Lwnn Typing Rules

CS 260, Fall 2013

1 Typing Rules

1.1 Helpers

We use a ClassTable object to embody the necessary global set of classes. Our ClassTable is adapted from the one used in the FetherweightJava paper and is a map from class names to class declarations. Providing a mechanism for looking up field and method types for a given class. A program then is a pair (CT, e), we also make the assumption that the ClassTable is fixed.

$t \in ClassTable = ClassName \to ((Variable \to Type) \times (MethodName \to Type))$

We assume that the ClassTable is fixed for a given run of the interpreter.

$inittypechecker : Program \to (ClassTable \times Program)$
$inittypecheckerp = foldl(\lambda acc.\lambda class.\ acc \cup [\pi_c n \mapsto (fields, methods)], [], p)$ where
$fields = \lambda x.\ \pi_f(class)(x)$
$methods = \lambda x.\ \pi_m(class)(x)$

$field : ClassName \times Variable \to Type$
$field\ cn\ x = \pi_1(t(cn))(x)$

$method : ClassName \times Variable \to Type$
$method\ mn\ x = \pi_2(t(cn))(x)$

1.2 Subtyping

$$cn \sqsubseteq cn \qquad \text{(Reflexivity)}$$

$$\frac{cn_c \sqsubseteq cn_b \qquad cn_b \sqsubseteq cn_a}{cn_c \sqsubseteq cn_a} \qquad \text{(Transitivity)}$$

$$\frac{\textsf{classs } cn_1 \textsf{ extends } cn_2 \ \{...\}}{cn_1 \sqsubseteq cn_2} \qquad \text{(Definition)}$$

1.3 Classes

$$\frac{\textsf{class } cn_1 \textsf{ extends } cn_2 \ \{ \textsf{ fields } \overrightarrow{x : \tau} \ \cdot \ \textsf{methods } \vec{m} \ \} \in \vec{class} \qquad m \in \vec{m} \qquad \Gamma \vdash m : \tau}{\Gamma \vdash \vec{class} : \texttt{null}}$$

(T-Class)

1.4 Methods

$$\frac{\Gamma' = \Gamma[\overrightarrow{x : \tau}] \qquad \Gamma' \vdash \overrightarrow{s : \tau_s} \qquad \Gamma' \vdash e : \tau_r}{\Gamma \vdash \textsf{def } mn(\overrightarrow{x : \tau}) : \tau_r = \{\vec{s} \cdot \textsf{return } e\} : \texttt{null}} \qquad \text{(T-Method)}$$

1.5 Statements

$$\frac{\Gamma \vdash x : \tau_1 \qquad \Gamma \vdash e : \tau_2 \qquad \tau_2 \sqsubseteq \tau_1}{\Gamma \vdash x := e : \texttt{null}} \qquad \text{(T-Assign)}$$

$$\frac{\Gamma \vdash e_1 : cn \qquad field(cn, x) = \tau_f \qquad \Gamma \vdash e_2 : \tau_v \qquad \tau_v \sqsubseteq \tau_f}{\Gamma \vdash e_1.x := e_2 : \texttt{null}} \text{ (T-Update)}$$

$$\frac{\Gamma \vdash x : \tau_x \qquad \Gamma \vdash e : cn \qquad method(cn, mn) = \vec{\tau}\,' \to \tau_r \qquad \Gamma \vdash \overrightarrow{e_i : \tau_i} \qquad \overrightarrow{\tau_i' \sqsubseteq \tau_i} \qquad \tau_r \sqsubseteq \tau_x}{\Gamma \vdash x := e.mn(\vec{e}) : \texttt{null}}$$

(T-Method-Invocation)

$$\frac{\Gamma \vdash x : \tau_x \qquad \begin{array}{l}\Gamma \vdash e : cn \qquad method(cn, cn) = \vec{\tau}\,' \to \tau_r \qquad \Gamma \vdash \overrightarrow{e_i : \tau_i} \\ \overrightarrow{\tau_i' \sqsubseteq \tau_i} \qquad \Gamma \vdash \tau_r \sqsubseteq \tau_x \qquad \Gamma \vdash \textbf{new } C(\vec{e}) : \tau_r\end{array}}{\Gamma \vdash x := \textsf{new } cn(\vec{e}) : \texttt{null}}$$

(T-New)

$$\frac{\Gamma \vdash e : \texttt{bool} \qquad \Gamma \vdash \overrightarrow{s_i : \tau_i} \qquad \Gamma \vdash \overrightarrow{s_j : \tau_j}}{\Gamma \vdash \textbf{if } (e) \ \vec{s_1} \textbf{ else } \vec{s_2} : \texttt{null}} \qquad \text{(T-If)}$$

$$\frac{\Gamma \vdash e : \texttt{bool} \qquad \Gamma \vdash \overrightarrow{s_i : \tau_i}}{\Gamma \vdash \textbf{while } (e) \ \vec{s} : \texttt{null}} \qquad \text{(T-While)}$$

1.6 Expressions

$$\Gamma \vdash i : \texttt{int} \qquad \text{(T-Int)}$$

$$\Gamma \vdash str : \texttt{string} \qquad \text{(T-String)}$$

$$\Gamma \vdash \textsf{true} : \texttt{bool} \qquad \text{(T-True)}$$

$$\Gamma \vdash \textbf{false} : \texttt{bool} \qquad \text{(T-False)}$$

$$\Gamma \vdash \textbf{null} : \texttt{null} \qquad \text{(T-Nulls)}$$

$$\Gamma \vdash x : \Gamma(x) \qquad \text{(T-Var)}$$

$$\frac{\Gamma \vdash e : cn \qquad field(cn, f) = \tau}{\Gamma \vdash e.f : \tau} \qquad \text{(T-Access)}$$

$$\frac{\oplus \in \{+, -, *, \div\} \qquad \Gamma \vdash e_1 : \texttt{int} \qquad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 \oplus e_2 : \texttt{int}} \qquad \text{(T-NumOps)}$$

$$\frac{\oplus \in \{<, \leq\} \qquad \Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau \qquad \tau \in \{\texttt{string}, \texttt{int}\}}{\Gamma \vdash e_1 \oplus e_2 : \texttt{bool}} \qquad \text{(T-Comparison)}$$

$$\frac{\oplus \in \{\wedge, \vee\} \qquad \Gamma \vdash e_1 : \texttt{bool} \qquad \Gamma \vdash e_2 : \texttt{bool}}{\Gamma \vdash e_1 \oplus e_2 : \texttt{bool}} \qquad \text{(T-BoolOps)}$$

$$\frac{\oplus \in \{=, \neq\} \qquad \Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1 \oplus e_2 : \texttt{bool}} \qquad \text{(T-EqOps)}$$