Semantics and Types - Assignment 5

Mikkel Willén
`bmq419`

March 20, 2024

Task 5.1

Lemma 5.3 *If $[] \vdash t : \tau$ then either $t = c$ for some canonical form c, or $t \to t'$ for some term t'.*

$$\text{Case } \mathcal{T} = \textsc{T-Vnt}'\frac{\overset{\mathcal{T}_0}{[] \vdash t_0 : \tau_0}}{[] \vdash \langle l = t_0 \rangle : \langle l : \tau_0 \rangle}$$

As in the case for select, we consider only the $\textsc{T-Vnt}'$ rule. By IH on $\mathcal{T}_0$, the term t_0 either $t_0 \to t_0'$ and then $t = \langle l = t_0 \rangle \to \langle l = t_0' \rangle$ by $\textsc{S-Vnt1}$ or it is on canonical form, which means we have $t = \langle l = c_0 \rangle$, which is also on canonical form.

$$\text{Case } \mathcal{T} = \textsc{T-Case}\frac{\overset{\mathcal{T}_0}{[] \vdash t_0 : \langle (l_i : \tau_i)^{i \in 1 \dots n} \rangle} \quad \overset{\mathcal{T}_i}{[x_i \mapsto \tau_i] \vdash t_i : \tau}}{[] \vdash \textbf{case } t_0 \textbf{ of } (\langle l_i = x_i \rangle \Rightarrow t_i)^{i \in 1 \dots n} : \tau}$$

By IH on $\mathcal{T}_0$ either $t_0 \to t_0'$, and then $t = \textbf{case } t_0 \textbf{ of } (\langle l_i = x_i \rangle \Rightarrow t_i)^{i \in 1 \dots n} \to \textbf{case } t_0' \textbf{ of } (\langle l_i = x_i \rangle \Rightarrow t_i)^{i \in 1 \dots n}$ by $\textsc{S-Case1}$ or t_0 is on canonical form. In the latter case, by Lemma 5.2(d) on $\mathcal{T}_0$, we must have $t_0 = \langle l = c_0 \rangle$, with $l = l_k$ for some $k \in [1 \dots n]$, and thus $\textbf{case } t_0 \textbf{ of } (\langle l_i = x_i \rangle \Rightarrow t_i)^{i \in 1 \dots n}$ reduces to $t_k[c_0/x_k]$ by $\textsc{T-Case}$.

Lemma 5.5 *If $[] \vdash t : \tau$ and $t \to t'$, then $[] \vdash t' : \tau$.*

$$\text{Case } \mathcal{S} = \textsc{S-Sel}\frac{}{\{(l_i = c_i)^{i \in 1 \dots n}\}.l_k \to c_k}$$

where $k \in 1 \dots n$. Since the typing derivation cannot end in $\textsc{T-Sub}$, it must end with a use of $\textsc{T-Sel}$'.

$$\mathcal{T} = \textsc{T-Sel}'\frac{\overset{\mathcal{T}_0}{[] \vdash \{(l_i = c_i)^{i \in 1 \dots n}\} : \{l_k : \tau\}}}{[] \vdash \{(l_i = c_i)^{i \in 1 \dots n}\}.l_k : \tau}$$

Using Lemma 5.2(c) on $\mathcal{T}_0$ (where m is taken as 1, and $c = \{(l_i = c_i)^{i \in 1 \dots n}\}$, we get that there exists an $i \in 1 \dots n$ such that $l_i = l_k$ and that $[] \vdash c_i : \tau$ (by some $\mathcal{T}_{00}$). So we must have that $i = k$ and thus we also have that $[] \vdash c_k : \tau$.

Task 5.2

We consider the system with just $\textsc{T-Var}$, $\textsc{T-Lam}$, $\textsc{T-App}$, $\textsc{T-Sub}$, $\textsc{ST-Refl}$, $\textsc{ST-Trans}$ and $\textsc{ST-Fun}$. We show, that the following type is admissible:

$$\textsc{T-Eta} : \frac{\Gamma \vdash \lambda x.(tx) : \tau}{\Gamma \vdash t : \tau} \; (x \notin FV(t))$$

We have that if $\Gamma[x \mapsto \tau'] \vdash t : \tau$ and $x \notin FV(t)$, then also $\Gamma \vdash t : \tau$ (strengthening). We start by showing the following Lemma.

Lemma S. *If $\mathcal{T}$ is a derivation of $\Gamma \vdash t : \tau$, then for some τ', there exists a derivation $\mathcal{T}'$ of $\Gamma \vdash t : \tau'$, where $\mathcal{T}'$ does not end in a use of $\textsc{T-Sub}$, and a derivation $\mathcal{ST}$ of $\tau' \leq \tau$.*

We do this by induction on derivation of $\mathcal{T}$:

$$\text{Case } \mathcal{T} = \textsc{T-Var}\frac{}{\Gamma \vdash x : \tau} \; (\Gamma(x) = \tau)$$

Since the typing derivation $\mathcal{T}'$ cannot end in T-Sub, it must end with T-Var.

$$\mathcal{T}' = \text{T-Var}\frac{}{\Gamma \vdash x : \tau'} \ (\Gamma(x) = \tau')$$

and so we have $\mathcal{T}' = \mathcal{T}$. We have that x maps to some τ_0 in Γ, and the lookup of this in the same Γ must have the same type, so we have that $\tau' = \tau$, and so we get $\tau \leq \tau$ by ST-Refl.

$$\text{Case } \mathcal{T} = \text{T-Lam}\frac{\overset{\mathcal{T}_0}{\Gamma[x \mapsto \tau_1] \vdash t_0 : \tau_2}}{\Gamma \vdash \lambda x.t_0 : \tau_1 \to \tau_2}$$

Since the typing derivation $\mathcal{T}'$ cannot end in T-Sub, it must end with T-Lam.

$$\mathcal{T}' = \text{T-Lam}\frac{\overset{\mathcal{T}_0'}{\Gamma[x \mapsto \tau_1] \vdash t_0 : \tau_2'}}{\Gamma \vdash \lambda x.t_0 : \tau_1 \to \tau_2'}$$

and so we again have $\mathcal{T}' = \mathcal{T}$. So we again have that $\tau' = \tau$, so by ST-Refl we have that $\tau \leq \tau$ or $\tau_1 \to \tau_2 \leq \tau_1 \to \tau_2$.

$$\text{Case } \mathcal{T} = \text{T-App}\frac{\overset{\mathcal{T}_0}{\Gamma \vdash t_1 : \tau_1 \to \tau_2} \quad \overset{\mathcal{T}_1}{\Gamma \vdash t_2 : \tau_1}}{\Gamma \vdash t_1 t_2 : \tau_2}$$

Since the typing derivation $\mathcal{T}'$ cannot end in T-Sub, must must end with T-App.

$$\mathcal{T}' = \text{T-App}\frac{\overset{\mathcal{T}_0'}{\Gamma \vdash t_1 : \tau_1 \to \tau_2'} \quad \overset{\mathcal{T}_1'}{\Gamma \vdash t_2 : \tau_1}}{\Gamma \vdash t_1 t_2 : \tau_2'}$$

and so we again have $\mathcal{T}' = \mathcal{T}$. So we again have that $\tau' = \tau$, so by ST-Refl we have that $\tau \leq \tau$ or $\tau_2 \leq \tau_2$.

$$\text{Case } \mathcal{T} = \text{T-Sub}\frac{\overset{\mathcal{T}_0}{\Gamma \vdash t : \tau''} \quad \overset{\mathcal{ST}_0}{\tau'' \leq \tau}}{\Gamma \vdash t : \tau}$$

Now by IH on $\mathcal{T}_0$ we have that for some τ' there exists a derivation $\mathcal{T}_0'$ of $\Gamma \vdash t : \tau'$, where $\mathcal{T}_0'$ does not end in T-Sub, and a derivation $\mathcal{ST}$ of $\tau' \leq \tau''$ and thus by ST-Trans on $\mathcal{ST}$ and $\mathcal{ST}_0$, we have that $\tau' \leq \tau$.

This completes the proof of Lemma S.

Now by Lemma S on $\Gamma \vdash \lambda x.(tx) : \tau$, we have that for some τ' there exists a derivation $\mathcal{T}'$ of $\Gamma \vdash \lambda x.(tx) : \tau'$, where $\mathcal{T}'$ does not end in a use of T-Sub, and a derivation $\mathcal{ST}$ of $\tau' \leq \tau$, and thus we create the derivation:

$$\mathcal{T}' = \text{T-Lam}\frac{\text{T-App}\dfrac{\overset{\mathcal{T}_0}{\Gamma[x \mapsto \tau_1] \vdash t : \tau_1 \to \tau_2} \quad \Gamma[x \mapsto \tau_1] \vdash t_2 : \tau_1}{\Gamma[x \mapsto \tau_1] \vdash tx : \tau_2}}{\Gamma \vdash \lambda x.(tx) : \tau} \ (x \notin FV(t))$$

where $\mathcal{T}'$ does not use T-Sub. Then by (*strengthening*) on $\mathcal{T}_0$ (where $\tau' = \tau_1$, $\tau = \tau_1 \to \tau_2$ and $x \notin FV(t)$) we have that $\Gamma \vdash t : \tau$, showing that if the premise is derivable, then so is the conclusion T-Eta.

Task 3

a)

We write out the typing rules for the **binary-sum** type.

$$\text{T-BSum} : \frac{\Gamma \vdash t_0 : \tau_1 + \tau_2 \quad \Gamma[x_1 \mapsto \tau_1] \vdash t_1 : \tau \quad \Gamma[x_2 \mapsto \tau_2] \vdash t_2 : \tau}{\Gamma \vdash \textbf{case } t_0 \textbf{ of inl}(x_1) \Rightarrow t_1,\ \textbf{inr}(x_2) \Rightarrow t_2 : \tau}$$

$$\text{T-InL} : \frac{\Gamma \vdash t_1 : \tau_1}{\Gamma \vdash \textbf{inl}(t_1) : \tau_1 + \tau_2} \qquad \text{T-InR} : \frac{\Gamma \vdash t_2 : \tau_2}{\Gamma \vdash \textbf{inr}(t_2) : \tau_1 + \tau_2}$$

b)

We write out the constraint-generating typing rules for the **binary-sum** type:

$$\text{CT-BSum}: \frac{\hat{\Gamma} \vdash^{i} t_0 : \hat{\tau}_1 + \hat{\tau}_2 \mid^{i''} C_0 \quad \hat{\Gamma}[x_1 \mapsto \boxed{i''}] \vdash^{i''} t_1 : \hat{\tau} \mid^{i'''} C_1 \quad \hat{\Gamma}[x_2 \mapsto \boxed{i'''}] \vdash^{i'''} t_2 : \hat{\tau} \mid^{i'} C_2}{\hat{\Gamma} \vdash^{i} \textbf{case } t_0 \textbf{ of inl}(x_1) \Rightarrow t_1,\ \textbf{inr}(x_2) \Rightarrow t_2 : \hat{\tau} \mid^{i'} C_0, C_1, C_2, \hat{\tau}_1 \doteq \boxed{i''}, \hat{\tau}_2 \doteq \boxed{i'''}}$$

$$\text{CT-InL}: \frac{\hat{\Gamma} \vdash^{i} t_1 : \hat{\tau}_1 \mid^{i'} C_1}{\hat{\Gamma} \vdash^{i} \textbf{inl}(t_1) : \boxed{i'} + \boxed{i'+1} \mid^{i'+2} C_1, \hat{\tau}_1 \doteq \boxed{i'}}$$

$$\text{CT-InR}: \frac{\hat{\Gamma} \vdash^{i} t_2 : \hat{\tau}_2 \mid^{i'} C_2}{\hat{\Gamma} \vdash^{i} \textbf{inr}(t_2) : \boxed{i'} + \boxed{i'+1} \mid^{i'+2} C_2, \hat{\tau}_2 \doteq \boxed{i'+1}}$$

c)

We employ the constrain-generation algorithm on:

$$\lambda x.\textbf{case } x \textbf{ of inl}(y) \Rightarrow x, \textbf{inr}(z) \Rightarrow \textbf{inl}(z)$$

By constructing a full derivation tree:

$$\text{CT-Lam}\frac{\overbrace{[x \mapsto \boxed{0}] \vdash^{1} \textbf{case } x \textbf{ of inl}(y) \Rightarrow x, \textbf{inr}(z) \Rightarrow \textbf{inl}(z) : \hat{\tau}_0 \mid^{2} C_0)}^{\text{CT-BSum}}}{[] \vdash^{0} \lambda x.\textbf{case } x \textbf{ of inl}(y) \Rightarrow x, \textbf{inr}(z) \Rightarrow \textbf{inl}(z) : \boxed{0} \to \hat{\tau}_0 \mid^{1} C_0}$$

$$\text{CT-BSum}\frac{\overbrace{[x \mapsto \boxed{0}] \vdash^{3} x : \hat{\tau}_1 + \hat{\tau}_2 \mid^{3} C_0}^{\text{CT-Var}_1} \quad \overbrace{[x \mapsto \boxed{0}, y \mapsto \boxed{3}] \vdash^{3} x : \hat{\tau} \mid^{4} C_1}^{\text{CT-Var}_2} \quad \overbrace{[x \mapsto \boxed{0}, z \mapsto \boxed{4}] \vdash^{4} \textbf{inl}(z) : \hat{\tau} \mid^{2} C_2}^{\text{CT-InL}}}{[x \mapsto \boxed{0}] \vdash^{1} \textbf{case } x \textbf{ of inl}(y) \Rightarrow x, \textbf{inr}(z) \Rightarrow \textbf{inl}(z) : \hat{\tau}_0 \mid^{7} C_0}$$

$$\text{CT-Var}_1\frac{}{[x \mapsto \boxed{0}] \vdash^{3} x : \hat{\tau}_1 + \hat{\tau}_2 \mid^{3} C_0}$$

$$\text{CT-Var}_2\frac{}{[x \mapsto \boxed{0}, y \mapsto \boxed{3}] \vdash^{3} x : \hat{\tau} \mid^{4} C_1}$$

$$\text{CT-InL}\frac{\text{T-Var}\dfrac{}{[x \mapsto \boxed{0}, z \mapsto \boxed{4}] \vdash^{4} z : \hat{\tau}_1 \mid^{5}}}{[x \mapsto \boxed{0}, z \mapsto \boxed{4}] \vdash^{4} \textbf{inl}(z) : \hat{\tau} \mid^{7} C_2, \hat{\tau}_1 \doteq \boxed{5}]}$$

We find that the expression has candidate types $\boxed{0} \to \boxed{4}$ and $\boxed{0} \to \boxed{5}$, subject to the following constraints: $\boxed{0} \doteq \boxed{4}, \boxed{0} \doteq \boxed{3} + \boxed{4}, \boxed{4} \doteq \boxed{5}$

d)

$\boxed{0} \doteq \boxed{4}, \boxed{0} \doteq \boxed{3} + \boxed{4}, \boxed{4} \doteq \boxed{5}$
$\rightsquigarrow$ (by CS-Triv)
$(\boxed{0} \doteq \boxed{4}, \boxed{0} \doteq \boxed{3} + \boxed{4}, \boxed{4} \doteq \boxed{5})[\boxed{4}/\boxed{0}], \boxed{0} \doteq^{\checkmark} \boxed{4}$
$=$
$\boxed{0} \doteq \boxed{3} + \boxed{0}, \boxed{0} \doteq \boxed{5}$ $\rightsquigarrow$ (by CS-Triv)
Now we get to a problem, since $\boxed{0}$ is about to be substitued with itself and something more, and so we got infinte recursion.