1 Introduction

The purpose of this assignment is demonstrate some technical issues for type checking more practical, Java-like languages.

2 Type Checking Method Invocations

Consider proving the Subject Reduction (type preservation) theorem for Featherweight Java; that is, theorem 2.4.1 on p. 11. Specifically, proving this theorem for the case when the rules `R-INVK` and `T-INVK` were applied, repeated here for quick lookup:

$$\frac{mbody(m, C) = \overline{\texttt{x}}.e_0}{\textbf{new}\ C\texttt{(}\overline{\texttt{e}}\texttt{)}.m(\overline{\texttt{d}}) \mapsto [\overline{\texttt{d}}/\overline{\texttt{x}},\ \textbf{new}\ C\texttt{(}\overline{\texttt{e}}\texttt{)}/\texttt{this}]e_0}\ \texttt{R-INVK}$$

$$\frac{\Gamma \vdash e_0\ :\ C_0 \quad mtype(m, C_0) = \overline{\texttt{D}} \to C \quad \Gamma \vdash \overline{\texttt{e}}\ :\ \overline{\texttt{C}} \quad \overline{\texttt{C}} <: \overline{\texttt{D}}}{\Gamma \vdash e_0.m(\overline{\texttt{e}})\ :\ C}\ \texttt{T-INVK}$$

The `R-INVK` (beta) reduction rule describes how a method invocation expression is evaluated. The `T-INVK` typing rule describes how a method invocation expression is typed. Notice that the body of method m was *not* type checked in the `T-INVK` rule. Why doesn't the body need to be checked to determine the type of a method invocation of m? The `T-INVK` rule says that it was enough to just look up the type signature of method m to see that invoking it returns an expression of type C.

Again, consider proving type preservation for Featherweight Java for the case when a reduction step was performed because rule `R-INVK` was applied. Given that $\Gamma \vdash \textbf{new}\ C\texttt{(}\overline{\texttt{e}}\texttt{)}.m(\overline{\texttt{d}})\ :\ C_1$, to prove type preservation, one needs to show that $\Gamma \vdash [\overline{\texttt{d}}/\overline{\texttt{x}},\ \textbf{new}\ C\texttt{(}\overline{\texttt{e}}\texttt{)}/\texttt{this}]e_0\ :\ C_1$ also holds. The only typing rule that could assign expression $\textbf{new}\ C\texttt{(}\overline{\texttt{e}}\texttt{)}.m(\overline{\texttt{d}})$ a type is rule `T-INVK` because no other typing rule has an expression in the conclusion that can unify to this expression. The unifier for the `R-INVK` and `T-INVK` rules:

1. e_0 substituted with $\textbf{new}\ C\texttt{(}\overline{\texttt{e}}\texttt{)}$.
2. $\overline{\texttt{e}}$ substituted with $\overline{\texttt{d}}$; so $e_i = d_i, \forall i$.

After applying the unifier to the `I-INVK` rule we have the following:

$$\frac{\Gamma \vdash \textbf{new}\ C\texttt{(}\overline{\texttt{e}}\texttt{)}\ :\ C_0 \quad mtype(m, C_0) = \overline{\texttt{D}} \to C_1 \quad \Gamma \vdash \overline{\texttt{d}}\ :\ \overline{\texttt{C}} \quad \overline{\texttt{C}} <: \overline{\texttt{D}}}{\Gamma \vdash \textbf{new}\ C\texttt{(}\overline{\texttt{e}}\texttt{)}.m(\overline{\texttt{d}})\ :\ C_1}\ \texttt{T-INVK}$$

Also, only rule `T-NEW` could have assigned a type to $\textbf{new}\ C\texttt{(}\overline{\texttt{e}}\texttt{)}$, so by inversion, $\Gamma \vdash$

new $C(\overline{\texttt{e}})$: C_0 implies $C = C_0$. Substituting C for C_0 in `T-INVK` rule:

$$\frac{\Gamma \vdash \textbf{new}\ C(\overline{\texttt{e}})\ :\ C \quad \mathit{mtype}(m, C) = \overline{\texttt{D}} \to C_1 \quad \Gamma \vdash \overline{\texttt{d}}\ :\ \overline{\texttt{C}} \quad \overline{\texttt{C}} <: \overline{\texttt{D}}}{\Gamma \vdash \textbf{new}\ C(\overline{\texttt{e}}).m(\overline{\texttt{d}})\ :\ C_1}\ \texttt{T-INVK}$$

So now we can assume, $\mathit{mbody}(m, C) = \overline{\texttt{x}}.e_0$ by the `R-INVK` rule and $\mathit{mtype}(m, C) = \overline{\texttt{D}} \to C_1$ by the `T-INVK` rule. From just the premises of the `R-INVK` and `T-INVK` rules, it is not clear how to show $\Gamma \vdash [\overline{\texttt{d}}/\overline{\texttt{x}},\ \textbf{new}\ C(\overline{\texttt{e}})/\texttt{this}]e_0\ :\ C_1$ holds because the type of body e_0 is not mentioned in the premises.

2.1 Question

Clearly, there needs to be a lemma that connects the type of a method body with the type signature to prove the type preservation theorem for this case. Which lemma in the paper establishes this connection?

2.2 Hint

The proof of this lemma assumes the `T-METHOD` rule on page 9 holds, which says m is well-formed method. Moreover, there is an implicit assumption that if $\mathit{mtype}(m, C)$ is defined, then C is a well-formed class (the `T-CLASS` judgment holds) and that m is well-formed method in C (the `T-METHOD` holds). This allows typing a method invocation with just the method signature and enables not having to type check the body of the method for every invocation.