CS611 Lecture 32 **Progress + Preservation $\Rightarrow$ Soundness** 17 November 2003

Scribe: Milos Hasan Lecturer: Radu Rugina

1 Untypable Expressions

Not all expressions in the simply typed lambda calculus can be typed. The expression $\mathit{self} \equiv \lambda x : \tau . x\ x$ is one possible counterexample. To prove that it cannot be typed, we suppose the reverse, i.e. $\Gamma \vdash (\lambda x : \tau . x\ x) : \tau'$, and infer a contradiction. The proof tree for this formula must have the following form:

$$\frac{\dfrac{\Gamma[x \mapsto \tau] \vdash x : \tau'' \to \tau' \quad \Gamma[x \mapsto \tau] \vdash x : \tau''}{\Gamma[x \mapsto \tau] \vdash x\ x : \tau'}}{\Gamma \vdash (\lambda x : \tau . x\ x) : \tau'}$$

But this means that $\tau = \tau'' \to \tau'$ and at the same time $\tau = \tau''$, therefore we have $\tau'' = \tau'' \to \tau'$, which is a contradiction. (In fact, it is possible to type this expression with recursive types, which will be discuss in the following lectures.)

2 Soundness in Typed Lambda Calculus

The main goal of introducing types is to statically eliminate possible run-time errors. In other words, we want to make sure that the evaluation of any well-typed expression will never get stuck. In that case, we say that the type system is sound. We can formally write the soundness property as:

$$(\ \vdash e : \tau\ \wedge\ e \to_1^* e'\ )\ \Rightarrow\ (\ e \in \mathit{Value}\ \vee\ \exists e''\ .\ e \to_1 e''\ )$$

The standard way to prove the soundness of a type system is to break this property down into two simpler properties:

- *Progress*: any well-typed expression is either a value, or can be reduced further;
- *Preservation*: the evaluation of a well-typed expression preserves the type of that expression;

Together, the Progress and Preservation properties imply the Soundness of the type system. However, the reverse implication is not necessarily true. Next, we formally express and prove the Progress and Preservation properties for the simply-typed lambda calculus.

2.1 Progress

Progress means that a well-typed expression is either a value or can be reduced in one step:

$$\vdash e : \tau\ \Rightarrow\ (\ e \in \mathit{Value} \vee \exists e'\ .\ e \to_1 e'\ )$$

Here by $\to_1$ we mean CBN evaluation, and by $\vdash e : \tau$ we mean $\emptyset \vdash e : \tau$. Recall that the sets of primitives, expressions and values are defined as:

$$\begin{array}{rcl} e & ::= & x\ \mid\ \lambda x : \tau . e\ \mid\ e_0\ e_1\ \mid\ p \\ p & ::= & n\ \mid\ \mathit{true}\ \mid\ \mathit{false}\ \mid\ \mathit{unit} \\ v & ::= & p\ \mid\ \lambda x : \tau . e \end{array}$$

We will prove the progress property by structural induction on e. So, suppose that $\vdash e : \tau$ and also that $e \notin \mathit{Value}$. We need to prove that there is an e' such that $e \to_1 e'$. Since $e \notin \mathit{Value}$, we have only two possibilities:

- $e = x$: We have $\emptyset \vdash x : \tau$. But this is not possible, since a variable x can only be typed in an environment Γ which contains x:

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}$$

  Here $\Gamma = \emptyset$, so $\vdash x : \tau$ doesn't hold. Therefore the implication is trivially true.

- $e = e_0\ e_1$: Recall that the rule for application is:

$$\frac{\Gamma \vdash e_0 : \tau' \to \tau \quad \Gamma \vdash e_1 : \tau'}{\Gamma \vdash e_0\ e_1 : \tau}$$

  So the fact that $\vdash e_0\ e_1 : \tau$ implies $\vdash e_0 : \tau' \to \tau$ and $\vdash e_1 : \tau'$.

  Now we consider two cases: either $e_0 \in \mathsf{Value}$ or not.

  - Case 1: If e_0 is a value, then it must be an abstraction, so

$$e = e_0\ e_1 = (\lambda x : \tau'.e_2)\ e_1 \to_1 e_2[x \mapsto e_1]$$

    and we set $e' = e_2[x \mapsto e_1]$.
  - Case 2: If e_0 is not a value, then by induction hypothesis there must be an e_0' such that $e_0 \to_1 e_0'$. (The IH applies here, because e_0 is a subexpression of e). Therefore $e_0\ e_1 \to_1 e_0'\ e_1$ and we set our e' to be $e_0'\ e_1$.

2.2 Preservation

This property simply states that computation should preserve type:

$$(\ \vdash e : \tau \ \wedge \ e \to_1 e'\ ) \ \Rightarrow \vdash e' : \tau$$

We prove the preservation property by structural induction on e. We have several cases:

- $e = x$: We have $\vdash x : \tau$, which is actually $\emptyset \vdash x : \tau$. As shown above, this cannot hold and the implication is trivially true in this case.

- $e \in \mathit{Value}$: A value cannot be reduced, so $e \to_1 e'$ doesn't hold and the implication is trivially true.

- $e = e_0\ e_1$: We have $\vdash e_0\ e_1 : \tau$ and because there is just one typing rule for application, we get $\vdash e_0 : \tau' \to \tau$ and $\vdash e_1 : \tau'$ for some τ'. We also know that $e_0\ e_1 \to_1 e'$ – we have two ways to get this assertion:

  - By the rule

$$\frac{e_0 \to_1 e_0'}{e_0\ e_1 \to_1 e_0'\ e_1}$$

    where $e' = e_0'\ e_1$. Now, by induction hypothesis for $\vdash e_0 : \tau' \to \tau$, we get $\vdash e_0' : \tau' \to \tau$. Now we just create this proof tree

$$\frac{\vdash e_0' : \tau' \to \tau \quad \vdash e_1 : \tau'}{\vdash e_0'\ e_1 : \tau}$$

    and because $e' = e_0'\ e_1$, we get $\vdash e' : \tau$, which is what we wanted to prove.
  - By the rule

$$\frac{}{(\lambda x : \tau''.e_2)\ e_1 \to_1 e_2[x \mapsto e_1]}$$

    and by setting $e_0 = \lambda x : \tau''.e_2$ and $e' = e_2[x \mapsto e_1]$. Recall that $\vdash e_0 : \tau' \to \tau$, so $\vdash (\lambda x : \tau'').e_2 : \tau' \to \tau$. The (only) rule for abstractions is

$$\frac{\Gamma[x \mapsto \tau] \vdash e : \tau'}{\Gamma \vdash (\lambda x : \tau.e) : \tau \to \tau'}$$

so we get $\tau'' = \tau'$ and

$$\emptyset[x \mapsto \tau'] \vdash e_2 : \tau$$

But we want to prove a slightly different thing, $\vdash e' : \tau$, which is equivalent to:

$$\emptyset \vdash e_2[x \mapsto e_1] : \tau$$

This statement follows from the previous one by using the *substitution lemma*, which is given below.

2.3 Substitution Lemma

This lemma says that for any expression e_2 with a free variable x typed as τ', we can substitute an expression e_1 (with type τ') for x and the type of the whole expression will be preserved:

$$(\ \Gamma[x \mapsto \tau'] \vdash e_2 : \tau \ \wedge\ \Gamma \vdash e_1 : \tau'\ ) \ \Rightarrow\ \Gamma \vdash e_2[x \mapsto e_1] : \tau$$

We leave the proof as an exercise.

3 Expressing Soundness with Denotational Semantics

We would like to express the soundness property using denotational semantics. For this, we first need to write a function $\mathcal{T}[\![\tau]\!]$ that takes a type and returns the domain of values that τ corresponds to:

$$\begin{array}{rcl} \mathcal{T}[\![\mathsf{int}]\!] & = & \mathbb{Z} \\ \mathcal{T}[\![\mathsf{bool}]\!] & = & \mathbb{T} \\ \mathcal{T}[\![\mathsf{unit}]\!] & = & \mathbb{U} \\ \mathcal{T}[\![\tau_1 \to \tau_2]\!] & = & \mathcal{T}[\![\tau_1]\!] \to \mathcal{T}[\![\tau_2]\!] \end{array}$$

With this definition, the desired property is that the evaluation of a well-typed expression e of type τ, must a value in the domain $\mathcal{T}[\![\tau]\!]$:

$$(\ \Gamma \vdash e : \tau \wedge \rho \models \Gamma\ ) \ \Rightarrow\ \mathcal{C}[\![e]\!]\rho \in \mathcal{T}[\![\tau]\!]$$

where the satisfiability relation $\rho \models \Gamma$ requires that the values in the variable environment ρ must agree with the types in the type environment Γ:

$$\rho \models \Gamma \ \text{ iff }\ dom(\rho) = dom(\Gamma)\ \wedge\ \forall x \in dom(\Gamma).\rho(x) \in \mathcal{T}[\![\Gamma(x)]\!]$$