Chapter 08

Newton Ni

August 9, 2020

1 8.2.3

Prove that every subterm of a well-typed term is well typed.

Proof. By induction on the typing derivation of term $\mathtt{t}$. Our inductive hypothesis is:

$$H(\mathtt{t}) : \mathtt{t} \textit{ is well-typed} \implies \textit{all subterms of } \mathtt{t} \textit{ are well-typed}$$

We proceed by case analysis on the final step of the derivation.

Case (`true`, `false`, 0). These terms are well-typed by the inversion lemma, and have no subterms.

Case (`if` $\mathtt{t}_1$ `then` $\mathtt{t}_2$ `else` $\mathtt{t}_3$). By the inversion lemma, this term is well-typed, and we have three well-typed subterms:

- $\mathtt{t}_1 : \mathtt{Bool}$
- $\mathtt{t}_2 : \mathtt{R}$
- $\mathtt{t}_3 : \mathtt{R}$

We can apply the inductive hypothesis to each.

Case (`succ` $\mathtt{t}_1$, `pred` $\mathtt{t}_1$, `iszero` $\mathtt{t}_1$). These cases are analagous to the previous case, except they only have one subterm each.

□

2 8.3.4

Restructure [the PRESERVATION] *proof so that it goes by induction on evaluation derivations rather than typing derivations.*

Proof. By induction on the derivation of $\mathtt{t} \longrightarrow \mathtt{t}'$. Our inductive hypothesis is:

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

We proceed by case analysis on the last step of the derivation. In all cases, we have that $\mathtt{t} : T$.

Case (E-IfTrue). Here, $\mathtt{t} \longrightarrow \mathtt{t}'$ is:

$$\texttt{if true then } \mathtt{t}_2 \texttt{ else } \mathtt{t}_3 \longrightarrow \mathtt{t}_2$$

By the inversion lemma, $\mathtt{t} : T \implies \mathtt{t}_2 : T$ as desired.

Case (E-IfFalse). This case is analagous to the previous one.

Case (E-If). Here, we have:

$$\texttt{if } \mathtt{t}_1 \texttt{ then } \mathtt{t}_2 \texttt{ else } \mathtt{t}_3 \longrightarrow \texttt{if } \mathtt{t}_1\text{'} \texttt{ then } \mathtt{t}_2 \texttt{ else } \mathtt{t}_3$$

$$\mathtt{t}_1 \longrightarrow \mathtt{t}_1{}'$$

By the inversion lemma, we have:

- $\mathtt{t}_1 : \mathtt{Bool}$
- $\mathtt{t}_2 : \mathtt{T}$
- $\mathtt{t}_3 : \mathtt{T}$

By the inductive hypothesis, we also have $\mathtt{t}_1{}' : \mathtt{Bool}$. We conclude $\mathtt{t}' : \mathtt{T}$ by T-If.

Case (E-Succ). Here, we have:

$$\texttt{succ } \mathtt{t}_1 \longrightarrow \texttt{succ } \mathtt{t}_1\text{'}$$

$$\mathtt{t}_1 \longrightarrow \mathtt{t}_1{}'$$

By the inversion lemma, we have $\mathtt{t}_1 : \mathtt{Nat}$. By the inductive hypothesis, we have $\mathtt{t}_1{}' : \mathtt{Nat}$. We conclude that $\texttt{succ } \mathtt{t}_1\text{'} : \mathtt{Nat}$ by T-Succ.

Case (E-Pred, E-IsZero). These cases are analagous to the previous case.

Case (E-PredZero, E-IsZeroZero). These cases follow directly from the inversion lemma and typing rules T-Zero and T-True, respectively.

Case (E-PredSucc). Here, we have:

$$\texttt{pred (succ } \mathtt{t}_1\texttt{)} \longrightarrow \mathtt{t}_1$$

By two applications of the inversion lemma, we have:

- $\texttt{succ } \mathtt{t}_1 : \mathtt{Nat}$
- $\mathtt{t}_1 : \mathtt{Nat}$

Case (E-IsZeroSucc). This case is analagous to the previous case.

□

3 8.3.5

The evaluation rule E-PredZero *(Figure 3–2) is a bit counterintuitive: we might feel that it makes more sense for the predecessor of zero to be undefined, rather than being defined to be zero. Can we achieve this simply by removing the rule from the definition of single-step evaluation?*

No, because the term `pred 0` would be well-typed by T-Zero and T-Pred, but would be stuck, violating the progress property.

4 8.3.6

Having seen the subject reduction property, it is reasonable to wonder whether the opoosite property—subject expansion—also holds. Is it always the case that, if $\mathtt{t} \longrightarrow \mathtt{t}'$ *and* $\mathtt{t}' : \mathtt{T}$ *, then* $\mathtt{t} : \mathtt{T}$*? If so, prove it. If not, give a counterexample.*

No, since ill-typed terms can also take a step. For example:

$$\mathtt{if\ true\ then\ 0\ else\ true} \longrightarrow 0$$

5 8.3.7

Suppose our evaluation relation is defined in the big-step style, as in exercise 3.5.17. How should the intuitive property of type safety be formalized?

Type preservation remains the same, but progress is modified to state that all well-typed terms evaluate to a value.

6 8.3.8

Suppose our evaluation relation is augmented with rules for reducing nonsensical terms to an explicit `wrong` *state, as in Exercise 3.5.16. Now how should type safety be formalized?*

Type preservation remains the same, but progress is no longer useful, as terms can always take a step (sometimes to `wrong`).