Chapter 09

Newton Ni

August 17, 2020

1 9.2.1

The pure simply typed lambda-calculus with no base types is actually degenerate*, in the sense that it has no well-typed terms at all. Why?*

There are no base cases in the definition of types. There are only function types $\texttt{T} ::= \texttt{T} \to \texttt{T}$, which cannot generate a finite type.

2 9.2.2

Show (by drawing derivation trees) that the following terms have the indicated types:

$$\texttt{f:Bool} \to \texttt{Bool} \vdash \texttt{f (if false then true else false) : Bool}$$

$$\dfrac{\dfrac{\texttt{f:Bool}\to\texttt{Bool} \in \texttt{f:Bool}\to\texttt{Bool}}{\texttt{f:Bool}\to\texttt{Bool} \vdash \texttt{f:Bool}\to\texttt{Bool}}\,\text{Var} \qquad \dfrac{\dfrac{}{\vdash \texttt{false:Bool}}\,\texttt{F} \quad \dfrac{}{\vdash \texttt{true:Bool}}\,\texttt{T} \quad \dfrac{}{\vdash \texttt{false:Bool}}\,\texttt{F}}{\texttt{if false then true else false:Bool}}\,\text{If}}{\texttt{f (if false then true else false):Bool}}\,\text{App}$$

$$\texttt{f:Bool} \to \texttt{Bool} \vdash \lambda\texttt{x:Bool. f (if x then false else x):Bool} \to \texttt{Bool}$$

$$\dfrac{\dfrac{\dfrac{\ldots}{\ldots \vdash \texttt{f:Bool} \to \texttt{Bool}}\,\text{Var} \qquad \dfrac{\dfrac{\ldots}{\ldots \vdash \texttt{x:Bool}}\,\text{Var} \quad \dfrac{}{\vdash \texttt{false:Bool}}\,\texttt{F} \quad \dfrac{\ldots}{\ldots \vdash \texttt{x:Bool}}\,\text{Var}}{\ldots \vdash \texttt{if x then false else x:Bool}}\,\text{If}}{\texttt{f:Bool} \to \texttt{Bool}, \texttt{x:Bool} \vdash \texttt{f (if x then false else x):Bool}}\,\text{App}}{\texttt{f:Bool} \to \texttt{Bool} \vdash \ldots\texttt{:Bool} \to \texttt{Bool}}\,\text{Abs}$$

3 9.2.3

Find a context Γ *under which the term* `f x y` *has type* `Bool`*. Can you give a simple description of the set of* all *such contexts?*

The type of f could be any $\texttt{T}_1 \to \texttt{T}_2 \to \texttt{Bool}$, where $x : \texttt{T}_1$ and $y : \texttt{T}_2$ and the latter two types are unconstrained.

4 9.3.2

Is there any context Γ *and type* $\mathtt{T}$ *such that* $\Gamma \vdash \mathtt{x\ x{:}T}$*? If so, give* Γ *and* $\mathtt{T}$ *and show a typing derivation; if not, prove it.*

No. Suppose $\mathtt{x}$: $\mathtt{T_1}{\to}\mathtt{T_2}$. To successfully apply it to itself, T-App requires that $\mathtt{x}$: $\mathtt{T_1}$. But there is no finite type such that $\mathtt{T_1} = \mathtt{T_1} \to \mathtt{T_2}$.

5 9.3.9

Prove the preservation theorem: $\Gamma \vdash \mathtt{t} : \mathtt{T} \wedge \mathtt{t} \longrightarrow \mathtt{t}' \implies \Gamma \vdash \mathtt{t}' : \mathtt{T}$.

Proof. By induction on the small-step derivation for $\mathtt{t} \longrightarrow \mathtt{t}'$. Our induction hypothesis is:

$$H(\mathtt{t} \longrightarrow \mathtt{t}') : \Gamma \vdash \mathtt{t} : \mathtt{T} \implies \Gamma \vdash \mathtt{t}' : \mathtt{T}$$

Case (E-App1). Here we have $\mathtt{t} = \mathtt{t_1}\ \mathtt{t_2}$, where:

1. $\mathtt{t_1}\ \mathtt{t_2} \longrightarrow \mathtt{t_1}'\ \mathtt{t_2}$
2. $\mathtt{t_1} \longrightarrow \mathtt{t_1}'$

By the inversion lemma, we have:

3. $\Gamma \vdash \mathtt{t_1}$: $\mathtt{T_1}{\to}\mathtt{T}$
4. $\Gamma \vdash \mathtt{t_2}$: $\mathtt{T_1}$

We apply the induction hypothesis to (2) and (3) to obtain

5. $\Gamma \vdash \mathtt{t_1}'$: $\mathtt{T_1}{\to}\mathtt{T}$

We then apply T-App to (4) and (5) to conclude that $\mathtt{t}' = \mathtt{t_1}'\ \mathtt{t_2} : \mathtt{T}$, as desired.

Case (E-App2). This case is analagous to above, except we apply the induction hypothesis to $\mathtt{t_2}$ instead.

Case (E-AppAbs). Here we have:

1. $\mathtt{t} = (\lambda \mathtt{x}.\ \mathtt{t_{12}})\mathtt{v_2}$
2. $\mathtt{t}' = [\mathtt{x} \mapsto \mathtt{v_2}]\ \mathtt{t_{12}}$

By the inversion lemma, we have:

3. $\Gamma \vdash \lambda\mathtt{x}.\ \mathtt{t_{12}}$: $\mathtt{T_1}{\to}\mathtt{T}$
4. $\Gamma \vdash \mathtt{v_2} : \mathtt{T_1}$

By T-Abs, we have:

5. $\Gamma,\ \mathtt{x} : \mathtt{T_1} \vdash \mathtt{t_{12}}$: $\mathtt{T}$

By applying the substitution lemma to (5) and (4), we have:

6. $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{v_2}]\ \mathtt{t_{12}} : \mathtt{T}$

As desired.

□

6 9.3.10

In Exercise 8.3.6 we investigated the subject expansion *property for our simple calculus of typed arithmetic expressions. Does it hold for the 'functional part' of the simply typed lambda-calculus? That is, suppose* $\mathtt{t}$ *does not contain any conditional expressions. Do* $\mathtt{t} \longrightarrow \mathtt{t}'$ *and* $\Gamma \vdash \mathtt{t}' : \mathtt{T}$ *imply* $\Gamma \vdash \mathtt{t} : \mathtt{T}$?

TODO

7 9.4.1

Which of the rules for the type `Bool` *in Figure 8-1 are introduction rules and which are elimination rules? What about the rules for* `Nat` *in figure 8-2?*

For `Bool`, T-True and T-False are introduction rules, while T-If is an elimination rule.

For `Nat`, T-Zero, T-Succ, and T-Pred are introduction rules, while T-IsZero is an elimination rule.