CS611 Lecture 28 — Logical Relations, Strong Normalization — November 1, 2000

Scribe: Nikita Proskourine, Matt Harren — Lecturer: Andrew Myers

In the last class we showed that the typing rules for the typed lambda calculus ($\lambda^{\rightarrow}$) are sound with respect to operational semantics. Today we give another argument that all $\lambda^{\rightarrow}$ programs terminate using logical relations.

How do we show that all $\lambda^{\rightarrow}$ programs terminate? One approach would be to show that the size of a program decreases as it executes; however, this is not necessarily the case. For example, take

$$(\lambda f : int \rightarrow int\ .\ (+\ (f\ 0)\ (f\ 1)))\ (\lambda y : int\ .\ (*\ y\ 2))$$

Instead, we will show that the size of the type of a program decreases as the program executes.

- Idea: Show $\vdash e : \tau \Rightarrow e \in \mathsf{SN}_\tau$, where SN_τ is the set of strongly normalizing expressions of type τ.

1 Stable Expressions

We'll use an induction on the type derivation to prove our claim. However, we will need a somewhat stronger induction hypothesis in order to prove the application case. So instead of strongly normalizing expressions, we will show the stronger claim that all typed expressions in $\lambda^{\rightarrow}$ are *stable* expressions. The stable expressions of a type τ are a subset of the strongly normalizing expressions of type τ which always result in stable expressions when applied to other stable expressions.

We define the set T_τ of stable expressions with type τ inductively as follows:

$$\begin{aligned}\mathsf{T}_{\mathsf{int}} &= \{e \mid \vdash e : \mathsf{int}\ \wedge\ e \Downarrow n\}\\ \mathsf{T}_{\tau\rightarrow\tau'} &= \{e \mid \vdash e : \tau \rightarrow \tau'\ \wedge\ e \Downarrow v\ \wedge\ (\forall e' \in \mathsf{T}_\tau\ .\ (e\ e') \in \mathsf{T}_{\tau'})\}\end{aligned}$$

(Additional base types could easily be added, but we will only be concerned with *int* in this proof. Also, note that the definition of T_τ is not recursive, since it is based on smaller terms.)

2 Type Contexts

Currently, we are trying to show $\vdash e : \tau \Rightarrow e \in \mathsf{T}_\tau$. However, this is still not strong enough to allow us to use induction, because we will need to work with expressions with free variables.

Consider a function $\gamma : \mathsf{Var} \rightarrow \mathsf{Expr}$ that maps variables to stable expressions of the correct type:

$$\gamma \models \Gamma \iff \forall x \in \mathsf{dom}(\Gamma)\ .\ \gamma(x) \in \mathsf{T}_{\Gamma(x)}$$

For any γ, define a function $\hat{\gamma} : \mathsf{Expr} \rightarrow \mathsf{Expr}$ that performs the substitution specified by:

$$\begin{aligned}\hat{\gamma}[\![n]\!] &= n\\ \hat{\gamma}[\![x]\!] &= \gamma(x) \qquad \mathsf{if} x \in \mathsf{dom}(\Gamma)\\ \hat{\gamma}[\![x]\!] &= x \qquad\quad \mathsf{if} x \notin \mathsf{dom}(\Gamma)\\ \hat{\gamma}[\![e_0\ e_1]\!] &= \hat{\gamma}[\![e_0]\!]\ \hat{\gamma}[\![e_1]\!]\\ \hat{\gamma}[\![\lambda x : \tau\ .\ e]\!] &= \lambda x : \tau\ .\ \underline{\gamma'}[\![e]\!]\\ &\quad \text{where } \gamma' \text{ is the same as } \gamma \text{ except that it doesn't map } x.\end{aligned}$$

We can now prove this claim:

$$\Gamma \vdash e : \tau \quad \Rightarrow \quad \forall \gamma \models \Gamma\ .\ \hat{\gamma}[\![e]\!] \in \mathsf{T}_\tau$$

Note that when $\Gamma = \emptyset$ and $\gamma = \emptyset$ we have $\emptyset \vdash e : \tau \Rightarrow \hat{\gamma}[\![e]\!] \in \mathsf{T}_\tau$, which implies $\vdash e : \tau \Rightarrow e \in \mathsf{T}_\tau$ (which in turn implies $\vdash e : \tau \Rightarrow e \in \mathsf{SN}_\tau$).

3 The Proof

Prove $\Gamma \vdash e : \tau \Rightarrow \forall \gamma \models \Gamma \,.\, \hat{\gamma}[\![e]\!] \in \mathsf{T}_\tau$ by structural induction on e.

- Case 1: n

$$\Gamma \vdash n : \tau \Rightarrow \forall \gamma \models \Gamma \,.\, n \in \mathsf{T}_{\mathsf{int}}$$

  by definition of $\mathsf{T}_{\mathsf{int}}$.

- Case 2: x
  If $\gamma \models \Gamma$, then $\gamma(x) \in \mathsf{T}_{\Gamma(x)}$. Therefore,

$$\Gamma \vdash x : \mathsf{T}_{\Gamma(x)} \quad \Rightarrow \quad \forall \gamma \models \Gamma \,.\, \hat{\gamma}[\![x]\!] \in \mathsf{T}_{\Gamma(x)}$$

- Case 3: $e_1\ e_2$ (application)
  If $\Gamma \vdash e_1\ e_2 : \tau$, then there exists τ' such that $\Gamma \vdash e_1 : \tau' \to \tau$ and $\Gamma \vdash e_2 : \tau'$.

  For any $\gamma \models \Gamma$, the induction hypothesis tells us that $\hat{\gamma}[\![e_1]\!] \in \mathsf{T}_\tau$ and $\hat{\gamma}[\![e_2]\!] \in \mathsf{T}_{\tau'}$. Since $\hat{\gamma}[\![e_1]\!]$ and $\hat{\gamma}[\![e_2]\!]$ are stable, their application is too (by definition of stable).

$$\begin{aligned}\Gamma \vdash e_1\ e_2 : \tau \quad &\Rightarrow \quad \forall \gamma \models \Gamma \,.\, (\hat{\gamma}[\![e_1]\!]\ \hat{\gamma}[\![e_2]\!]) \in \mathsf{T}_\tau \\ &\Rightarrow \quad \forall \gamma \models \Gamma \,.\, \hat{\gamma}[\![e_1\ e_2]\!] \in \mathsf{T}_\tau\end{aligned}$$

- Case 4: $(\lambda x : \tau \,.\, e)$ (abstraction)
  We'll assume there is an arbitrary $\gamma \models \Gamma$. We need to show:
  - $\vdash \hat{\gamma}[\![(\lambda x : \tau \,.\, e)]\!] : \tau \to \tau'$
    This follows from the Substitution Lemma (This lemma is not proved here, but it is similar to the lemma proved in the last lecture.)
  - $\hat{\gamma}[\![(\lambda x : \tau \,.\, e)]\!] \Downarrow v$.
    All abstractions are values.
  - $\forall e' \in \mathsf{T}_\tau \,.\, (\hat{\gamma}[\![(\lambda x : \tau \,.\, e)]\!] e') \in \mathsf{T}_{\tau'}$ (Stability)
    Substituting free variables while applying e' will give an appropriate set of stable expressions:

$$\hat{\gamma}[\![(\lambda x : \tau \,.\, e)]\!]\ e' = (\lambda x : \tau \,.\, \hat{\gamma}'[\![e]\!])\ e' = \hat{\gamma}'[\![e]\!]\{e'/x\}$$

    Now, we know that $(\lambda x : \tau . \hat{\gamma}'[\![e]\!])e'$ is in $\mathsf{T}_{\tau'}$ if $\hat{\gamma}'[\![e]\!]\{e'/x\}$ is. Note that $\hat{\gamma}'[\![e]\!]\{e'/x\} = \hat{\gamma}''[\![e]\!]$, where $\gamma'' = \gamma[x \mapsto e']$.

    From the typing rule for abstractions, we know that $\Gamma[x \mapsto \tau] \vdash e : \tau'$. Since $\gamma'' \models \Gamma[x \mapsto \tau]$, we can apply the induction hypothesis on γ'' to obtain

$$\begin{aligned}\gamma'' \models \Gamma[x \mapsto \tau] \quad &\Rightarrow \quad \hat{\gamma}''[\![e]\!] \in \mathsf{T}_{\tau'} \\ &\Rightarrow \quad \hat{\gamma}[\![\lambda x : \tau \,.\, e]\!] \in \mathsf{T}_\tau\end{aligned}$$

4 Agreement of semantics

We can also use logical relations to show agreement between denotational and operational semantics. We would like to show that each step operationally the meaning of the expressions is the same:

$$e \to^* v \quad \wedge \quad \vdash e : \tau \Rightarrow \mathcal{C}[\![\vdash e : \tau]\!]\rho_0 = \mathcal{C}[\![\vdash v : \tau]\!]\rho_0$$

We also require the semantics to diverge in exactly the same places:

$$\exists v \,.\, e \to^* v \quad \wedge \quad \vdash e : \tau \iff \mathcal{C}[\![\vdash e : \tau]\!]\rho_0 \neq \bot$$

And we also should agree on base values:

$$e \to^* v \quad \wedge \quad \vdash e : \mathsf{int} \iff \mathcal{C}[\![\vdash e : \mathsf{int}]\!]\rho_0 = v$$

What about the general case (any type)? Does implication hold in both directions? It does not, because there are multiple values v that have the same meaning.

Winskel shows how to do these proofs in the reading (11.4).