Notes on Normalization Proof (Pierce Ch. 12)

Andrew Tolmach

May 7, 2024

Here are a few comments on the normalization proof as presented in Pierce Ch. 12.

1. Although Pierce is not too explicit about this, the definition of $R_T(t)$ (for every case of $T$) needs to include the fact that $\vdash t : T$, i.e. $t$ is typable in the empty environment. An immediate consequence is that $t$ must be closed, which is needed at one point in the proof.

2. At various points in the proof, we need to make use of the following facts about the language:

   - (one-step) Determinism: $t \to t'$ and $t \to t''$ implies $t' = t''$.
   - Progress: $\vdash t : T$ implies that either $t$ is a value or there exists $t'$ such that $t \to t'$.
   - Preservation: If $\vdash t : T$ and $t \to t'$ then $\vdash t' : T$.

3. A corollary of Progress is that the notion of normal form (term that cannot take a step) and value coincide for well-typed terms. Thus when we say that $t$ halts, we could equally well say that it multisteps to a value $v$. This fact is used by Pierce implicitly at a few points.

4. In the statement of Lemma 12.1.4, the assumption $\vdash t : T$ is not actually needed in the forward ($\Rightarrow$) direction, since it is built into the definition of $R_T(t)$. But it is needed in the reverse ($\Leftarrow$) direction, since the language does not have “subject expansion,” i.e. there are ill-typed terms that step to well-typed ones (cf. Exercise 9.3.10). It is worth your trouble to write out the proof of the reverse direction explicitly.

5. Also in the forward direction of Lemma 12.1.4, the claim that $t \to t'$ and $t$ halts together imply that $t'$ also halts is making an implicit appeal to Determinism. (If the language were not deterministic, $t'$ might not be the first step on the path by which $t$ halts, and indeed might not halt at all.)

6. Here is a more careful rendering of the T-Abs case of Lemma 12.1.5.

First, we write the Lemma statement more compactly and precisely as follows:

**Lemma** If

- $\Gamma \vdash t : T$ where $\Gamma = x_1 : T_1, \ldots, x_n : T_n$ for distinct $x_1, \ldots, x_n$;
- $v_1, \ldots, v_n$ are values and $T_1, \ldots, T_n$ are types such that $R_{T_i}(v_i)$ for each $i$; and
- $\sigma = [x_1 \mapsto v_1] \cdots [x_n \mapsto v_n]$
then \( R_T(\sigma t) \).

Note that \( \sigma \) is a composition of \( n \) substitutions. Since the \( x_i \) are required to be distinct and the \( v_i \) are closed (because \( R_T(v_i) \) implies \( \vdash v_i : T_i \)), the order of the substitutions doesn’t matter.

The proof is by induction on typing derivation \( \Gamma \vdash t : T \).

The T-Abs case goes as follows. Since we have \( \vdash t : T \) using T-Abs, we know:

\[
\begin{align*}
t & = \lambda x : S_1 . s_2 \\
\Gamma, x : S_1 & \vdash s_2 : S_2 \\
T & = S_1 \rightarrow S_2.
\end{align*}
\]

We wish to show that \( R_{S_1 \rightarrow S_2}(\sigma t) \), i.e. that

1. \( \vdash \sigma t : S_1 \rightarrow S_2 \)
2. \( \sigma t \) halts
3. If \( R_{S_i}(s) \) then \( R_{S_2}(\sigma t s) \)

(i) We know that \( \Gamma \vdash t : T \). Since \( R_{T_i}(v_i) \), we have that \( \vdash v_i : T_i \) for each \( i \). So, by repeated application of the Substitution Lemma 9.3.8 (proved as part of Preservation) we have that \( \Gamma \vdash \sigma t : T \). In fact, by a slightly stronger version of that lemma, since the \( v_i \) are typable in the empty context and \( \sigma t \) no longer contains any of the \( x_i \) free, we can conclude \( \vdash \sigma t : T \), i.e. \( \vdash \sigma t : S_1 \rightarrow S_2 \), as required.

(ii) Since \( t = \lambda x : S_1 . s_2 \), \( \sigma t \) is also a lambda abstraction, which is already a value, so certainly halts.

(iii) Choose any \( s \) such that \( R_{S_1}(s) \). We must show that \( R_{S_2}(\sigma t s) \), i.e. \( R_{S_2}(\sigma(\lambda x : S_1 . s_2) s) \).

From the supposition, we know that \( \vdash s : S_1 \) and that \( s \) halts. Indeed, by Progress, \( s \rightarrow^* v \) for some value \( v \). By repeated use of Lemma 12.1.4 (\( \Rightarrow \)), we have \( R_{S_1}(v) \). Thus we can build an extended substitution \( \sigma' = \sigma[x \mapsto v] \) matching the extended context \( \Gamma' = \Gamma, x : S_1 \). (We can assume that \( x \) is not among the \( x_i \) in \( \sigma \) and \( \Gamma \) by renaming it if necessary, following Convention 5.3.4.) Since \( \Gamma' \vdash s_2 : S_2 \) is a sub-derivation of \( \Gamma \vdash t : T \), we can apply induction and obtain \( R_{S_2}(\sigma' s_2) \).

But we also know that

\[
(\sigma(\lambda x : S_1 . s_2) s) = (\lambda s : S_1 . \sigma s_2) s \rightarrow^* (\lambda x : S_1 . \sigma s_2) v \rightarrow [x \mapsto v](\sigma s_2) = \sigma' s_2
\]

where the first equality is by the definition of substitution (assuming renaming of bound variables as necessary, as in Definition 5.3.5), the multisteps are by repeated use of E-App2, the last step is by E-AppAbs, and the final equality is justified by the fact that \( x \) is not among the \( x_i \) and that \( v \) and all the \( v_i \) are closed. By repeated use of Lemma 12.1.4 (\( \Leftarrow \)), we have \( R_{S_2}(\sigma(\lambda x : S_1 . s_2) s) \).

But this is what we needed to show. \( \square \)

7. Note the errata for the suggested solution for 12.1.7. In particular, in case T-If, the reasoning beginning “We now continue by induction on T” is much more complicated than it needs to be. At this point we can instead use canonical forms to reason about the possible values of \( v_1 \) and quickly reach the desired result.