CS611 Lecture 28:
Soundness Of Typing Rules and
an Introduction To Denotational Semantics of TLC 11/03/99

by Ranveer Chandra and Dongmin Liang

1 Soundness Of Typing Rules

1.1 Overview

In the previous lecture we had introduced two properties of Typing Rules that we thought would help us in proving their **Soundness**. In this section we first prove the two properties and then use them to prove the Soundness of Typing Rules.

1.2 Preservation Property

This property states that :

$$\forall e, e'. \ (\vdash e : \tau \quad \wedge \quad e \rightarrow e') \quad \Rightarrow \ \vdash e' : \tau$$

Proof: We Prove this property by induction on the height of the derivation tree. For the base case, the argument is trivial, that is when e is a variable x or a number n. In both the cases the property is vacuously true. In the induction step, if e goes to e', then it is due to the following rules :

Rule 1 : $$\frac{e_1 \rightarrow e_1'}{(e_1 e_2) \rightarrow (e_1' e_2)}$$

Rule 2 : $$\frac{}{((\lambda x : \tau . e_1) e_2) \rightarrow e_1[e_2/x]}$$

Now if e goes to e' using Rule 1, then we need to prove that : $(\vdash (e_1 e_2) : \tau \wedge e_1 \rightarrow e_1') \Rightarrow \vdash (e_1' e_2) : \tau$. And from the Typing Rules we know that :

$$\frac{\vdash e_1 : \tau_2 \rightarrow \tau \quad \vdash e_2 : \tau_2}{\vdash (e_1 e_2) : \tau} \tag{1}$$

Since $e_1 : \tau_2 \rightarrow \tau$ from (1) and $e_1 \rightarrow e_1'$ from Rule 1, therefore applying these two conditions to the Induction Hypothesis, we get :

$$\vdash e_1 : \tau_2 \rightarrow \tau \wedge e_1 \rightarrow e_1' \Rightarrow \vdash e_1' : \tau_2 \rightarrow \tau \tag{2}$$

So we now know that the type of e_1' is $\tau_2 \rightarrow \tau$ from (2) and the type of e_2 is τ_2, from (1). Then applying the Typing Rules we get,

$$\frac{\vdash e_1' : \tau_2 \rightarrow \tau \quad \vdash e_2 : \tau_2}{\vdash (e_1' e_2) : \tau} \tag{3}$$

So we have proved if e goes to e' using Rule 1 then the preservation property holds.

In the other case when the transition is using Rule 2 above, then the corresponding Typing Rule is :

$$\frac{\dfrac{\{x:\tau_2\} \vdash e_1 : \tau}{\vdash (\lambda x:\tau_2.e_1):\tau_2 \rightarrow \tau} \quad \vdash e_2 : \tau_2}{\vdash ((\lambda x:\tau_2.e_1)e_2):\tau} \tag{4}$$

So we know from (4) that for the Lambda expression to have type τ , $\{x : \tau_2\} \vdash e_1 : \tau$ and $\vdash e_2 : \tau_2$ should hold. Now if this holds then from the substitution lemma $\vdash e_1[e_2/x] : \tau$ is true. Therefore the Preservation property also holds for this case.

1.3 The Progress Property

This property states that :

$$\vdash e : \tau \Rightarrow (\,Value(e) \lor \exists e'.e \rightarrow e')$$

Proof : Here we use structural induction to prove this property. In the base case when e is a Lambda Expression or a variable or a number, then it is a value and hence the property holds true trivially. In the induction step, when e is an application of the form (e_1e_2), then (1) holds. Now when e_1 is not a value, then applying the induction hypothesis on e_1, we get that $\exists e_1'.e_1 \rightarrow e_1'$. When e_1 is a value, then e_1 is definitely a lambda expression and so the reduction given by Rule 2 in the previous proof could be applied. So in either case there exists another expression to which this application could be reduced. Hence the Progress Property is proved by induction.

1.4 Soundness Of The Typing Rules

Another way to say that the set of Typing Rules is sound is by proving :

$$\forall e, e'.\vdash e : \tau \land e \rightarrow^* e' \Rightarrow \vdash e' : \tau \land (\,Value(e') \lor \exists e''.e' \rightarrow e'')$$

Proof : We prove the above by Induction on the number of steps. In the base case, when there are zero steps, we need to prove that :

$$\forall e.\vdash e : \tau \land e \rightarrow^0 e \Rightarrow \vdash e : \tau \land (\,Value(e) \lor \exists e''.e \rightarrow e'')$$

this is obviously true because of the Progress Property of Typing Rules. Now in the Induction Step, we assume for n steps:

$$\forall e.e'' \vdash e : \tau \land e \rightarrow^n e'' \Rightarrow \vdash e'' : \tau \land (\,Value(e'') \lor \exists e'''.e'' \rightarrow e''') \tag{5}$$

So for (n+1) steps, i.e. $e \rightarrow^{(n+1)} e'$, there definitely exists an e'' such that $e \rightarrow^n e''$ and $e'' \rightarrow e'$. From (5) and the Preservation Property we can conclude $e' : \tau$. And from the Progress Property we can conclude that either e' is a value or there exists an e'' such that e' goes to e'' in one step. Hence by Induction we have proved that the set of Type Rules is **SOUND**.

2 Introduction to Denotational Semantics of TLC

2.1 Denotational View

$\Gamma \vdash e : \tau$ is just another way of saying that e in the context Γ evaluates to a value of type τ. By our knowledge of Denotational Semantics we could say that $\mathcal{C}[\![e]\!]\rho$ evaluates to a value of type τ. For the purposes of translation of τ to our required domain, we define $\mathcal{T}[\![\tau]\!]$ as below :
$\mathcal{C}[\![e]\!]\rho \in \mathcal{T}[\![\tau]\!] \quad (V_\tau^e \,:\, Winskel\,notation)$
where ρ is value environment and Γ is type environment.

If ρ is consistent with Γ then :

$\forall x \in dom(\rho).\rho(x) \in \mathcal{T}[\![\Gamma(x)]\!]$
and so we can write our Denotational Semantics as :
$\mathcal{C}[\![\Gamma \vdash e : \tau]\!]\rho \in \mathcal{T}[\![\tau]\!]_\perp$

2.2 Definition of $\mathcal{T}$

$\mathcal{T}[\![int]\!] = \mathcal{Z}$ (type int translates to the Domain Of Integers)
$\mathcal{T}[\![\tau_0 \rightarrow \tau_1]\!] = \mathcal{T}[\![\tau_0]\!] \rightarrow \mathcal{T}[\![\tau_1]\!]_\perp$
It should be noted that we only need to lift τ_1 if the language eager. In the other case we will need to lift both τ_0 and τ_1. For Simply Typed Lambda Calculus we do not need to lift either domain, since all expressions are guaranteed to terminate. This is proved in the next lecture.

2.3 The New Denotational Semantics

So we can define the Denotational Semantics for the Types Lambda Calculus as :

$\mathcal{C}[\![\Gamma \vdash x : \tau]\!]\rho = \rho(x) \quad$ where $\rho(x) \in \mathcal{T}[\![\tau]\!]$

$\mathcal{C}[\![\Gamma \vdash e_1 e_2 : \tau]\!]\rho = \mathcal{C}[\![\Gamma \vdash e_1 : \tau_2 \rightarrow \tau_1]\!]\rho(\mathcal{C}[\![\Gamma \vdash e_2 : \tau_2]\!]\rho)$

$\mathcal{C}[\![\Gamma \vdash (\lambda x : \tau.e) : \tau \rightarrow \tau_2]\!]\rho = \lambda v \in \mathcal{T}[\![\tau]\!].\, \mathcal{C}[\![\Gamma[x \mapsto \tau] \vdash e : \tau_2]\!]\rho[x \mapsto v]$