More types
Section and Practice Problems

Mar 6-9, 2018

---

1 Products and Sums

For these questions, use the lambda calculus with products and sums (Lecture 13§1.1).

(a) Write a program that constructs two values of type $\textbf{int} + (\textbf{int} \rightarrow \textbf{int})$, one using left injection, and one using right injection.

Answer:

$$\textit{let } a : \textbf{int} + (\textbf{int} \rightarrow \textbf{int}) = \textit{inl}_{\textbf{int}+(\textbf{int}\rightarrow\textbf{int})}\ 3 \textit{ in}$$
$$\textit{inr}_{\textbf{int}+(\textbf{int}\rightarrow\textbf{int})}\ \lambda x : \textbf{int}.\, 3$$

(b) Write a function that takes a value of type $\textbf{int} + (\textbf{int} \rightarrow \textbf{int})$ and if the value is an integer, it adds 7 to it, and if the value is a function it applies the function to 42.

Answer:

$$\lambda a : \textbf{int} + (\textbf{int} \rightarrow \textbf{int}).\ \textit{case } a \textit{ of } \lambda y : \textbf{int}.\, y + 7 \mid \lambda f : \textbf{int} \rightarrow \textbf{int}.\, f\ 42$$

(c) Give a typing derivation for the following program.

$$\lambda p : (\textbf{unit} \rightarrow \textbf{int}) \times (\textbf{int} \rightarrow \textbf{int}).\ \lambda x : \textbf{unit} + \textbf{int}.\ \text{case } x \text{ of } \#1\ p \mid \#2\ p$$

Answer: *For brevity, let* $e_1 \equiv \lambda x : \textbf{unit} + \textbf{int}.\ \textit{case } x \textit{ of } \#1\ p \mid \#2\ p$ *and let* $\Gamma = \{p : (\textbf{unit} \rightarrow \textbf{int}) \times (\textbf{int} \rightarrow \textbf{int}), x : \textbf{unit} + \textbf{int}\}$

$$\text{T-ABS}\ \dfrac{\text{T-ABS}\ \dfrac{\text{T-CASE}\ \dfrac{\text{T-VAR}\ \dfrac{}{\Gamma \vdash x : \textbf{unit} + \textbf{int}} \quad \text{T-LPROJ}\ \dfrac{\text{T-VAR}\ \dfrac{}{\Gamma \vdash p : (\textbf{unit} \rightarrow \textbf{int} \times \textbf{int} \rightarrow \textbf{int})}}{\Gamma \vdash \#1\ p : \textbf{unit} \rightarrow \textbf{int}} \quad \text{T-RPROJ}\ \dfrac{\text{T-VAR}\ \dfrac{}{\Gamma \vdash p : (\textbf{unit} \rightarrow \textbf{int} \times \textbf{int} \rightarrow \textbf{int})}}{\Gamma \vdash \#2\ p : \textbf{int} \rightarrow \textbf{int}}}{\Gamma \vdash \textit{case } x \textit{ of } \#1\ p \mid \#2\ p : \textbf{int}}}{p : (\textbf{unit} \rightarrow \textbf{int}) \times (\textbf{int} \rightarrow \textbf{int}) \vdash \lambda x : \textbf{unit} + \textbf{int}.\ \textit{case } x \textit{ of } \#1\ p \mid \#2\ p : (\textbf{unit} + \textbf{int}) \rightarrow \textbf{int}}}{\vdash \lambda p : (\textbf{unit} \rightarrow \textbf{int}) \times (\textbf{int} \rightarrow \textbf{int}).\ e_1 : ((\textbf{unit} \rightarrow \textbf{int}) \times (\textbf{int} \rightarrow \textbf{int})) \rightarrow (\textbf{unit} + \textbf{int}) \rightarrow \textbf{int}}$$

(d) Write a program that uses the term in part (c) above to produce the value 42.

Answer: *We refer to the term in part (c) above as* f.

$$f\ (\lambda x : \textbf{unit} \rightarrow \textbf{int}.\, 42, \lambda x : \textbf{int}.\, 41)\ \textit{inl}_{\textbf{unit}+\textbf{int}}\ ()$$

2 Recursion

(a) Use the $\mu x.\ e$ expression to write a function that takes a natural number n and returns the sum of all even natural numbers less than or equal to n. (You can assume you have appropriate integer comparison operators, and also a modulus operator.)

Answer:

$$\mu f.\ \lambda n.\ \textbf{if}\ n \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (n\ \textit{mod}\ 2) = 0\ \textbf{then}\ n + f\ (n-2)\ \textbf{else}\ f\ (n-1)$$

(b) Try executing your program by applying it to the number 5.

Answer: *The program executes correctly and returns 6. For brevity, we will refer to the expression from the answer above as F.*

$$
\begin{aligned}
& F\ 5 \\
\longrightarrow & (\lambda n.\ \textbf{if}\ n \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (n\ \textit{mod}\ 2) = 0\ \textbf{then}\ n + F\ (n-2)\ \textbf{else}\ F\ (n-1))\ 5 \\
\longrightarrow & \textbf{if}\ 5 \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (5\ \textit{mod}\ 2) = 0\ \textbf{then}\ 5 + F\ (5-2)\ \textbf{else}\ F\ (5-1) \\
\longrightarrow & \textbf{if}\ \textit{false}\ \textbf{then}\ 0\ \textbf{else if}\ (5\ \textit{mod}\ 2) = 0\ \textbf{then}\ 5 + F\ (5-2)\ \textbf{else}\ F\ (5-1) \\
\longrightarrow & \textbf{if}\ (5\ \textit{mod}\ 2) = 0\ \textbf{then}\ 5 + F\ (5-2)\ \textbf{else}\ F\ (5-1) \\
\longrightarrow & \textbf{if}\ 1 = 0\ \textbf{then}\ 5 + F\ (5-2)\ \textbf{else}\ F\ (5-1) \\
\longrightarrow & \textbf{if}\ \textit{false}\ \textbf{then}\ 5 + F\ (5-2)\ \textbf{else}\ F\ (5-1) \\
\longrightarrow & F\ (5-1) \\
\longrightarrow & (\lambda n.\ \textbf{if}\ n \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (n\ \textit{mod}\ 2) = 0\ \textbf{then}\ n + F\ (n-2)\ \textbf{else}\ F\ (n-1))\ (5-1) \\
\longrightarrow & (\lambda n.\ \textbf{if}\ n \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (n\ \textit{mod}\ 2) = 0\ \textbf{then}\ n + F\ (n-2)\ \textbf{else}\ F\ (n-1))\ 4 \\
\longrightarrow & \textbf{if}\ 4 \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (4\ \textit{mod}\ 2) = 0\ \textbf{then}\ 4 + F\ (4-2)\ \textbf{else}\ F\ (4-1) \\
\longrightarrow & \textbf{if}\ \textit{false}\ \textbf{then}\ 0\ \textbf{else if}\ (4\ \textit{mod}\ 2) = 0\ \textbf{then}\ 4 + F\ (4-2)\ \textbf{else}\ F\ (4-1) \\
\longrightarrow & \textbf{if}\ (4\ \textit{mod}\ 2) = 0\ \textbf{then}\ 4 + F\ (4-2)\ \textbf{else}\ F\ (4-1) \\
\longrightarrow & \textbf{if}\ 0 = 0\ \textbf{then}\ 4 + F\ (4-2)\ \textbf{else}\ F\ (4-1) \\
\longrightarrow & \textbf{if}\ \textit{true}\ \textbf{then}\ 4 + F\ (4-2)\ \textbf{else}\ F\ (4-1) \\
\longrightarrow & 4 + F\ (4-2) \\
\longrightarrow & 4 + (\lambda n.\ \textbf{if}\ n \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (n\ \textit{mod}\ 2) = 0\ \textbf{then}\ n + F\ (n-2)\ \textbf{else}\ F\ (n-1))\ (4-2) \\
\longrightarrow & 4 + (\lambda n.\ \textbf{if}\ n \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (n\ \textit{mod}\ 2) = 0\ \textbf{then}\ n + F\ (n-2)\ \textbf{else}\ F\ (n-1))\ 2 \\
\longrightarrow & 4 + (\textbf{if}\ 2 \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (2\ \textit{mod}\ 2) = 0\ \textbf{then}\ 2 + F\ (2-2)\ \textbf{else}\ F\ (2-1)) \\
\longrightarrow & 4 + (\textbf{if}\ \textit{false}\ \textbf{then}\ 0\ \textbf{else if}\ (2\ \textit{mod}\ 2) = 0\ \textbf{then}\ 2 + F\ (2-2)\ \textbf{else}\ F\ (2-1)) \\
\longrightarrow & 4 + (\textbf{if}\ (2\ \textit{mod}\ 2) = 0\ \textbf{then}\ 2 + F\ (2-2)\ \textbf{else}\ F\ (2-1)) \\
\longrightarrow & 4 + (\textbf{if}\ 0 = 0\ \textbf{then}\ 2 + F\ (2-2)\ \textbf{else}\ F\ (2-1)) \\
\longrightarrow & 4 + (\textbf{if}\ \textit{true}\ \textbf{then}\ 2 + F\ (2-2)\ \textbf{else}\ F\ (2-1)) \\
\longrightarrow & 4 + (2 + F\ (2-2)) \\
\longrightarrow & 4 + (2 + (\lambda n.\ \textbf{if}\ n \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (n\ \textit{mod}\ 2) = 0\ \textbf{then}\ n + F\ (n-2)\ \textbf{else}\ F\ (n-1))\ (2-2)) \\
\longrightarrow & 4 + (2 + (\lambda n.\ \textbf{if}\ n \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (n\ \textit{mod}\ 2) = 0\ \textbf{then}\ n + F\ (n-2)\ \textbf{else}\ F\ (n-1))\ 0) \\
\longrightarrow & 4 + (2 + (\lambda n.\ \textbf{if}\ n \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (n\ \textit{mod}\ 2) = 0\ \textbf{then}\ n + F\ (n-2)\ \textbf{else}\ F\ (n-1))\ 0) \\
\longrightarrow & 4 + (2 + (\textbf{if}\ 0 \leq 0\ \textbf{then}\ 0\ \textbf{else if}\ (0\ \textit{mod}\ 2) = 0\ \textbf{then}\ 0 + F\ (0-2)\ \textbf{else}\ F\ (0-1))) \\
\longrightarrow & 4 + (2 + (\textbf{if}\ \textit{true}\ \textbf{then}\ 0\ \textbf{else if}\ (0\ \textit{mod}\ 2) = 0\ \textbf{then}\ 0 + F\ (0-2)\ \textbf{else}\ F\ (0-1)))
\end{aligned}
$$

$\longrightarrow 4 + (2 + (0))$
$\longrightarrow^{*} 6$

(c) Give a typing derivation for the following program. What happens if you execute the program?

$$\mu p\colon(\textbf{int} \to \textbf{int}) \times (\textbf{int} \to \textbf{int}).\ (\lambda n\colon\textbf{int}.\ n + 1, \#1\ p)$$

Answer: *For brevity, we write τ_p for the type $(\textbf{int} \to \textbf{int}) \times (\textbf{int} \to \textbf{int})$.*

$$\textsc{T-Rec}\ \dfrac{\textsc{T-Pair}\ \dfrac{\textsc{T-Abs}\ \dfrac{\textsc{T-Sum}\ \dfrac{\textsc{T-Var}\ \dfrac{}{p\colon\tau_p, n\colon\textbf{int} \vdash n\colon\textbf{int}} \quad \textsc{T-Int}\ \dfrac{}{p\colon\tau_p, n\colon\textbf{int} \vdash 1\colon\textbf{int}}}{p\colon\tau_p, n\colon\textbf{int} \vdash n + 1\colon\textbf{int}}}{p\colon\tau_p \vdash \lambda n\colon\textbf{int}.\ n + 1\colon\textbf{int} \to \textbf{int}} \quad \textsc{T-Proj}\ \dfrac{\textsc{T-Var}\ \dfrac{}{p\colon\tau_p \vdash p\colon\tau_p}}{p\colon\tau_p \vdash \#1\ p\colon\textbf{int} \to \textbf{int}}}{p\colon\tau_p \vdash (\lambda n\colon\textbf{int}.\ n + 1, \#1\ p)\colon\tau_p}}{\vdash \mu p\colon\tau_p.\ (\lambda n\colon\textbf{int}.\ n + 1, \#1\ p)\colon\tau_p}$$

Now, if you actually tried to execute this expression under a Call-By-Name semantics, it would unfold the recursive expression to $(\lambda n\colon\textbf{int}.\ n + 1, \#1\ P)$, where P is the recursive expression $\mu p\colon(\textbf{int} \to \textbf{int}) \times (\textbf{int} \to \textbf{int}).\ (\lambda n\colon\textbf{int}.\ n + 1, \#1\ p)$. While the first element of the pair is a value, the second $\#2\ P$ is not, and so we would attempt to evaluate that expression. However, that requires evaluating the expression $P \equiv \mu p\colon(\textbf{int} \to \textbf{int}) \times (\textbf{int} \to \textbf{int}).\ (\lambda n\colon\textbf{int}.\ n + 1, \#1\ p)$.

So, under Call-by-Name semantics, the program will not terminate.

3 References

(a) Give a typing derivation for the following program.

$$\begin{aligned}&\text{let } a\colon\textbf{int ref} = \text{ref } 4 \text{ in}\\ &\text{let } b\colon(\textbf{int} \to \textbf{int})\ \textbf{ref} = \text{ref } \lambda x\colon\textbf{int}.\ x + 38 \text{ in}\\ &!b\ !a\end{aligned}$$

Answer: *For brevity, we will write e for the expression above, and e_b for the subexpression **let** $b : (\textbf{int} \to \textbf{int})\ \textbf{ref} = \textit{ref}\ \lambda x\colon\textbf{int}.\ x + 38\ \textit{in}\ !b\ !a$*

$$\textsc{T-Let}\ \dfrac{\textsc{T-Alloc}\ \dfrac{\textsc{T-Int}\ \dfrac{}{\vdash 4\colon\textbf{int}}}{\vdash \textit{ref}\ 4\colon\textbf{int ref}} \quad \textsc{T-Let}\ \dfrac{\textsc{T-Alloc}\ \dfrac{\textsc{T-Abs}\ \dfrac{\dfrac{\vdots_1}{a\colon\textbf{int ref}, x\colon\textbf{int} \vdash x + 38\colon\textbf{int}}}{a\colon\textbf{int ref} \vdash \lambda x\colon\textbf{int}.\ x + 38\colon\textbf{int} \to \textbf{int}}}{a\colon\textbf{int ref} \vdash \textit{ref}\ \lambda x\colon\textbf{int}.\ x + 38\colon(\textbf{int} \to \textbf{int})\ \textbf{ref}} \quad \dfrac{\vdots_2}{a\colon\textbf{int ref}, b\colon(\textbf{int} \to \textbf{int})\ \textbf{ref} \vdash\ !b\ !a\colon\textbf{int}}}{a\colon\textbf{int ref} \vdash e_b\colon\textbf{int}}}{\vdash e\colon\textbf{int}}$$

The subderivation marked $\vdots_1$ is:

$$\textsc{T-Add}\ \dfrac{\textsc{T-Var}\ \dfrac{}{a\colon\textbf{int ref}, x\colon\textbf{int} \vdash x\colon\textbf{int}} \quad \textsc{T-Int}\ \dfrac{}{a\colon\textbf{int ref}, x\colon\textbf{int} \vdash 38\colon\textbf{int}}}{a\colon\textbf{int ref}, x\colon\textbf{int} \vdash x + 38\colon\textbf{int}}$$

The subderivation marked $\vdots_2$ *is:*

$$\dfrac{\dfrac{\dfrac{}{\Gamma_{ab} \vdash b : (\textbf{\textit{int}} \to \textbf{\textit{int}})\ \textbf{\textit{ref}}}\ \text{T-Var}}{\Gamma_{ab} \vdash\ !b : \textbf{\textit{int}} \to \textbf{\textit{int}}}\ \text{T-Deref} \qquad \dfrac{\dfrac{}{\Gamma_{ab} \vdash a : \textbf{\textit{int ref}}}\ \text{T-Var}}{\Gamma_{ab} \vdash\ !a : \textbf{\textit{int}}}\ \text{T-Deref}}{\Gamma_{ab} \vdash\ !b\ !a : \textbf{\textit{int}}}\ \text{T-App}$$

where $\Gamma_{ab} = a : \textbf{\textit{int ref}}, b : (\textbf{\textit{int}} \to \textbf{\textit{int}})\ \textbf{\textit{ref}}$.

(b) Execute the program above for 4 small steps, to get configuration $\langle e, \sigma\rangle$. What is an appropriate Σ such that $\emptyset, \Sigma \vdash e : \tau$ and $\Sigma \vdash \sigma$?

Answer:

$$\begin{aligned}
&\langle \textit{let}\ a : \textbf{\textit{int ref}} = \textit{ref}\ 4\ \textit{in let}\ b : (\textbf{\textit{int}} \to \textbf{\textit{int}})\ \textbf{\textit{ref}} = \textit{ref}\ \lambda x : \textbf{\textit{int}}.\ x + 38\ \textit{in}\ !b\ !a, \emptyset\rangle \\
\longrightarrow &\langle \textit{let}\ a : \textbf{\textit{int ref}} = \ell_a\ \textit{in let}\ b : (\textbf{\textit{int}} \to \textbf{\textit{int}})\ \textbf{\textit{ref}} = \textit{ref}\ \lambda x : \textbf{\textit{int}}.\ x + 38\ \textit{in}\ !b\ !a, [\ell_a \mapsto 4]\rangle \\
\longrightarrow &\langle \textit{let}\ b : (\textbf{\textit{int}} \to \textbf{\textit{int}})\ \textbf{\textit{ref}} = \textit{ref}\ \lambda x : \textbf{\textit{int}}.\ x + 38\ \textit{in}\ !b\ !\ell_a, [\ell_a \mapsto 4]\rangle \\
\longrightarrow &\langle \textit{let}\ b : (\textbf{\textit{int}} \to \textbf{\textit{int}})\ \textbf{\textit{ref}} = \ell_b\ \textit{in}\ !b\ !\ell_a, [\ell_a \mapsto 4, \ell_b \mapsto \lambda x : \textbf{\textit{int}}.\ x + 38]\rangle \\
\longrightarrow &\langle !\ell_b\ !\ell_a, [\ell_a \mapsto 4, \ell_b \mapsto \lambda x : \textbf{\textit{int}}.\ x + 38]\rangle
\end{aligned}$$

An appropriate store typing context is $\Sigma = \ell_a \mapsto \textbf{\textit{int}}, \ell_b \mapsto \textbf{\textit{int}} \to \textbf{\textit{int}}$.