

Type Systems for Programming Languages (15-814)

Lecture Notes, Fall 2005, 10/3 and 10/5

Debabrata Dash

October 15, 2006

1 Gödel's T

Gödel's T is defined as a language with the following components:

- **Types** $\tau ::= \text{nat} \mid \tau_1 \rightarrow \tau_2$
- **Expressions** $e ::= x \mid \text{zero} \mid \text{succ}(\tau) \mid \text{rec}(\tau, e, e_0, x.e_1)$

1.1 Specifying a Type

To specify a type we need two forms:

1. Introduction forms — these specify the canonical expressions of the type
2. Elimination forms — these specify how to compute values of the type

These two forms work together by a principle called *inversion principle*, which specifies that the elimination forms and the introduction forms are inverse of each other. While the introduction and elimination forms specify statics of the language, the inversion principle is its dynamics. The inversion principle is also called as *operational semantics*.

1.1.1 Specifying *nat*

Introduction Forms

$$\frac{}{\Gamma \vdash \text{zero} : \text{nat}} \qquad \frac{\Gamma \vdash e : \text{nat}}{\Gamma \vdash \text{succ}(e) : \text{nat}}$$

Elimination Forms

$$\frac{\Gamma \vdash e : \text{nat} \quad \Gamma \vdash e_0 : \tau \quad \Gamma, x : \tau \vdash e_1 : \tau}{\Gamma \vdash \text{rec}(e, e_0, x : \tau.e_1) : \tau}$$

Operational Semantics

$$\frac{}{\text{rec}(\text{zero}, e_0, x.e_1) \mapsto e_0} \qquad \frac{}{\text{rec}(\text{succ}(e), e_0, x.e_1) \mapsto [\text{rec}(e, e_0, x.e_1)/x] e_1}$$
$$\frac{e \mapsto e'}{\text{rec}(e, e_0, x.e_1) \mapsto \text{rec}(e', e_0, x.e_1)}$$

1.1.2 Specifying functions

Introduction Forms

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2 \quad x \# \Gamma}{\Gamma \vdash \lambda x : \tau_1. e : \tau_1 \rightarrow \tau_2}$$

Elimination Forms

$$\frac{\Gamma \vdash e_1 : \tau_2 \rightarrow \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1 e_2 : \tau_1}$$

Operational Semantics Recall the definition of value in *call by name* evaluation as:

$$\overline{\text{zero value}} \qquad \overline{\text{succ}(e) \text{ value}} \qquad \overline{\lambda x : \tau. e \text{ value}}$$

Given this definition of value, the evaluation rules are:

$$\frac{}{(\lambda x : \tau. e)e_2 \mapsto [e_2/x]e} \qquad \frac{e_1 \mapsto e'_1}{e_1 e_2 \mapsto e'_1 e_2}$$

2 Structural Rules

The typing judgement, $x : \tau$ can be defined using the following inductive rules:

$$\text{(Reflexive)} \qquad \frac{}{\Gamma, x : \tau \vdash x : \tau} \tag{1}$$

$$\text{(Substitution, Transitive)} \qquad \frac{\Gamma, x : \tau_2 \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash [e_2/x] e_1 : \tau_1} \tag{2}$$

$$\text{(Weakening)} \qquad \frac{\Gamma \vdash e : \tau}{\Gamma, x : \tau_1 \vdash e : \tau} \tag{3}$$

$$\text{(Exchange)} \qquad \text{Order of hypothesis does not matter} \tag{4}$$

Reflexivity is always true. The others are principles to be proven.

3 Safety Theorem

The basic premise of the theorem is that the static typing rules and the dynamic evaluation rules fit together. In other words

1. The evaluation cannot run into illegal state
2. The evaluation cannot get illegal instruction

The first one is usually called *Preservation* property and the second one is called *Progress*. More formally the safety theorem is defined as:

Safety Theorem.

- **Preservation:** In a well formed language if $e : \tau$ and $e \mapsto e'$ then $e' : \tau$.
- **Progress:** If $e : \tau$ then either

1. e is a value
2. $e \mapsto e'$, i.e. you can make one step after e .

The proof of above theorem depends on three important lemmas.

Substitution Lemma.

$$\frac{\Gamma, x: \tau_2 \vdash e_1: \tau_1 \quad \Gamma \vdash e_2: \tau_2}{\Gamma \vdash [e_2/x]e_1: \tau_1}$$

Inversion Lemma.

We can invert the typing relations. i.e.

$$\begin{aligned} \text{succ}(e): \text{nat} &\models e: \text{nat} \\ \text{rec}(e, e_0, x: \tau. e_1): \tau &\models e: \text{nat}, e_0: \tau, x: \tau \vdash e_1: \tau \\ \lambda x: \tau_1. e: \tau_1 \rightarrow \tau_2 &\models x: \tau_1 \vdash e: \tau_2 \\ e_1 e_2: \tau &\models e_1: \tau_2 \rightarrow \tau, e_2: \tau_2 \end{aligned}$$

Canonical Forms Lemma. If $e: \tau$ and e is a closed value, then

1. if $\tau = \text{nat}$ then $e = \text{zero}$ or $\exists e' \text{ nat s.t. } e = \text{succ}(e')$
2. if $\tau = \tau_2 \rightarrow \tau$ then $\exists e' \text{ s.t. } e = \lambda x: \tau_2. e'$

Proof of Preservation. Below we give two cases for the preservation for functions, cases for natural numbers can be proved in a similar way:

- **Case 1:** $\frac{e_1 \mapsto e'_1}{e_1 e_2 \mapsto e'_1 e_2}$

If $e_1 e_2: \tau$ then we have to show that $e'_1 e_2: \tau$

by inversion lemma, $\exists \tau_2$ s.t. $e_1: \tau_2 \rightarrow \tau, e_2: \tau_2$

by inductive hypothesis $e_1: \tau_2 \rightarrow \tau$ implies $e'_1: \tau_2 \rightarrow \tau$, so using the application rule $e' = e'_1 e_2: \tau$

- **Case 2:** $(\lambda x: \tau. e) e_2 \mapsto [e_2/x]e$ If $e = (\lambda x: \tau. e) e_2: \tau$ we have to show that $e' = [e_2/x]e: \tau$
By inversion lemma, $\exists \tau'_2$ s.t. $\lambda x: \tau_2. e_1: \tau'_2 \rightarrow \tau e_2: \tau'_2$
Again by inversion lemma, $\tau'_2 = \tau_2$
Using substitution lemma, $[e_2/x]e_1: \tau$

□

Proof of Progress. Using the above lemma, let us consider the type typing inductions

1. $\frac{x: \tau_1 \vdash e: \tau_2}{\lambda x: \tau_1. e: \tau_1 \rightarrow \tau_2}$

In this case $\lambda x: \tau_1. e: \tau_1 \rightarrow \tau_2$ is already a value.

2. $\frac{e_1: \tau_2 \rightarrow \tau \quad e_2: \tau_2}{e_1 e_2: \tau}$

By induction hypothesis either e_1 value or $e_1 \mapsto e'_1$ and either e_2 value or $e_2 \mapsto e'_2$

Consider the case when e_1 value then using canonical forms lemma e_1 has to be of form $\lambda x: \tau_2. e'_1$

By using substitution $e_1 e_2 \mapsto [e_2/x]e'_1$

Consider when $e_1 \mapsto e'_1$, then by evaluation rules for functions $e_1 e_2 \mapsto e'_1 e_2$

Similar argument applies for e_2 's cases.

□

4 Termination Property of Gödel's T

Theorem 4.1. if $e : \tau$ then $e \searrow$, i.e. $\exists e'$ s.t. e' value and $e \mapsto^* e'$

Proof. This is a special case of Hereditary Termination where $n = 0$. □

While proving this theorem it is tempting to use structural induction on the typing judgements. However that is not sufficient and we need a much stronger judgement in terms of *Hereditary Termination* property.

Hereditary Termination

Hereditary termination $HT_\tau(e)$, where $e : \tau$ is defined as:

$$\begin{aligned} HT_{nat}(e) \quad & \text{iff} \quad e \mapsto^* \text{zero} \\ & \text{or } e \mapsto^* \text{succ}(e') \text{ s.t. } HT_{nat}(e') \\ HT_{\tau_1 \rightarrow \tau_2} \quad & \text{iff} \quad e \mapsto^* \lambda x : \tau_1. e' \text{ s.t. } \forall e_1 HT_{\tau_1}(e_1) \Rightarrow (HT_{\tau_2}[e_1/x]e') \end{aligned}$$

Theorem 4.2.

$$\frac{x_1 : \tau_1, x_2 : \tau_2, \dots, x_n : \tau_n \vdash e : \tau \quad HT_{\tau_1}(e_1), HT_{\tau_2}(e_2), \dots, HT_{\tau_n}(e_n)}{HT_\tau([e_1/x_1, e_2/x_2, \dots, e_n/x_n]e)}$$

Proof. Proof by induction on typing judgements.

Case 1

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1. e : \tau_1 \rightarrow \tau_2}$$

We have to show that: $HT_{\tau_1 \rightarrow \tau_2}(\lambda x : \tau_1. e)$

In other words, Given $HT_{\tau_1}(e_1)$ we have to show that $HT_{\tau_2}([e_1/x]e)$. This follows directly from the induction hypothesis.

Case 2

$$\frac{\Gamma, e_1 : \tau_2 \rightarrow \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 e_2 : \tau} \quad \text{To Show: } HT_\tau(e_1 e_2)$$

We have the following inductive hypotheses

- $HT_{\tau_2 \rightarrow \tau}(e_1)$
- $HT_{\tau_2}(e_2)$

$$\begin{aligned} e_1 & \mapsto^* \lambda x : \tau_2. e'_1 \text{ By CFL} \\ e_1 e_2 & \mapsto^* (\lambda x : \tau_2. e'_1) e_2 \\ & \mapsto [e_2/x]e'_1 \Rightarrow HT_\tau(e_1 e_2) \end{aligned}$$

$\therefore HT_\tau(e_1 e_2)$. The above proof is valid because of the following lemma

Head Expansion/Inverse Execution. If $HT_\tau(e')$ and $e \mapsto e'$ then $HT_\tau(e)$

Proof. By definition of hereditary termination. □

Case 3

$$\frac{\Gamma \vdash e : \text{nat} \quad \Gamma \vdash e_0 : \tau \quad \Gamma, x : \tau \vdash e_1 : \tau}{\Gamma \vdash \text{rec}(e, e_0, x. e_1) : \tau}$$

To Show: $HT_\tau(\text{rec}(e, e_0, x. e_1))$

Induction Hypotheses:

1. $HT_{\text{nat}}(e')$
2. $HT_\tau(e_0)$
3. $\forall HT_\tau(e') \Rightarrow HT_\tau([e'/x]e_1)$

1. **sub-case 1:** $e \mapsto^* \text{zero}$

$$\text{rec}(e, e_0, x, e_1.) \mapsto^* \text{rec}(\text{zero}, e_0, x, e_1.) \mapsto e_0$$

By induction hypothesis (2) and head expansion lemma $HT_\tau(\text{rec}(e, e_0, x. e_1))$

2. **sub-case 2:** $e \mapsto^* \text{succ}(e')$

$$\begin{aligned} \text{rec}(e, e_0, x, e_1.) &\mapsto^* \text{rec}(\text{succ}(e'), e_0, x, e_1.) \\ &\mapsto [\text{rec}(e', e_0, x. e_1)/x] e_1 \end{aligned}$$

Inductively $HT_\tau(\text{rec}(e', e_0, x. e_1))$, hence by head expansion lemma, $HT_\tau(\text{rec}(e, e_0, x. e_1))$

□