Lecture 3

Arithmetic expressions

This is in Chapter 8 of Pierce's book.

We presented the language

$$e \ ::= \ \mathsf{true} \mid \mathsf{false} \mid \mathsf{if}\ e\ e\ e \mid 0 \mid \mathsf{succ}\ e \mid \mathsf{pred}\ e \mid \mathsf{isZero}\ e$$

with the values

$$v \ ::= \ bv \mid nv \qquad bv \ ::= \ \mathsf{true} \mid \mathsf{false} \qquad nv \ ::= \ 0 \mid \mathsf{succ}\ nv$$

and the one step evaluation rule

$$\frac{}{\mathsf{if\ true}\ e_0\ e_1 \to e_0} \qquad \frac{}{\mathsf{if\ false}\ e_0\ e_1 \to e_1} \qquad \frac{e \to e'}{\mathsf{if}\ e\ e_0\ e_1 \to \mathsf{if}\ e'\ e_0\ e_1}$$

$$\frac{e \to e'}{\mathsf{succ}\ e \to \ \mathsf{succ}\ e'}$$

$$\frac{}{\mathsf{isZero}\ 0 \to \mathsf{true}} \qquad \frac{}{\mathsf{isZero}\ (\mathsf{succ}\ nv) \to \mathsf{false}} \qquad \frac{e \to e'}{\mathsf{isZero}\ e \to \mathsf{isZero}\ e'}$$

$$\frac{}{\mathsf{pred}\ 0 \to 0} \qquad \frac{}{\mathsf{pred}\ (\mathsf{succ}\ nv) \to nv} \qquad \frac{e \to e'}{\mathsf{pred}\ e \to \mathsf{pred}\ e'}$$

For this language we have some expressions that are in normal form but are not values, e.g. $\mathsf{isZero\ true}$ or $\mathsf{if}\ 0\ 0\ 0$.

We write $\mathit{stuck}(e)$ if e is in normal form, i.e. $\neg\exists e'\ e \to e'$ and e is not a value.

Types

We introduce types

$$T \ ::= \ \mathsf{Nat} \mid \mathsf{Bool}$$

with the following *typing rules*. What should be noted is that we use the same formalism of *inference rules* to describe the evaluation relation and the typing system.

$$\frac{}{\mathsf{true} : \mathsf{Bool}} \qquad \frac{}{\mathsf{false} : \mathsf{Bool}} \qquad \frac{e : \mathsf{Bool} \quad e_0 : T \quad e_1 : T}{\mathsf{if}\ e\ e_0\ e_1 : T}$$

$$\frac{}{0 : \mathsf{Nat}} \qquad \frac{e : \mathsf{Nat}}{\mathsf{succ}\ e : \mathsf{Nat}} \qquad \frac{e : \mathsf{Nat}}{\mathsf{pred}\ e : \mathsf{Nat}} \qquad \frac{e : \mathsf{Nat}}{\mathsf{isZero}\ e : \mathsf{Bool}}$$

Note that $e = \mathsf{if\ true}\ 0\ \mathsf{true}$ is *not* of type Nat but we have $e \to 0$ and $0 : \mathsf{Nat}$

Theorem 0.1 *(progress) If $e : T$ then e is a value or $\exists e'\ e \to e'$*

Theorem 0.2 *(preservation) If $e : T$ and $e \to e'$ then $e' : T$*

Theorem 0.3 *If $e : T$ and $e \to^* e'$ then e' is not stuck.*

This expresses that "well-typed programs cannot go wrong" which was first proved by Milner 1978 using a different method.

Confluence

All the evaluation relations we have seen so far are deterministic. A more general notion is to be *confluent*: if $e \to^* e_1$ and $e \to^* e_2$ then there exists e' such that $e_1 \to^* e'$ and $e_2 \to^* e'$.

Theorem 0.4 *If $\to$ is deterministic it is confluent.*

It is simply because in this case if $e \to^* e_1$ and $e \to^* e_2$ then there we have $e_1 \to^* e_2$ (and we can take $e' = e_2$) or $e_2 \to^* e_1$ (and we can take $e' = e_1$).

Theorem 0.5 *If $\to$ is confluent and $NF(e, e_1)$ and $NF(e, e_2)$ then $e_1 = e_2$.*

We recall that $NF(e, e')$ means that $e \to^* e'$ and e' is in normal form.

Untyped λ-calculus

This is in Chapter 5 of Pierce's book and in the Agda book of Kokke and Wadler.

We now introduce a programming language such that the predicate $\exists e'\ NF(e, e')$ (halting problem) is *not* decidable. Historically, this was actually the *first* example of a provably non decidable problem in mathematics (Church, 1936).

$$e \ ::=\ x \mid e\ e \mid \lambda x\ e$$

We define the set of free variables of e as follows.

$$FV(x) = \{x\} \qquad FV(e_0\ e_1) = FV(e_0) \cup FV(e_1) \qquad FV(\lambda\ x\ e) = FV(e) - \{x\}$$

An expression e is *closed* if we have $FV(e) = \emptyset$.
We define subsitution $e(t/x)$ for t *closed.* It is by case on e

- if $e = x$ then $e(t/x) = t$
- if $e = y \neq x$ then $e(t/x) = y$
- if $e = e_0\ e_1$ then $e(t/x) = e_0(t/x)\ e_1(t/x)$
- if $e = \lambda x\ e'$ then $e(t/x) = e$
- if $e = \lambda y\ e'$ with $y \neq x$ then $e(t/x) = \lambda y\ e'(t/x)$

We then define a *value* to be a closed expression of the form $\lambda x\ e$.

$$v \ ::= \ \lambda x\ e$$

We define the *call-by-value* evaluation relation $e \to e'$ for e and e' *closed* expressions

$$\frac{e \to e'}{e\ e_1 \to e'\ e_1} \qquad \frac{e_1 \to e_1'}{v\ e_1 \to v\ e_1'} \qquad \frac{}{(\lambda x\ e)\ v \to e(v/x)}$$

Note that if $\delta = \lambda x\ x\ x$ then δ is a value and $\delta\ \delta \to \delta\ \delta$, so we have $\neg\exists e'\ NF(\delta\ \delta, e')$

Church (1936) has essentially proved the following result.

Theorem 0.6 *The predicate $\exists e'\ NF(e, e')$ is* not *decidable.*