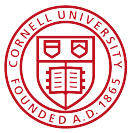
CS 4110 – Programming Languages and Logics
Lecture #20: Normalization



1 Introduction

A major limitation of the simply-typed lambda-calculus is that we can no longer write recursive functions. Consider the nonterminating expression $\Omega = (\lambda x.\, x\ x)\ (\lambda x.\, x\ x)$. What type does it have? Let's suppose that the type of $\lambda x.\, x\ x$ is $\tau \to \tau'$. But $\lambda x.\, x\ x$ is applied to itself! So that means that the type of $\lambda x.\, x\ x$ is the argument type τ. So we have that τ must be equal to $\tau \to \tau'$. There is no such type for which this equality holds. (At least, not in this type system...)

This means that every well-typed program in the simply-typed lambda calculus terminates. Formally:

Theorem (Normalization)**.** *If* $\vdash e : \tau$ *then there exists a value* v *such that* $e \to^* v$.

The rest of this lecture is devoted to proving this theorem.

2 Notation

We work with the simply-typed lambda calculus over unit,

$$\begin{array}{rcl} e & ::= & x \mid () \mid \lambda x : \tau.\, e \mid e_1\ e_2 \\ v & ::= & () \mid \lambda x : \tau.\, e \\ \tau & ::= & \textbf{unit} \mid \tau_1 \to \tau_2 \end{array}$$

with the standard call-by value semantics:

$$E ::= [\cdot] \mid E\ e \mid v\ E$$

$$\textsc{Context}\ \frac{e \to e'}{E[e] \to E[e']}$$

$$\beta\textsc{-reduction}\ \frac{}{(\lambda x.\, e)\ v \to e\{v/x\}}$$

3 A First Attempt

As a first attempt toward proving normalization, let us try a proof by structural induction on e. We will need the following lemmas, all of which are standard. Each of these lemmas can be proved by straightforward induction on the typing derivation. We leave these proofs as an exercise.

Lemma (Inversion)**.**

- *If* $\Gamma \vdash x : \tau$ *then* $\Gamma(x) = \tau$
- *If* $\Gamma \vdash \lambda x : \tau_1 . e : \tau$ *then* $\tau = \tau_1 \to \tau_2$ *and* $\Gamma, x : \tau_1 \vdash e : \tau_2$.
- *If* $\Gamma \vdash e_1\ e_2 : \tau$ *then* $\Gamma \vdash e_1 : \tau' \to \tau$ *and* $\Gamma \vdash e_2 : \tau'$.

Lemma (Canonical Forms)**.**

- *If* $\Gamma \vdash v : \textbf{\textit{unit}}$ *then* $v = ()$
- *If* $\Gamma \vdash v : \tau_1 \to \tau_2$ *then* $v = \lambda x : \tau_1 . e$ *and* $\Gamma, x : \tau_1 \vdash e : \tau_2$.

Now let us attempt to prove prove the main theorem.

Theorem (Normalization)**.** *If* $\vdash e : \tau$ *then there exists a value* v *such that* $e \to^* v$.

Proof. By structural induction on e.

Case $e = x$**:**
By inversion, we have that the empty typing context maps x to τ, which is a contradiction. Hence, the case vacuously holds.

Case $e = ()$**:**
Immediate since e is already a value.

Case $e = \lambda x : \tau . e$**:**
Immediate since e is already a value.

Case $e = e_1\ e_2$**:**
By inversion we have $\vdash e_1 : \tau' \to \tau$ and $\vdash e_2 : \tau'$. Hence, by induction hypothesis there exist v_1 and v_2 such that that $e_1 \to^* v_1$ and $e_2 \to^* v_2$. Moreover, by canonical forms we have that $v_1 = \lambda x : \tau' . e'$. Hence, $v_1\ v_2 \to e'\{v_2/x\}$.

At this point we would *like* to apply the induction hypothesis to $e'\{v_2/x\}$ to show that it also evaluates to a value, but doing this would *not* be valid—the induction hypothesis only applies to immediate subexpressions of e! Moreover, we cannot get around this by using the other induction principles we have seen before, such as induction on the size of the expression or on the typing derivation—these induction hypotheses do not apply to $e'\{v_2/x\}$ either!

We need a different proof technique.

□

4 Logical Relations

One way to prove normalization for the simply-typed lambda calculus, which was invented by Tait in 1967, is to use a *logical relation*. The idea in a logical relation is to define a predicate on expressions indexed on types that captures the property we want. At base types this set will simply contain all expressions satisfying the property. At function types, we additionally require that the property be preserved whenever we apply the function to an argument of appropriate type that also has the property.

Formally, we define the following predicate $R_\tau(e)$ inductively on τ. We use e halts as an abbreviation for exists v such that $e \to^* v$.

Definition (Logical Relation)**.**

- $R_{\mathbf{unit}}(e)$ iff $\vdash e : \mathbf{unit}$ and e halts.
- $R_{\tau_1 \to \tau_2}(e)$ iff $\vdash e : \tau_1 \to \tau_2$ and e halts, and for every e' such that $R_{\tau_1}(e')$ we have $R_{\tau_2}(e\ e')$.

Normalization then follows from the following two lemmas:

Lemma. *If* $\vdash e : \tau$ *then* $R_\tau(e)$

Lemma. *If* $R_\tau(e)$ *then* e *halts.*

The proof of the second is trivial, since halting is built into the definition of the logical relation. To prove the first, we need the following additional lemma:

Lemma. *If* $\vdash e : \tau$ *and* $e \to e'$ *then* $R_\tau(e)$ *iff* $R_\tau(e')$*.*

We leave the proof of this lemma as an exercise.

Finally, the lemma above is proved as follows. We strengthen the induction hypothesis to allow a non-empty typing context.

Lemma. *If* $x_1 : \tau_1 \ldots x_k : \tau_k \vdash e : \tau$*, and* v_1 *to* v_k *are values such that* $\vdash v_1 : \tau_1$ *to* $\vdash v_k : \tau_k$ *and* $R_{\tau_1}(v_1)$ *to* $R_{\tau_k}(v_k)$*, then* $R_\tau(e\{v_1/x_1\} \ldots \{v_k/x_k\})$*.*

Proof. By structural induction on e.

- **Case** $e = x$**:**
  By inversion we have that $x = x_i$ and $\tau = \tau_i$ for some i. By definition, $e\{v_1/x_1\} \ldots \{v_k/x_k\} = v_i$. We have $R_{\tau_i}(v_i)$ by assumption.
- **Case** $e = ()$**:**
  By inversion we have that $\tau = \mathbf{unit}$. By definition, $e\{v_1/x_1\} \ldots \{v_k/x_k\} = ()$. We obtain $R_{\mathbf{unit}}(())$ by the definition of the logical relation as $\vdash () : \mathbf{unit}$ and $()$ halts.
- **Case** $e = \lambda x : \tau'.\, e'$**:**
  By inversion we have $\tau = \tau' \to \tau''$ and $x_1 : \tau_1 \ldots x_k : \tau_k \vdash e' : \tau''$. We immediately have that $(\lambda x : \tau'.\, e')\{v_1/x_1\} \ldots \{v_k/x_k\}$ halts since it is already a value.

  Let e'' be an arbitrary expression such that $R_{\tau'}(e'')$. By definition of the logical relation we have $\vdash e'' : \tau'$ and e'' halts. So there exists a v'' such that $e'' \to^* v''$. We have a lemma above stating that evaluation preserves membership in our logical relation; by this lemma, we have that $R_{\tau'}(v'')$. By the induction hypothesis applied to e', which is a subexpression of the current expression e, we have $R_{\tau''}(e'\{v_1/x_1\} \ldots \{v_k/x_k\}\{v''/x\})$.

  Hence, by the definition of the operational semantics and the aforementioned lemma again we also have $R_{\tau''}(e\{v_1/x_1\} \ldots \{v_k/x_k\})\ e'')$. Therefore by the definition of the logical relation we have $R_{\tau' \to \tau''}(e\{v_1/x_1\} \ldots \{v_k/x_k\})$ as required.
- **Case** $e = e_1\ e_2$**:** Left as an exercise.

□