COMP 617 RAP Seminar, Fall 2006

Presentor: Walid Taha
Scribe: Gregory Malecha
09/27/2006

This lecture was an introduction to type systems focusing on what it means for a language to be type-safe.

1 Type Safety

A typed language is defined as a tuple $(\mathbb{S}, \mapsto: \mathbb{S} \rightharpoonup \mathbb{S}, \vdash: \mathbb{S} \mapsto \mathit{bool}, \mathbb{V} : \mathit{set})$ where:

- $\mathbb{S}$ is the set of terms (defined by syntax),
- $\mapsto$ is the reduction semantics to use, generally small-step semantics,
- $\vdash$ is the typing predicate, and
- $\mathbb{V}$ is the set of values

Type safty is a property which relates the semantics of a language to the type system. We want to be guaranteed two things when executing, preservation and progress.

- **Progress** says that a well typed term is either a value or can be reduced by one of the rules given in the operational semantics.
- **Preservation** says that reduction on a well typed term yields a well-typed term.

Based on the above definition, we define a *type safe* language to be one in which

$$\forall e, t. \cdot \vdash e : t \rightarrow e \notin \mathbb{V} \rightarrow \exists e'. e \mapsto e' \wedge \cdot \vdash e' : t$$

It is important to notice that the type t is preserved completely. This property must hold even in type-systems which allow subtyping, i.e. it is not sufficient for initial t to be a super type of the final t.

The multiple implies operations in the above statement make it difficult to read. The following form is considerably easier to read.

$$\forall e, t. (\cdot \vdash e : t \wedge e \notin \mathbb{V}) \rightarrow \exists e'. e \mapsto e' \wedge \cdot \vdash e' : t$$

Two ways to verify the equivalence of the statements are to use the equivalence $A \rightarrow B \equiv \neg A \vee B$, or to simply enumerate all possibilities of each predicate using a truth table. In general, $(A \rightarrow B) \rightarrow C \equiv (A \wedge B) \rightarrow C$.

This result has an interesting result based on the Curry-Howard isomorphism which relates type systems to proofs. Using the Curry-Howard isomorphism, we can relate the equality to:

$$A \rightarrow B \rightarrow C \Longrightarrow (A \times B) \rightarrow C$$

Which we refer to as Currying. In general, the idea of decomposing a function of multiple arguements into a series of functions which return functions. In implementation this would look like:

$$\lambda \text{ x y z. t} \equiv \lambda \text{ x. } \lambda \text{ y. } \lambda \text{ z. t}$$

2 Typing Rules in $\lambda_{\rightarrow}$

The typing rules for $\lambda_{\rightarrow}$ are embodied in the following rules:

$$\frac{\Gamma(x) = t}{\Gamma \vdash x : t}\ \text{T-Var} \qquad \frac{\Gamma, x : t \vdash e : t_2}{\Gamma \vdash \lambda x.e : t_1 \rightarrow t_2}\ \text{T-Abs}$$

$$\frac{\Gamma \vdash e_2 : t_1 \qquad \Gamma \vdash e_1 : t_1 \rightarrow t_2}{\Gamma \vdash e_1\, e_2 : t_2}\ \text{T-App}$$

3 Proving Type Safty for $\lambda_{\rightarrow}$

In order to prove type safty, we start by proving a lemma called the substitution lemma. The lemma states:

$$x : t_1 \vdash e_1 : t \wedge \cdot \vdash e_2 : t_1 \rightarrow \cdot \vdash e_1[x = e_2] : t$$

This can be proven via straightforward induction on typing judgements.

Based on the above lemma, we can prove the type safty of $\lambda_{\rightarrow}$ using structural induction on terms. To do this we consider the types of terms in $\mathbb{S}$.

1. For the case $e = x$, the statement is vacuously true since the type system does not assign a type to free variables.

2. For the case $e = \lambda x.e_1$, the statement is again vacuously true since $e \in \mathbb{V}$.

3. For the case $e = e_1\, e_2$, the one of two semantic rules apply depending on whether $e_1 \in \mathbb{V}$. Therefore, we perform case analysis.

   (a) $e = e_1\, e_2$ where $e_1 \notin \mathbb{V}$. We know that if $e_1\, e_2$ is typeable then e_1 and e_2 are independently typeable with types $t_1 \rightarrow t$ and t_1 respectively. Since e_1 is typeable and has size smaller than the size of $e_1\, e_2$, by the inductive hypothesis we know that there exists some e' such that if $e_1 \mapsto e_2$ and $\cdot \vdash e' : t_1 \rightarrow t$. Applying the substitution lemma here yields that $e_1\, e_2$ is typeable with type t which is what we wanted to show.

   (b) $e = \lambda x.e_3\, e_2$. We know that if $\lambda x.e_3\, e_2$ is typeable then λ x. e_3 and e_2 are independently typeable with types $t_1 \rightarrow t$ and t_1 respectively. Furthermore, based on T-Abs, we can conclude that $x : t_1 \vdash e_3 : t$. By β-reduction we know that $e' = e_3[x = e_2]$. Applying the substitution lemma yields that this is typeable with type t.

Based on the properties of structural induction, this holds for all $s \in \mathbb{S}$.