textbf{fold}(t_n, x.y.t_c)([])$ and $t' = t_n$, and the typing derivation for t must look like

$$\mathcal{T} = \text{T-Fold}\frac{\overset{\mathcal{T}_0}{[] \vdash t_n : \tau} \quad \overset{\mathcal{T}_1}{[x \mapsto \tau_0][y \mapsto \tau] \vdash t_c : \tau} \quad \text{T-Nil}\frac{}{[] \vdash [] : \textbf{list } (\tau_0)}}{[] \vdash \textbf{fold}(t_n, x.y.t_c)([]) : \tau}$$

But then we can directly take $\mathcal{T}' = \mathcal{T}_0$ as the typing derivation for t'.

$$\text{Case } \mathcal{S} = \text{S-FoldC}\frac{}{\textbf{fold}(t_n, x.y.t_c)(c_1 :: c_2) \to \textbf{let } y \Leftarrow \textbf{fold}(t_n, x.y.t_c)(c_2) \textbf{ in } t_c[c_1/x]}$$

Here $t = \textbf{fold}(t_n, x.y.t_c)(c_1 :: c_2)$ and $t' = \textbf{let } y \Leftarrow \textbf{fold}(t_n, x.y.t_c)(c_2) \textbf{ in } t_c[c_1/x]$, and the typing derivation for t must look like

$$\mathcal{T} = \text{T-Fold}\frac{\overset{\mathcal{T}_0}{[] \vdash t_n : \tau} \quad \overset{\mathcal{T}_1}{[x \mapsto \tau_0][y \mapsto \tau] \vdash t_c : \tau} \quad \text{T-Cons}\frac{\overset{\mathcal{T}_2}{[] \vdash c_1 : \tau_0} \quad \overset{\mathcal{T}_3}{[] \vdash c_2 : \textbf{list } (\tau_0)}}{[] \vdash (c_1 :: c_2) : \textbf{list } (\tau_0)}}{[] \vdash \textbf{fold}(t_n, x.y.t_c)(c_1 :: c_2) : \tau}$$

And so we get $\mathcal{T}_4'$ of $[y \mapsto \tau_1] \vdash t_c[c_1/x] : \tau$ by Lemma 4.10 on $\mathcal{T}_1$ and $\mathcal{T}_2$, and so we can construct $\mathcal{T}'$ as

$$\mathcal{T}' = \text{T-Let}\frac{\text{T-Fold}\frac{\overset{\mathcal{T}_0}{[] \vdash t_n : \tau_1} \quad \overset{\mathcal{T}_1}{[x \mapsto \tau_0][y \mapsto \tau_1] \vdash t_c : \tau_1} \quad \overset{\mathcal{T}_3}{[] \vdash c_2 : \textbf{list } (\tau_0)}}{\textbf{fold}(t_n, x.y.t_c)(c_2) : \tau_1} \quad \overset{\mathcal{T}_4'}{[y \mapsto \tau_1] \vdash t_c[c_1/x] : \tau_2}}{\textbf{let } y \Leftarrow \textbf{fold}(t_n, x.y.t_c)(c_2) \textbf{ in } t_c[c_1/x] : \tau_2}$$

where $\tau = \tau_2 = \tau_1$.

b)

Theorem 4.13 (Termination) *If $\Gamma \vdash t : \tau$ without using rule* T-Rec*, then $\Gamma \vDash t : \tau$.*

Proof. Let $\mathcal{T}$ by a derivation of $\Gamma \vdash t : \tau$, and let s be such that, whenever $\Gamma(x) = \tau'$, then $\vDash^c x[s] : \tau'$. We must show that $\vDash^t t[s] : \tau$. i.e., that $t[s] \downarrow c$ for some c with $\vDash^c c : \tau$. We proceed by induction on $\mathcal{T}$.

$$\text{Case } \mathcal{T} = \text{T-Nil}\frac{}{\Gamma \vdash [] : \textbf{list } (\tau_0)}$$

so $\tau = \textbf{list } (\tau_0)$. We then have $[][s] \downarrow []$, and $\vDash^c [] : \textbf{list } (\tau_0)$, as required.

$$\text{Case } \mathcal{T} = \text{T-Cons}\frac{\overset{\mathcal{T}_0}{\Gamma \vdash t_1 : \tau_0} \quad \overset{\mathcal{T}_1}{\Gamma \vdash t_2 : \textbf{list } (\tau_0)}}{\Gamma \vdash t_1 :: t_2 : \textbf{list } (\tau_0)}$$

By IH on $\mathcal{T}_0$ we get a derivation $\mathcal{E}_0$ of $t_1[s] \downarrow c_0$, where $\vDash^c c_0 : \tau_0$. By IH on $\mathcal{T}_1$, we get a derivation $\mathcal{E}_1$ of $t_1[s] \downarrow \textbf{list } (c_0)$. But then by E-Cons on $\mathcal{E}_0$ and $\mathcal{E}_1$, we get $t_1[s] :: t_2[s] \downarrow c_1 :: c_2$, and $\vDash^c c_1 :: c_2 : \textbf{list } (\tau_0)$ by definition.

$$\mathcal{T} = \text{T-Fold}\frac{\overset{\mathcal{T}_0}{\Gamma \vdash t_n : \tau} \quad \overset{\mathcal{T}_1}{\Gamma[x \mapsto \tau_0][y \mapsto \tau] \vdash t_c : \tau} \quad \overset{\mathcal{T}_2}{\Gamma \vdash t_0 : \textbf{list } \tau_0}}{\Gamma \vdash \textbf{fold}(t_n, x.y.t_c)(t_0) : \tau}$$

By IH on $\mathcal{T}_0$, we get a derivation $\mathcal{E}_0$ of $t_n[s] \downarrow c_n$, where $\vDash^c c_n : \tau$. By IH on $\mathcal{T}_2$, we get a derivation $\mathcal{E}_2$ of $t_0[s] \downarrow c_0$, where $\vDash^c c_0 : \textbf{list } (\tau_0)$.

We now need to prove, that for any t_n and any list of canoical forms c_0, the term $\textbf{fold}(t_n, x.y.t_c)(c_0)$ evaluates to a canonical form of type τ. We split it up into two cases.

For an empty list $c_0 = []$, we get $\textbf{fold}(t_n, x.y.t_c)([])[s] \downarrow c_n$, with $\vDash^c c_n : \tau$.

For a non-empty list $c_0 = (c_1 :: c_2)$ the term $\textbf{fold}(t_n, x.y.t_c)(c_1 :: c_2)$ can step by S-FoldC to
$\textbf{let } y \Leftarrow \textbf{fold}(t_n, x.y.t_c)(c_2) \textbf{ in } t_c[c_1/x]$, and by induction on the typing derivation of $\textbf{fold}(t_n, x.y.t_c)(c_2)$, we get $\mathcal{E}_1$ of $\textbf{fold}(t_n, x.y.t_c)(c_2)[s] \downarrow c_2'$ for some c_2' on canonical form, with $\vDash^c c_2' : \tau$. By Lemma 4.10 on $\Gamma[c_2'/y] \vdash t_c[c_1/x]$ with the typing derivation for c_2', we get $\mathcal{E}_3$ of $t_c[c_1/x, c_2'/y]$, and thus by E-FoldC on $\mathcal{E}_0$, $\mathcal{E}_1$ and $\mathcal{E}_3$ we obtain $\textbf{fold}(t_n[s], x.y.t_c[s])(c_1 :: c_2)[s] \downarrow c$ with $\vDash^c c : \tau$ as required.

Task 3.3

a)

We write out the constraint-typing rules for new list construct.

$$\text{CT-Nil} : \frac{}{\hat{\Gamma} \vdash^i [] : \textbf{list}(\hat{\tau}) | i'}$$

$$\text{CT-Cons}: \frac{\hat{\Gamma} \vdash^{i} t_1 : \tau_1 \mid^{i''} C_0 \quad \hat{\Gamma} \vdash^{i''} t_2 : \tau_2 \mid^{i'} C_1}{\hat{\Gamma} \vdash^{i} t_1 :: t_2 : \textbf{list } (\boxed{i}) \mid^{i'} C_0, C_1, \boxed{i} \doteq \tau_1, \textbf{list } (\boxed{i}) \doteq \tau_2}$$

$$\text{CT-Fold}: \frac{\hat{\Gamma} \vdash^{i} t_n : \hat{\tau_n} \mid^{i''} C_0 \quad \hat{\Gamma}[x \mapsto \boxed{i''}, y \mapsto \hat{\tau}] \vdash^{i''+1} t_c : \hat{\tau_c} \mid^{i'''} C_1 \quad \hat{\Gamma} \vdash^{i'''} t_0 : \tau_0 \mid^{i'} C_2}{\hat{\Gamma} \vdash^{i} \textbf{fold}(t_n, x.y.t_c)(t_0) : \tau_n \mid^{i'} C_0, C_1, C_2, \tau_0 \doteq \textbf{list}(\boxed{i''}), \hat{\tau_n} \doteq \hat{\tau_c}}$$

b)

($\Rightarrow$)
Claim: If $\Gamma \vdash \textbf{let } n \Leftarrow [] \textbf{ in } t : \tau \Rightarrow \Gamma \vdash t[[]/n] : \tau$.

The let binding introduces a local scope where n is bound to []. If t, in the context of this binding, has type τ, substituting n with [] directly should preserve the type, since the semantic meaning of n as [] is the same in both expressions.

($\Leftarrow$)
Claim: If $\Gamma \vdash t[[]/n] : \tau \Rightarrow \Gamma \vdash \textbf{let } n \Leftarrow [] \textbf{ in } t : \tau$.

Here $t[[]/n]$ being well-typed indicates that substituting [] for n in t results in a term of type τ. Introducing a **let**-binding that assigns [] to n before evaluating t essentially provides the same setup for n within the scope of t. Therefore the typing should be preserved.

This proof relies on the fact that if n is bound to some other type in Γ neither expression would be well-typed.