Featherweight JPred

Christopher Frost and Todd Millstein
Computer Science Department
University of California, Los Angeles
{frost,todd}@cs.ucla.edu

Technical Report CSD-TR-050038
October 2005

This document defines an extension to Featherweight Java (FJ) [4] that formalizes our approach to modularly typesafe interface dispatch in JPred [5]. To this end, we augment FJ to include interfaces, and we allow methods to have JPred-style `when` clauses. To make it easier to find all the methods in the same "method family," the syntax groups all methods of the same name in each class as a single declaration. The order of each method implementation is still irrelevant.

1 Syntax

$$
\begin{array}{rcl}
\texttt{TD} & ::= & \texttt{class C extends C implements } \overline{\texttt{I}}\ \{\overline{\texttt{T}}\ \overline{\texttt{f}}\texttt{; K } \overline{\texttt{M}}\} \\
 & | & \texttt{interface I extends } \overline{\texttt{I}}\ \{\overline{\texttt{MH}}\} \\
\texttt{K} & ::= & \texttt{C(}\overline{\texttt{T}}\ \overline{\texttt{f}}\texttt{) \{super(}\overline{\texttt{f}}\texttt{); this.}\overline{\texttt{f}}\texttt{ = }\overline{\texttt{f}}\texttt{;\}} \\
\texttt{M} & ::= & \texttt{T m(}\overline{\texttt{T}}\ \overline{\texttt{x}}\texttt{) when } \overline{\texttt{P}}\texttt{ \{return } \overline{\texttt{t}}\texttt{;\}} \\
\texttt{MH} & ::= & \texttt{T m(}\overline{\texttt{T}}\ \overline{\texttt{x}}\texttt{);} \\
\texttt{P,Q,R} & ::= & \texttt{true} \mid \texttt{x@T} \mid \neg\texttt{P} \mid \texttt{P}\wedge\texttt{P} \mid \texttt{P}\vee\texttt{P} \\
\texttt{S,T,U} & ::= & \texttt{C} \mid \texttt{I} \\
\texttt{t} & ::= & \texttt{x} \mid \texttt{t.f} \mid \texttt{t.m(}\overline{\texttt{t}}\texttt{)} \mid \texttt{new C(}\overline{\texttt{t}}\texttt{)} \mid \texttt{(T) t} \\
\texttt{v} & ::= & \texttt{new C(}\overline{\texttt{v}}\texttt{)}
\end{array}
$$

Figure 1: The syntax of FJPred.

Figure 1 gives the syntax of FJPred. It augments FJ to have interfaces and to support method predicates. For uniformity, all methods have a predicate; a method with the predicate `true` has the same semantics as a regular Java method. Predicates include type tests on formals and conjunctions, disjunctions, and negations of such tests. JPred supports several other kinds of predicates, including type tests on fields and linear arithmetic, but these constructs do not interact in interesting ways with interface dispatch, which is the focus of our formalization.

Technically, to be a legal Java program any class that implements some interface must make the implementations of the associated methods `public`. We assume that all methods are implicitly declared `public` in FJPred.

We have analogous notational conventions and sanity conditions to those in FJ, namely:

- The metavariables `C`, `D`, and `E` range over class names, `I` and `J` over interface names, `f` and `g` over field names, `m` and `n` method names, `x` and `y` over parameter names, `s` and `t` over term names, `u` and `v` over value names.

- We assume that `this` is a special variable name that is never used as the name of an argument to a method.
- We use the $\overline{\texttt{D}}$ shorthand as a sequence of elements from domain `D`, using the appropriate separator for each domain. We sometimes treat a sequence as a set. The empty sequence is denoted •.

  We also extend this shorthand as in Featherweight Java to larger syntactic fragments (e.g., $\overline{\texttt{T}}$ $\overline{\texttt{f}}$). A new shorthand of this form is $\texttt{T m(}\overline{\texttt{T}}\ \overline{\texttt{x}}\texttt{) when } \overline{\texttt{P}}\ \{\texttt{return } \overline{\texttt{t}}\texttt{;}\}$, which abbreviates the following method declaration:

```
T m(T_1 x_1, ··· T_n x_n)
   when P_1 {return t_1;}
   ...
   when P_m {return t_m;}
```

- We assume that a class's fields have distinct names from one another and from inherited fields. We also assume that the methods declared in a class or interface have distinct names from one another (but not necessarily from inherited methods). Finally, we assume that a method's formal names are distinct.
- A type table TT is a mapping from type (class or interface) names to their declarations. We sometimes consider a type table as a sequence of pairs, denoted $(\overline{\texttt{T}},\overline{\texttt{TD}})$. A program is a pair of a type table and a term. The rules assume a fixed global type table TT; it is sometimes shadowed by an explicit type table in the context of a few judgments.
- Every type table TT is assumed to satisfy a few conditions. First, $TT(\texttt{C}) = \texttt{class C} \ldots$ for every $\texttt{C} \in \text{dom}(TT)$ and $TT(\texttt{I}) = \texttt{interface I} \ldots$ for every $\texttt{I} \in \text{dom}(TT)$. Second, $\texttt{Object} \notin \text{dom}(TT)$. Third, every type name except `Object` that appears anywhere in TT is in $\text{dom}(TT)$. Finally, there are no nontrivial cycles in the subtyping relation induced by TT; this relation is defined next.

2 Subtyping

The subtyping relation among types is standard and is defined by the following rules:

$\boxed{TT \vdash \texttt{T}_1\texttt{<:}\texttt{T}_2}$

$$TT \vdash \texttt{T<:T} \qquad \text{(S-Ref)}$$

$$\frac{TT \vdash \texttt{T}_1\texttt{<:}\texttt{T}_3 \qquad TT \vdash \texttt{T}_3\texttt{<:}\texttt{T}_2}{TT \vdash \texttt{T}_1\texttt{<:}\texttt{T}_2} \qquad \text{(S-Trans)}$$

$$\frac{TT(\texttt{C}) = \texttt{class C extends D} \ldots}{TT \vdash \texttt{C<:D}} \qquad \text{(S-CExt)}$$

$$\frac{TT(\texttt{C}) = \texttt{class C extends D implements } \overline{\texttt{I}} \ldots}{TT \vdash \texttt{C<:I}_i} \qquad \text{(S-CImp)}$$

$$\frac{TT(\texttt{I}) = \texttt{interface I extends } \overline{\texttt{I}} \ldots}{TT \vdash \texttt{I<:I}_i} \qquad \text{(S-IExt)}$$

3 Evaluating and Reasoning About Predicates

These rules formalize the evaluation of predicates, and they also formalize the ways in which the automatic theorem prover statically reasons about predicates. The first judgment below is straightforward and represents predicate evaluation. As usual, a type environment Γ is a mapping from variables to types, written $\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{T}}$.

$\boxed{TT;\Gamma \models \mathtt{P}}$

$$TT;\Gamma \models \mathtt{true} \qquad \text{(P-True)}$$

$$\frac{\mathtt{x}\mathtt{:}\mathtt{D} \in \Gamma \qquad TT \vdash \mathtt{D}\mathtt{<:}\mathtt{T}}{TT;\Gamma \models \mathtt{x@T}} \qquad \text{(P-Spec)}$$

$$\frac{TT;\Gamma \not\models \mathtt{P}}{TT;\Gamma \models \neg\mathtt{P}} \qquad \text{(P-Not)}$$

$$\frac{TT;\Gamma \models \mathtt{P}_1 \qquad TT;\Gamma \models \mathtt{P}_2}{TT;\Gamma \models \mathtt{P}_1\wedge\mathtt{P}_2} \qquad \text{(P-And)}$$

$$\frac{TT;\Gamma \models \mathtt{P}_1}{TT;\Gamma \models \mathtt{P}_1\vee\mathtt{P}_2} \qquad \text{(P-Or1)}$$

$$\frac{TT;\Gamma \models \mathtt{P}_2}{TT;\Gamma \models \mathtt{P}_1\vee\mathtt{P}_2} \qquad \text{(P-Or2)}$$

It is beyond the scope of this formalization, and would be quite difficult, to formally model the ways in which an automatic theorem prover manipulates predicates in order to prove them valid. Instead, we formalize the consequence of such manipulation by providing a formal notion of predicate validity. This is represented by the judgment $\overline{\mathtt{x}} \models \mathtt{P}$, defined below.

The function *restrict*$(TT, \mathtt{P})$ is used in the formalization of validity and is defined below. It creates a type table that only includes types mentioned in P and supertypes of these types. In the JPred implementation, it is only these types that the theorem prover is given axioms about. If the predicate is proven given only these axioms, the consequence is that the predicate will evaluate to true in any program that includes these types, for any possible actual arguments. That property is formalized in the rule below by the quantification over all TT' and all $\overline{\mathtt{D}}$.

An important point about the use of *restrict* is that it makes the predicate reasoning *modular*: the theorem prover is only given information about types that are mentioned in the given predicate or are depended upon by such types. In particular, the theorem prover does not have access to all subtypes of a particular type mentioned in the predicate. And more to the point, the theorem prover therefore must always assume that a pair of interfaces (or one interface and one class) could have a common subtype. As a result, the notion of validity is stronger than it otherwise would be, applying to all possible programs that include the accessible types, rather than just the program TT.

$\boxed{\overline{\mathtt{x}} \models \mathtt{P}}$

$$\frac{\begin{array}{c}\text{for all } TT' \supseteq \mathit{restrict}(TT,\mathtt{P}),\\ \text{for all } \overline{\mathtt{D}} \subseteq \mathrm{dom}(TT') \text{ of the same length as } \overline{\mathtt{x}},\\ TT';\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \mathtt{P}\end{array}}{\overline{\mathtt{x}} \models \mathtt{P}} \qquad \text{(Valid)}$$

$\boxed{\mathit{restrict}(TT,\mathtt{P}) = TT'}$

$$\frac{\mathit{neededTypes}(\mathtt{P}) = \overline{\mathtt{T}}}{\mathit{restrict}(TT,\mathtt{P}) = (\overline{\mathtt{T}}\mathtt{,}TT\mathtt{(}\overline{\mathtt{T}}\mathtt{))}} \qquad \text{(Restrict)}$$

Below we use a form of comprehension to simplify rules. The notation [A | *condition*] denotes a sequence of all elements A such that the associated condition holds.

$\boxed{\mathit{neededTypes}(\mathtt{P}) = \overline{\mathtt{T}}}$

$$\mathit{neededTypes}(\mathtt{true}) = \bullet$$

$$\frac{\overline{\mathtt{T}} = [\mathtt{T} \mid TT \vdash \mathtt{S<:T}]}{\mathit{neededTypes}(\mathtt{x@S}) = \overline{\mathtt{T}}}$$

$$\mathit{neededTypes}(\neg\mathtt{P}) = \mathit{neededTypes}(\mathtt{P})$$

$$\frac{\mathit{neededTypes}(\mathtt{P}_1) = \overline{\mathtt{S}} \qquad \mathit{neededTypes}(\mathtt{P}_2) = \overline{\mathtt{T}}}{\mathit{neededTypes}(\mathtt{P}_1\wedge\mathtt{P}_2) = \overline{\mathtt{S}}\mathtt{,}\overline{\mathtt{T}}}$$

$$\frac{\mathit{neededTypes}(\mathtt{P}_1) = \overline{\mathtt{S}} \qquad \mathit{neededTypes}(\mathtt{P}_2) = \overline{\mathtt{T}}}{\mathit{neededTypes}(\mathtt{P}_1\vee\mathtt{P}_2) = \overline{\mathtt{S}}\mathtt{,}\overline{\mathtt{T}}}$$

4 Dynamic Semantics

The evaluation rules are identical to those of FJ, except that they user our slightly modified syntax (e.g., the use of types instead of classes) and they use predicate dispatch for the invocation semantics.

$\boxed{\mathtt{t}_1 \longrightarrow \mathtt{t}_2}$

$$\frac{\mathit{fields}(\mathtt{C}) = \overline{\mathtt{T}}\ \overline{\mathtt{f}}}{\mathtt{(new\ C(}\overline{\mathtt{v}}\mathtt{)).f}_i \longrightarrow \mathtt{v}_i} \qquad \text{(E-ProjNew)}$$

$$\frac{\overline{\mathtt{u}} = \mathtt{new}\ \overline{\mathtt{D}}\mathtt{(\ldots)} \qquad \mathit{mbody}(\mathtt{m},\mathtt{C},\overline{\mathtt{D}}) = (\overline{\mathtt{x}},\mathtt{t}_0)}{\mathtt{(new\ C(}\overline{\mathtt{v}}\mathtt{)).m(}\overline{\mathtt{u}}\mathtt{)} \longrightarrow [\overline{\mathtt{x}} \mapsto \overline{\mathtt{u}},\ \mathtt{this} \mapsto \mathtt{new\ C(}\overline{\mathtt{v}}\mathtt{)}]\mathtt{t}_0} \qquad \text{(E-InvkNew)}$$

$$\frac{TT \vdash \mathtt{C<:T}}{\mathtt{(T)(new\ C(}\overline{\mathtt{v}}\mathtt{))} \longrightarrow \mathtt{new\ C(}\overline{\mathtt{v}}\mathtt{)}} \qquad \text{(E-CastNew)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_2}{\mathtt{t}_1\mathtt{.f} \longrightarrow \mathtt{t}_2\mathtt{.f}} \qquad \text{(E-Field)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_2}{\mathtt{t}_1\mathtt{.m(}\overline{\mathtt{t}}\mathtt{)} \longrightarrow \mathtt{t}_2\mathtt{.m(}\overline{\mathtt{t}}\mathtt{)}} \qquad \text{(E-INVK-RECV)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_2}{\mathtt{v.m(}\overline{\mathtt{v}}\mathtt{,t}_1\mathtt{,}\overline{\mathtt{s}}\mathtt{)} \longrightarrow \mathtt{v.m(}\overline{\mathtt{v}}\mathtt{,t}_2\mathtt{,}\overline{\mathtt{s}}\mathtt{)}} \qquad \text{(E-INVK-ARG)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_2}{\mathtt{new\ C(}\overline{\mathtt{v}}\mathtt{,t}_1\mathtt{,}\overline{\mathtt{s}}\mathtt{)} \longrightarrow \mathtt{new\ C(}\overline{\mathtt{v}}\mathtt{,t}_2\mathtt{,}\overline{\mathtt{s}}\mathtt{)}} \qquad \text{(E-NEW-ARG)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_2}{\mathtt{(T)t}_1 \longrightarrow \mathtt{(T)t}_2} \qquad \text{(E-CAST)}$$

These rules rely on the following auxiliary rules, which are again similar to those of FJ except for the addition of predicate dispatch (via the *mbody* function). In the rule defining the *overridesIfApplicable* function, $\mathtt{P}_1 \doteq \mathtt{P}_2$ holds if the two predicates refer to the same textual predicate declaration from the program. The notation $\mathtt{P}_1 \prec_{\overline{\mathtt{x}}} \mathtt{P}_2$ denotes that $\mathtt{P}_1 \preceq_{\overline{\mathtt{x}}} \mathtt{P}_2$ and $\mathtt{P}_2 \not\preceq_{\overline{\mathtt{x}}} \mathtt{P}_1$.

The rules implement the semantics analogous to so-called "encapsulated" multimethods [2, 1]. Namely, all methods in a class are considered to be more specific than inherited methods from superclasses. Another popular semantics is the so-called "symmetric" style [3], in which the receiver dispatch is not treated specially from the rest of the predicate. It would be a minor change of the rules to use this semantics, and it would not affect any of our results.

$\boxed{\mathit{fields}(\mathtt{C}) = \overline{\mathtt{T}}\ \overline{\mathtt{f}}}$

$$\mathit{fields}(\mathtt{Object}) = \bullet \qquad \text{(F-OBJ)}$$

$$\frac{\begin{array}{c} \mathit{TT}(\mathtt{C}) = \mathtt{class\ C\ extends\ D\ implements}\ \overline{\mathtt{I}}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\} \\ \mathit{fields}(\mathtt{D}) = \overline{\mathtt{S}}\ \overline{\mathtt{g}} \end{array}}{\mathit{fields}(\mathtt{C}) = \overline{\mathtt{S}}\ \overline{\mathtt{g}},\ \overline{\mathtt{T}}\ \overline{\mathtt{f}}} \qquad \text{(F-CLS)}$$

$\boxed{\mathit{mbody}(\mathtt{m},\mathtt{C},\overline{\mathtt{D}}) = (\overline{\mathtt{x}},\mathtt{t})}$

$$\frac{\begin{array}{c} \mathit{TT}(\mathtt{C}) = \mathtt{class\ C\ extends\ E\ implements}\ \overline{\mathtt{I}}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\} \\ \mathtt{S\ m(}\overline{\mathtt{S}}\ \overline{\mathtt{x}}\mathtt{)\ when}\ \overline{\mathtt{P}}\ \{\mathtt{return}\ \overline{\mathtt{t}};\} \in \overline{\mathtt{M}} \\ \mathit{TT};\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \mathtt{P}_i \qquad \mathit{overridesIfApplicable}(\mathtt{P}_i, \overline{\mathtt{P}}, \overline{\mathtt{x}}, \overline{\mathtt{D}}) \end{array}}{\mathit{mbody}(\mathtt{m},\mathtt{C},\overline{\mathtt{D}}) = (\overline{\mathtt{x}}, \mathtt{t}_i)} \qquad \text{(MBODY1)}$$

$$\frac{\begin{array}{c} \mathit{TT}(\mathtt{C}) = \mathtt{class\ C\ extends\ E\ implements}\ \overline{\mathtt{I}}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\} \\ \mathtt{S\ m(}\overline{\mathtt{S}}\ \overline{\mathtt{x}}\mathtt{)\ when}\ \overline{\mathtt{P}}\ \{\mathtt{return}\ \overline{\mathtt{t}};\} \in \overline{\mathtt{M}} \\ \text{there is no } \mathtt{P}_i \text{ such that } \mathit{TT};\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \mathtt{P}_i \end{array}}{\mathit{mbody}(\mathtt{m},\mathtt{C},\overline{\mathtt{D}}) = \mathit{mbody}(\mathtt{m},\mathtt{E},\overline{\mathtt{D}})} \qquad \text{(MBODY2)}$$

$$\frac{\begin{array}{c} \mathit{TT}(\mathtt{C}) = \mathtt{class\ C\ extends\ E\ implements}\ \overline{\mathtt{I}}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\} \\ \mathtt{m} \text{ is not defined in } \overline{\mathtt{M}} \end{array}}{\mathit{mbody}(\mathtt{m},\mathtt{C},\overline{\mathtt{D}}) = \mathit{mbody}(\mathtt{m},\mathtt{E},\overline{\mathtt{D}})} \qquad \text{(MBODY3)}$$

$\boxed{\mathit{overridesIfApplicable}(\mathtt{P}_1,\mathtt{P}_2,\overline{\mathtt{x}},\overline{\mathtt{C}})}$

$$\frac{\texttt{P}_1 \not\equiv \texttt{P}_2 \text{ and } TT; \overline{\texttt{x}}\,\texttt{:}\,\overline{\texttt{D}} \models \texttt{P}_2 \text{ implies } \texttt{P}_1 \prec_{\overline{\texttt{x}}} \texttt{P}_2}{\mathit{overridesIfApplicable}(\texttt{P}_1, \texttt{P}_2, \overline{\texttt{x}}, \overline{\texttt{D}})} \qquad (\textsc{OverApp})$$

$\boxed{\texttt{P}_1 \preceq_{\overline{\texttt{x}}} \texttt{P}_2}$

We use $\texttt{P}_1 \Rightarrow \texttt{P}_2$ to denote the predicate $\neg\texttt{P}_1 \vee \texttt{P}_2$.

$$\frac{\overline{\texttt{x}} \models \texttt{P}_1 \Rightarrow \texttt{P}_2}{\texttt{P}_1 \preceq_{\overline{\texttt{x}}} \texttt{P}_2} \qquad (\textsc{MoreSpecific})$$

5 Static Semantics

The rules for the static semantics make use of the *fields* auxiliary function defined for the dynamic semantics above.

The rules for typechecking terms are identical to those of FJ except for the slight update in syntax.

$\boxed{\Gamma \vdash \texttt{t} : \texttt{T}}$

$$\frac{\texttt{x}\,\texttt{:}\,\texttt{T} \in \Gamma}{\Gamma \vdash \texttt{x} : \texttt{T}} \qquad (\text{T-}\textsc{Var})$$

$$\frac{\Gamma \vdash \texttt{t} : \texttt{C} \qquad \mathit{fields}(\texttt{C}) = \overline{\texttt{T}}\ \overline{\texttt{f}}}{\Gamma \vdash \texttt{t.f}_i : \texttt{T}_i} \qquad (\text{T-}\textsc{Field})$$

$$\frac{\begin{array}{c}\Gamma \vdash \texttt{t}_0 : \texttt{T}_0 \qquad \mathit{mtype}(\texttt{m}, \texttt{T}_0) = \overline{\texttt{T}} \rightarrow \texttt{T} \\ \Gamma \vdash \overline{\texttt{t}} : \overline{\texttt{S}} \qquad TT \vdash \overline{\texttt{S}} \texttt{<:} \overline{\texttt{T}}\end{array}}{\Gamma \vdash \texttt{t}_0\texttt{.m(}\overline{\texttt{t}}\texttt{)} : \texttt{T}} \qquad (\text{T-}\textsc{Invk})$$

$$\frac{\mathit{fields}(\texttt{C}) = \overline{\texttt{T}}\ \overline{\texttt{f}} \qquad \Gamma \vdash \overline{\texttt{t}} : \overline{\texttt{S}} \qquad TT \vdash \overline{\texttt{S}} \texttt{<:} \overline{\texttt{T}}}{\Gamma \vdash \texttt{new C(}\overline{\texttt{t}}\texttt{)} : \texttt{C}} \qquad (\text{T-}\textsc{New})$$

$$\frac{\Gamma \vdash \texttt{t} : \texttt{S} \qquad TT \vdash \texttt{S<:T}}{\Gamma \vdash \texttt{(T)t} : \texttt{T}} \qquad (\text{T-U}\textsc{Cast})$$

$$\frac{\Gamma \vdash \texttt{t} : \texttt{S} \qquad TT \vdash \texttt{T<:S} \qquad \texttt{T} \neq \texttt{S}}{\Gamma \vdash \texttt{(T)t} : \texttt{T}} \qquad (\text{T-D}\textsc{Cast})$$

$$\frac{\begin{array}{c}\Gamma \vdash \texttt{t} : \texttt{S} \qquad TT \vdash \texttt{S} \not\texttt{<:} \texttt{T} \qquad TT \vdash \texttt{T} \not\texttt{<:} \texttt{S} \\ \textit{stupid warning}\end{array}}{\Gamma \vdash \texttt{(T)t} : \texttt{T}} \qquad (\text{T-S}\textsc{Cast})$$

These rules rely on the following auxiliary rules for *mtype*, which extend the rules used in FJ to allow for getting the type of a method from an interface. Note that the rules allow a method's type to be nondeterministically found by looking in any superinterface of an interface. Other rules will ensure that all these interfaces agree on the type of each method name that they have in common. Note also that we need not search the superinterfaces of a class, since other rules will guarantee that the class declares or inherits at least one implementation of the method.

$\boxed{\mathit{mtype}(\mathtt{m},\mathtt{T}_0) = \overline{\mathtt{T}}\rightarrow\mathtt{T}}$

$$\frac{\begin{array}{c}TT(\mathtt{C}) = \mathtt{class\ C\ extends\ D\ implements}\ \overline{\mathtt{I}}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\} \\ \mathtt{S\ m(}\overline{\mathtt{S}}\ \overline{\mathtt{x}}\mathtt{)\ when}\ \overline{\mathtt{P}}\ \{\mathtt{return}\ \overline{\mathtt{t}};\} \in \overline{\mathtt{M}}\end{array}}{\mathit{mtype}(\mathtt{m}, \mathtt{C}) = \overline{\mathtt{S}}\rightarrow\mathtt{S}} \qquad \text{(MTYPE-C1)}$$

$$\frac{\begin{array}{c}TT(\mathtt{C}) = \mathtt{class\ C\ extends\ D\ implements}\ \overline{\mathtt{I}}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\} \\ \mathtt{m}\text{ is not defined in }\overline{\mathtt{M}} \qquad \mathit{mtype}(\mathtt{m}, \mathtt{D}) = \overline{\mathtt{S}}\rightarrow\mathtt{S}\end{array}}{\mathit{mtype}(\mathtt{m}, \mathtt{C}) = \overline{\mathtt{S}}\rightarrow\mathtt{S}} \qquad \text{(MTYPE-C2)}$$

$$\frac{\begin{array}{c}TT(\mathtt{I}) = \mathtt{interface\ I\ extends}\ \overline{\mathtt{I}}\ \{\overline{\mathtt{MH}}\} \\ \mathtt{S\ m(}\overline{\mathtt{S}}\ \overline{\mathtt{x}}\mathtt{);} \in \overline{\mathtt{MH}}\end{array}}{\mathit{mtype}(\mathtt{m}, \mathtt{I}) = \overline{\mathtt{S}}\rightarrow\mathtt{S}} \qquad \text{(MTYPE-I1)}$$

$$\frac{\begin{array}{c}TT(\mathtt{I}) = \mathtt{interface\ I\ extends}\ \overline{\mathtt{I}}\ \{\overline{\mathtt{MH}}\} \\ \mathtt{m}\text{ is not defined in }\overline{\mathtt{MH}} \qquad \mathit{mtype}(\mathtt{m}, \mathtt{I}_i) = \overline{\mathtt{S}}\rightarrow\mathtt{S}\end{array}}{\mathit{mtype}(\mathtt{m}, \mathtt{I}) = \overline{\mathtt{S}}\rightarrow\mathtt{S}} \qquad \text{(MTYPE-I2)}$$

Next we have rules for typechecking predicates. These rules ensure that only variables in scope (i.e., the method's formal parameters) are referred to.

$\boxed{\overline{\mathtt{x}} \vdash \mathtt{P\ OK}}$

$$\overline{\mathtt{x}} \vdash \mathtt{true\ OK} \qquad \text{(P-TRUE)}$$

$$\frac{\mathtt{x} \in \overline{\mathtt{x}}}{\overline{\mathtt{x}} \vdash \mathtt{x@S\ OK}} \qquad \text{(P-TEST)}$$

$$\frac{\overline{\mathtt{x}} \vdash \mathtt{P\ OK}}{\overline{\mathtt{x}} \vdash \neg\mathtt{P\ OK}} \qquad \text{(P-NOT)}$$

$$\frac{\overline{\mathtt{x}} \vdash \mathtt{P}_1\ \mathtt{OK} \qquad \overline{\mathtt{x}} \vdash \mathtt{P}_2\ \mathtt{OK}}{\overline{\mathtt{x}} \vdash \mathtt{P}_1{\wedge}\mathtt{P}_2\ \mathtt{OK}} \qquad \text{(P-AND)}$$

$$\frac{\overline{\mathtt{x}} \vdash \mathtt{P}_1\ \mathtt{OK} \qquad \overline{\mathtt{x}} \vdash \mathtt{P}_2\ \mathtt{OK}}{\overline{\mathtt{x}} \vdash \mathtt{P}_1{\vee}\mathtt{P}_2\ \mathtt{OK}} \qquad \text{(P-OR)}$$

Next we have rules for typechecking methods. These rules augment the rule for typechecking methods in FJ to typecheck predicates and to ensure that a class's methods have the same type signatures as those of the same name that it inherits from interfaces.

Notice that we typecheck the body of a method using the declared static types of the formals. It would be safe to sometimes narrow these types based on the type tests in the method's predicate. The full JPred language does so, but we have elided it for simplicity.

$\boxed{\mathtt{M\ OK\ in\ C}}$

$$\frac{\overline{\mathtt{x}} \vdash \overline{\mathtt{P}}\ \mathtt{OK} \qquad \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{T}}\mathtt{,this:C} \vdash \overline{\mathtt{t}} : \overline{\mathtt{S}} \qquad TT \vdash \overline{\mathtt{S}}\mathtt{<:T} \\ \mathit{unambiguous}(\mathtt{P}_1, \overline{\mathtt{P}}, \overline{\mathtt{x}}, \overline{\mathtt{P}}) \cdots \mathit{unambiguous}(\mathtt{P}_n, \overline{\mathtt{P}}, \overline{\mathtt{x}}, \overline{\mathtt{P}})}{\mathtt{T\ m(}\overline{\mathtt{T}}\ \overline{\mathtt{x}}\mathtt{)\ when\ }\overline{\mathtt{P}}\ \{\mathtt{return\ }\overline{\mathtt{t}}\mathtt{;}\}\ \mathtt{OK\ in\ C}} \quad \text{(T-METH)}$$

The rules for typechecking methods rely on the following rule, which implements pairwise ambiguity checking of methods. The notation $\bigvee\overline{\mathtt{P}}$ denotes the disjunction of all predicates in $\overline{\mathtt{P}}$. We define $\bigvee\bullet$ as the predicate $\neg\mathtt{true}$. The rule first ensures that the two given predicates are not equivalent unless they are the same textual predicate. Finally, the rule requires that whenever both predicates are satisfied, then at least one predicate that is more specific than both of them is also satisfied.

There are two common cases of this last requirement that are worth noting. First, if $\mathtt{P}_1$ is strictly more specific than $\mathtt{P}_2$, then $\mathtt{P}_1$ is always satisfied whenever both are, so the methods are unambiguous (and similarly for the symmetric case). Second, if $\mathtt{P}_1$ and $\mathtt{P}_2$ are *disjoint*, meaning that they cannot be simultaneously true, then the requirement holds vacuously.

$\boxed{\mathit{unambiguous}(\mathtt{P}_1, \mathtt{P}_2, \overline{\mathtt{x}}, \overline{\mathtt{P}})}$

$$\frac{\mathtt{P}_1 \preceq_{\overline{\mathtt{x}}} \mathtt{P}_2 \text{ and } \mathtt{P}_2 \preceq_{\overline{\mathtt{x}}} \mathtt{P}_1 \text{ implies } \mathtt{P}_1 \doteq \mathtt{P}_2 \\ \overline{\mathtt{Q}} = [\mathtt{P} \mid \mathtt{P} \in \overline{\mathtt{P}} \text{ and } \mathtt{P} \preceq_{\overline{\mathtt{x}}} \mathtt{P}_1 \text{ and } \mathtt{P} \preceq_{\overline{\mathtt{x}}} \mathtt{P}_2] \\ \overline{\mathtt{x}} \models (\mathtt{P}_1 \wedge \mathtt{P}_2) \Rightarrow \bigvee\overline{\mathtt{Q}}}{\mathit{unambiguous}(\mathtt{P}_1, \mathtt{P}_2, \overline{\mathtt{x}}, \overline{\mathtt{P}})} \quad \text{(UNAMB)}$$

Finally we present the rules for typechecking classes and interfaces. The rule for classes is as in FJ, with the addition of exhaustiveness checking for all methods that should be implemented or inherited by the class.

$\boxed{\mathtt{TD\ OK}}$

$$\frac{\mathtt{K} = \mathtt{C(}\overline{\mathtt{S}}\ \overline{\mathtt{g}}\mathtt{,\ }\overline{\mathtt{T}}\ \overline{\mathtt{f}}\mathtt{)}\ \{\mathtt{super(}\overline{\mathtt{g}}\mathtt{);\ this.}\overline{\mathtt{f}}\ \mathtt{=}\ \overline{\mathtt{f}}\mathtt{;}\} \\ \mathit{fields}(\mathtt{D}) = \overline{\mathtt{S}}\ \overline{\mathtt{g}} \qquad \overline{\mathtt{M}}\ \mathtt{OK\ in\ C} \\ \mathit{allMethodNames}(\mathtt{C}) = \overline{\mathtt{m}} \\ \mathit{override}(\overline{\mathtt{m}}, \mathtt{C}) \qquad \mathit{exhaustive}(\overline{\mathtt{m}}, \mathtt{C})}{\mathtt{class\ C\ extends\ D\ implements\ }\overline{\mathtt{I}}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}}\mathtt{;\ K\ }\overline{\mathtt{M}}\}\ \mathtt{OK}} \quad \text{(T-CLASS)}$$

$$\frac{\mathit{allMethodNames}(\mathtt{I}) = \overline{\mathtt{m}} \qquad \mathit{override}(\overline{\mathtt{m}}, \mathtt{I})}{\mathtt{interface\ I\ extends\ }\overline{\mathtt{I}}\ \{\overline{\mathtt{MH}}\}\ \mathtt{OK}} \quad \text{(T-INT)}$$

These rules find all method names in a given class and all supertypes.

$\boxed{\mathit{allMethodNames}(\mathtt{T}) = \overline{\mathtt{m}}}$

$$\frac{TT(\mathtt{C}) = \mathtt{class\ C\ extends\ D\ implements\ }\overline{\mathtt{I}}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}}\mathtt{;\ K\ }\overline{\mathtt{M}}\} \\ \mathit{mname}(\overline{\mathtt{M}}) = \overline{\mathtt{m}} \qquad \mathit{allMethodNames}(\mathtt{D}) = \overline{\mathtt{m}_0} \\ \mathit{allMethodNames}(\mathtt{I}_1) = \overline{\mathtt{m}_1} \cdots \mathit{allMethodNames}(\mathtt{I}_n) = \overline{\mathtt{m}_n}}{\mathit{allMethodNames}(\mathtt{C}) = \overline{\mathtt{m}}, \overline{\mathtt{m}_0}, \overline{\mathtt{m}_1}, \ldots, \overline{\mathtt{m}_n}} \quad \text{(MNAMES-CLS)}$$

$$\frac{TT(\mathtt{I}) = \mathtt{interface\ I\ extends\ }\overline{\mathtt{I}}\ \{\overline{\mathtt{MH}}\} \\ \mathit{mname}(\overline{\mathtt{MH}}) = \overline{\mathtt{m}} \\ \mathit{allMethodNames}(\mathtt{I}_1) = \overline{\mathtt{m}_1} \cdots \mathit{allMethodNames}(\mathtt{I}_n) = \overline{\mathtt{m}_n}}{\mathit{allMethodNames}(\mathtt{I}) = \overline{\mathtt{m}}, \overline{\mathtt{m}_1}, \ldots, \overline{\mathtt{m}_n}} \quad \text{(MNAMES-INT)}$$

$\boxed{\textit{override}(\mathtt{m},\mathtt{T})}$

$$\frac{\begin{array}{c} TT(\mathtt{C}) = \mathtt{class\ C\ extends\ D\ implements\ \overline{I}\ \{\overline{T}\ \overline{f};\ K\ \overline{M}\}} \\ \textit{mtype}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{S}}{\rightarrow}\mathtt{S} \qquad \textit{mformals}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{x}} \\ \textit{override}(\mathtt{m},\mathtt{D},\overline{\mathtt{S}}{\rightarrow}\mathtt{S},\overline{\mathtt{x}}) \qquad \textit{override}(\mathtt{m},\overline{\mathtt{I}},\overline{\mathtt{S}}{\rightarrow}\mathtt{S}) \end{array}}{\textit{override}(\mathtt{m},\mathtt{C})} \qquad \text{(Over-Cls)}$$

$$\frac{\begin{array}{c} TT(\mathtt{I}) = \mathtt{interface\ I\ extends\ \overline{I}\ \{\overline{MH}\}} \\ \textit{mtype}(\mathtt{m},\mathtt{I}) = \overline{\mathtt{S}}{\rightarrow}\mathtt{S} \qquad \textit{override}(\mathtt{m},\overline{\mathtt{I}},\overline{\mathtt{S}}{\rightarrow}\mathtt{S}) \end{array}}{\textit{override}(\mathtt{m},\mathtt{I})} \qquad \text{(Over-Int)}$$

The above rules rely on the following rule, which is analogous to the one from FJ but additionally requires method implementations to have the same formal parameter names as the method implementations they override. The condition simplifies the rules for exhaustiveness checking. Interface methods are not required to obey this condition, so they have a separate rule defined subsequently.

$\boxed{\textit{override}(\mathtt{m},\mathtt{C},\overline{\mathtt{T}}{\rightarrow}\mathtt{T}_0,\overline{\mathtt{x}})}$

$$\frac{\begin{array}{c} \textit{mtype}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{S}}{\rightarrow}\mathtt{S}_0 \text{ implies } \overline{\mathtt{S}} = \overline{\mathtt{T}} \text{ and } \mathtt{S}_0 = \mathtt{T}_0 \\ \textit{mformals}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{y}} \text{ implies } \overline{\mathtt{x}} = \overline{\mathtt{y}} \end{array}}{\textit{override}(\mathtt{m},\mathtt{C},\overline{\mathtt{T}}{\rightarrow}\mathtt{T}_0,\overline{\mathtt{x}})} \qquad \text{(T-OverCls)}$$

$\boxed{\textit{override}(\mathtt{m},\mathtt{I},\overline{\mathtt{T}}{\rightarrow}\mathtt{T}_0)}$

$$\frac{\textit{mtype}(\mathtt{m},\mathtt{I}) = \overline{\mathtt{S}}{\rightarrow}\mathtt{S}_0 \text{ implies } \overline{\mathtt{S}} = \overline{\mathtt{T}} \text{ and } \mathtt{S}_0 = \mathtt{T}_0}{\textit{override}(\mathtt{m},\mathtt{I},\overline{\mathtt{T}}{\rightarrow}\mathtt{T}_0)} \qquad \text{(T-OverInt)}$$

$\boxed{\textit{mformals}(\mathtt{m},\mathtt{T}) = \overline{\mathtt{x}}}$

$$\frac{\begin{array}{c} TT(\mathtt{C}) = \mathtt{class\ C\ extends\ D\ implements\ \overline{I}\ \{\overline{T}\ \overline{f};\ K\ \overline{M}\}} \\ \mathtt{S\ m(\overline{S}\ \overline{x})\ when\ \overline{P}\ \{return\ \overline{t};\}} \in \overline{\mathtt{M}} \end{array}}{\textit{mformals}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{x}}} \qquad \text{(MFormals1)}$$

$$\frac{\begin{array}{c} TT(\mathtt{C}) = \mathtt{class\ C\ extends\ D\ implements\ \overline{I}\ \{\overline{T}\ \overline{f};\ K\ \overline{M}\}} \\ \mathtt{m} \text{ is not defined in } \overline{\mathtt{M}} \qquad \textit{mformals}(\mathtt{m},\mathtt{D}) = \overline{\mathtt{x}} \end{array}}{\textit{mformals}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{x}}} \qquad \text{(MFormals2)}$$

$\boxed{\textit{exhaustive}(\mathtt{m},\mathtt{C})}$

A method implementation is exhaustive if the disjunction of all predicates is valid.

$$\frac{\textit{mpreds}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{P}} \qquad \textit{mformals}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{x}} \qquad \overline{\mathtt{x}} \models \bigvee \overline{\mathtt{P}}}{\textit{exhaustive}(\mathtt{m},\mathtt{C})} \qquad \text{(T-Exhaust)}$$

$\boxed{\textit{mpreds}(\mathtt{m},\mathtt{C}) = \overline{\mathtt{P}}}$

This judgment is used to collect up all the predicates in implementations of a given method.

$$\mathit{mpreds}(\mathtt{m}, \mathtt{Object}) = \bullet \qquad \text{(MPreds1)}$$

$$\frac{\begin{array}{c} TT(\mathtt{C}) = \mathtt{class\ C\ extends\ D\ implements}\ \overline{\mathtt{I}}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\} \\ \mathtt{S\ m}(\overline{\mathtt{S}}\ \overline{\mathtt{x}})\ \mathtt{when}\ \overline{\mathtt{P}}\ \{\mathtt{return}\ \overline{\mathtt{t}};\} \in \overline{\mathtt{M}} \\ \mathit{mpreds}(\mathtt{m}, \mathtt{D}) = \overline{\mathtt{Q}} \end{array}}{\mathit{mpreds}(\mathtt{m}, \mathtt{C}) = \overline{\mathtt{P}}, \overline{\mathtt{Q}}} \qquad \text{(MPreds2)}$$

$\boxed{\mathit{mname}(\mathtt{M}) = \mathtt{m}}$

$$\mathit{mname}(\mathtt{T\ m}(\overline{\mathtt{T}}\ \overline{\mathtt{x}})\cdots) = \mathtt{m} \qquad \text{(MName-Cls)}$$

$\boxed{\mathit{mname}(\mathtt{MH}) = \mathtt{m}}$

$$\mathit{mname}(\mathtt{T\ m}(\overline{\mathtt{T}}\ \overline{\mathtt{x}});) = \mathtt{m} \qquad \text{(MName-Int)}$$

6 Type Soundness

We prove type soundness in the standard "progress and preservation" style. Preservation is pretty much standard; progress must reason about exhaustiveness and ambiguity checking in order to show that method lookup always succeeds at run time.

Analogous with FJ, we assume that `TD OK` holds for each type declaration `TD` in the range of TT.

6.1 Type Preservation

Lemma 6.1 If $TT \vdash \mathtt{T}\mathtt{<:}\mathtt{C}$, then `T` is a class.
Proof By induction on the depth of the derivation of $TT \vdash \mathtt{T}\mathtt{<:}\mathtt{C}$. Case analysis of the last rule in the derivation.

- Case S-Ref: Then $\mathtt{T} = \mathtt{C}$ and the result follows.
- Case S-Trans: Then $TT \vdash \mathtt{T}\mathtt{<:}\mathtt{T}_0$ and $TT \vdash \mathtt{T}_0\mathtt{<:}\mathtt{C}$. By induction $\mathtt{T}_0$ is a class, and by induction again so is `T`.
- Case S-CExt: Then we are given that `T` is a class.
- Case S-CImp: Then `C` is an interface, contradicting our initial assumption.
- Case S-IExt: Then `C` is an interface, contradicting our initial assumption.

□

Lemma 6.2 If $TT \vdash \mathtt{D}\mathtt{<:}\mathtt{C}$ and $\mathit{fields}(\mathtt{C}) = \overline{\mathtt{T}}\ \overline{\mathtt{f}}$, then $\mathit{fields}(\mathtt{C}) \subseteq \mathit{fields}(\mathtt{D})$.
Proof By induction on the depth of the derivation of $TT \vdash \mathtt{D}\mathtt{<:}\mathtt{C}$. Case analysis of the last rule in the derivation.

- Case S-Ref: Then $\mathtt{D} = \mathtt{C}$ and the result follows.

- Case S-Trans: Then $TT \vdash \mathtt{D<:T}$ and $TT \vdash \mathtt{T<:C}$. By Lemma 6.1 we have that T is some class E. Then by induction we have *fields*(C) $\subseteq$ *fields*(E), and by induction again we have *fields*(E) $\subseteq$ *fields*(D). Then by transitivity of $\subseteq$ the result follows.

- Case S-CExt: Then TT(D) = `class D extends C implements` $\overline{\mathtt{I}}$ $\{\overline{\mathtt{S}}\ \overline{\mathtt{g}}$; ...}. By F-Cls we have *fields*(D) = $\overline{\mathtt{T}}\ \overline{\mathtt{f}}$, $\overline{\mathtt{S}}\ \overline{\mathtt{g}}$, so the result follows.

- Case S-CImp: Then C is an interface, contradicting our initial assumption.

- Case S-IExt: Then C is an interface, contradicting our initial assumption.

□

Lemma 6.3 If *mtype*(m,T) = $\overline{\mathtt{S}}\to\mathtt{S}$ and *allMethodNames*(T) = $\overline{\mathtt{m}}$, then $\mathtt{m} \in \overline{\mathtt{m}}$.
Proof By induction on the depth of the derivation of *mtype*(m,T) = $\overline{\mathtt{S}}\to\mathtt{S}$. Case analysis of the last rule in the derivation.

- Case MType-C1. Then T is a class C and TT(C) = `class C extends D implements` $\overline{\mathtt{I}}$ $\{\overline{\mathtt{T}}\ \overline{\mathtt{f}}$; K $\overline{\mathtt{M}}\}$ and S m($\overline{\mathtt{S}}\ \overline{\mathtt{x}}$) `when` $\overline{\mathtt{P}}$ {`return` $\overline{\mathtt{t}}$;} $\in \overline{\mathtt{M}}$. Since *allMethodNames*(T) = $\overline{\mathtt{m}}$, by MNames-Cls we have *mname*($\overline{\mathtt{M}}$) $\subseteq \overline{\mathtt{m}}$, and by MName-Cls that means $\mathtt{m} \in \overline{\mathtt{m}}$.

- Case MType-C2. Then T is a class C and TT(C) = `class C extends D implements` $\overline{\mathtt{I}}$ $\{\overline{\mathtt{T}}\ \overline{\mathtt{f}}$; K $\overline{\mathtt{M}}\}$ and *mtype*(m,D) = $\overline{\mathtt{S}}\to\mathtt{S}$. By T-Class we have that *allMethodNames*(D) = $\overline{\mathtt{m_0}}$, so by induction we have $\mathtt{m} \in \overline{\mathtt{m_0}}$. Then by MNames-Cls also $\mathtt{m} \in \overline{\mathtt{m}}$.

- Case MType-I1. Then T is an interface I and TT(I) = `interface I extends` $\overline{\mathtt{I}}$ $\{\overline{\mathtt{MH}}\}$ and S m($\overline{\mathtt{S}}\ \overline{\mathtt{x}}$); $\in \overline{\mathtt{MH}}$. Since *allMethodNames*(T) = $\overline{\mathtt{m}}$, by MNames-Int we have *mname*($\overline{\mathtt{MH}}$) $\subseteq \overline{\mathtt{m}}$, and by MName-Int that means $\mathtt{m} \in \overline{\mathtt{m}}$.

- Case MType-I2. Then T is an interface I and TT(I) = `interface I extends` $\overline{\mathtt{I}}$ $\{\overline{\mathtt{MH}}\}$ and *mtype*(m,$\mathtt{I}_i$) = $\overline{\mathtt{S}}\to\mathtt{S}$. By T-Int we have that *allMethodNames*($\mathtt{I}_i$) = $\overline{\mathtt{m_0}}$, so by induction we have $\mathtt{m} \in \overline{\mathtt{m_0}}$. Then by MNames-Int also $\mathtt{m} \in \overline{\mathtt{m}}$.

□

Lemma 6.4 If $TT \vdash \mathtt{S'<:S}$ and *mtype*(m,S) = $\overline{\mathtt{T}}\to\mathtt{T}$, then *mtype*(m,S′) = $\overline{\mathtt{T}}\to\mathtt{T}$.
Proof By induction on the depth of the derivation of $TT \vdash \mathtt{S'<:S}$. Case analysis of the last rule in the derivation.

- Case S-Ref: Then S′ = S and the result follows.

- Case S-Trans: Then $TT \vdash \mathtt{S'<:S_0}$ and $TT \vdash \mathtt{S_0<:S}$. By induction we have *mtype*(m,$\mathtt{S}_0$) = $\overline{\mathtt{T}}\to\mathtt{T}$, and by induction again we have *mtype*(m,S′) = $\overline{\mathtt{T}}\to\mathtt{T}$.

- Case S-CExt: Then S′ is a class C and S is a class D and TT(C) = `class C extends D implements` $\overline{\mathtt{I}}$ $\{\cdots\}$. By T-Class we have *allMethodNames*(C) = $\overline{\mathtt{m}}$ and *allMethodNames*(D) = $\overline{\mathtt{m_0}}$. Then by Lemma 6.3 we have $\mathtt{m} \in \overline{\mathtt{m_0}}$, and by MNames-Cls also $\mathtt{m} \in \overline{\mathtt{m}}$. Then by T-Class we have *override*(m,C), so by Over-Cls we have *mtype*(m, C) = $\overline{\mathtt{S}}\to\mathtt{S}_0$ and *override*(m, D, $\overline{\mathtt{S}}\to\mathtt{S}_0$, $\overline{\mathtt{x}}$). Finally, by T-OverCls we have that $\overline{\mathtt{T}} = \overline{\mathtt{S}}$ and $\mathtt{T} = \mathtt{S}_0$, so the result follows.

- Case S-CImp: Then $\mathtt{S}'$ is a class $\mathtt{C}$ and $\mathtt{S}$ is an interface $\mathtt{I}_i$ and $TT(\mathtt{C}) =$ `class C extends D implements` $\overline{\mathtt{I}}$ $\{\cdots\}$. By T-Class we have *allMethodNames*$(\mathtt{C}) = \overline{\mathtt{m}}$, and by T-Int we have *allMethodNames*$(\mathtt{I}_i) = \overline{\mathtt{m}_0}$. Then by Lemma 6.3 we have $\mathtt{m} \in \overline{\mathtt{m}_0}$, and by MNames-Cls also $\mathtt{m} \in \overline{\mathtt{m}}$. Then by T-Class we have *override*$(\mathtt{m},\mathtt{C})$, so by Over-Cls we have *mtype*$(\mathtt{m},\mathtt{C}) = \overline{\mathtt{S}}\rightarrow\mathtt{S}_0$ and *override*$(\mathtt{m},\mathtt{I}_i,\overline{\mathtt{S}}\rightarrow\mathtt{S}_0)$. Finally, by T-OverInt we have that $\overline{\mathtt{T}} = \overline{\mathtt{S}}$ and $\mathtt{T} = \mathtt{S}_0$, so the result follows.

- Case S-IExt: Then $\mathtt{S}'$ is an interface $\mathtt{I}$ and $\mathtt{S}$ is an interface $\mathtt{I}_i$ and $TT(\mathtt{I}) =$ `interface I extends implements` $\overline{\mathtt{I}}$ $\{\cdots\}$. By T-Int we have *allMethodNames*$(\mathtt{I}) = \overline{\mathtt{m}}$ and *allMethodNames*$(\mathtt{I}_i) = \overline{\mathtt{m}_0}$. Then by Lemma 6.3 we have $\mathtt{m} \in \overline{\mathtt{m}_0}$, and by MNames-Int also $\mathtt{m} \in \overline{\mathtt{m}}$. Then by T-Int we have *override*$(\mathtt{m},\mathtt{I})$, so by Over-Int we have *mtype*$(\mathtt{m},\mathtt{I}) = \overline{\mathtt{S}}\rightarrow\mathtt{S}_0$ and *override*$(\mathtt{m},\mathtt{I}_i,\overline{\mathtt{S}}\rightarrow\mathtt{S}_0)$. Finally, by T-OverInt we have that $\overline{\mathtt{T}} = \overline{\mathtt{S}}$ and $\mathtt{T} = \mathtt{S}_0$, so the result follows.

□

Lemma 6.5 (Substitution) If $\Gamma,\overline{\mathtt{x}}{:}\overline{\mathtt{T}} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \overline{\mathtt{s}} : \overline{\mathtt{S}}$ and $TT \vdash \overline{\mathtt{S}}\mathtt{<:}\overline{\mathtt{T}}$, then $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{t} : \mathtt{S}$ for some $TT \vdash \mathtt{S}\mathtt{<:}\mathtt{T}$.
Proof By induction on the depth of the derivation of $\Gamma,\overline{\mathtt{x}}{:}\overline{\mathtt{T}} \vdash \mathtt{t} : \mathtt{T}$. Case analysis of the last rule in the derivation.

- Case T-Var: Then $\mathtt{t}$ has the form $\mathtt{x}$ and $\mathtt{x{:}T} \in \Gamma,\overline{\mathtt{x}}{:}\overline{\mathtt{T}}$. If $\mathtt{x} \notin \overline{\mathtt{x}}$ then we have $\mathtt{x{:}T} \in \Gamma$, so by T-Var we have $\Gamma \vdash \mathtt{x} : \mathtt{T}$. Since $\mathtt{x} \notin \overline{\mathtt{x}}$, we have $[\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{x} = \mathtt{x}$, and by S-Ref we know $TT \vdash \mathtt{T<:T}$, so the result follows. On the other hand, if $\mathtt{x} \in \overline{\mathtt{x}}$ then $\mathtt{x}$ has the form $\mathtt{x}_i$ and $\mathtt{T} = \mathtt{T}_i$ and $[\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{x} = \mathtt{s}_i$. We're given that $\Gamma \vdash \mathtt{s}_i : \mathrm{S}_i$ and $TT \vdash \mathtt{S}_i\mathtt{<:}\mathtt{T}_i$, so the result follows.

- Case T-Field: Then $\mathtt{t}$ has the form $\mathtt{s.f}_i$ and $\mathtt{T}$ has the form $\mathtt{U}_i$ and $\Gamma,\overline{\mathtt{x}}{:}\overline{\mathtt{T}} \vdash \mathtt{s} : \mathtt{C}$ and *fields*$(\mathtt{C}) = \overline{\mathtt{U}}\ \overline{\mathtt{f}}$. By induction we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{s} : \mathtt{C}_0$ and $TT \vdash \mathtt{C}_0\mathtt{<:C}$. By Lemma 6.2 we have *fields*$(\mathtt{C}) \subseteq$ *fields*$(\mathtt{C}_0)$, so by T-Field also $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{s.f}_i : \mathtt{T}_i$, and by S-Ref we have $TT \vdash \mathtt{T}_i\mathtt{<:}\mathtt{T}_i$.

- Case T-Invk: Then $\mathtt{t}$ has the form $\mathtt{t}_0\mathtt{.m}(\overline{\mathtt{t}})$ and $\Gamma,\overline{\mathtt{x}}{:}\overline{\mathtt{T}} \vdash \mathtt{t}_0 : \mathtt{T}_0$ and *mtype*$(\mathtt{m},\mathtt{T}_0) = \overline{\mathtt{U}}\rightarrow\mathtt{T}$ and $\Gamma,\overline{\mathtt{x}}{:}\overline{\mathtt{T}} \vdash \overline{\mathtt{t}} : \overline{\mathtt{U}_0}$ and $TT \vdash \overline{\mathtt{U}_0}\mathtt{<:}\overline{\mathtt{U}}$. By induction we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{t}_0 : \mathtt{T}_0'$ and $TT \vdash \mathtt{T}_0'\mathtt{<:}\mathtt{T}_0$. By Lemma 6.4 we have *mtype*$(\mathtt{m},\mathtt{T}_0') = \overline{\mathtt{U}}\rightarrow\mathtt{T}$. Also by induction we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\overline{\mathtt{t}} : \overline{\mathtt{U}_0'}$ and $TT \vdash \overline{\mathtt{U}_0'}\mathtt{<:}\overline{\mathtt{U}_0}$. Then by S-Trans we have $TT \vdash \overline{\mathtt{U}_0'}\mathtt{<:}\overline{\mathtt{U}}$. So by T-Invk we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{t}_0\mathtt{.m}(\overline{\mathtt{t}}) : \mathtt{T}$, and by S-Ref we have $TT \vdash \mathtt{T<:T}$.

- Case T-New: Then $\mathtt{t}$ has the form $\mathtt{new\ C}(\overline{\mathtt{t}})$ and $\mathtt{T} = \mathtt{C}$ and *fields*$(\mathtt{C}) = \overline{\mathtt{U}}\ \overline{\mathtt{f}}$ and $\Gamma,\overline{\mathtt{x}}{:}\overline{\mathtt{T}} \vdash \overline{\mathtt{t}} : \overline{\mathtt{U}_0}$ and $TT \vdash \overline{\mathtt{U}_0}\mathtt{<:}\overline{\mathtt{U}}$. By induction we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\overline{\mathtt{t}} : \overline{\mathtt{U}_0'}$ and $TT \vdash \overline{\mathtt{U}_0'}\mathtt{<:}\overline{\mathtt{U}_0}$. Then by S-Trans we have $TT \vdash \overline{\mathtt{U}_0'}\mathtt{<:}\overline{\mathtt{U}}$. So by T-New we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{new\ C}(\overline{\mathtt{t}}) : \mathtt{C}$, and by S-Ref we have $TT \vdash \mathtt{C<:C}$.

- Case T-UCast: Then $\mathtt{t}$ has the form $\mathtt{(T)t}_0$ and $\Gamma,\overline{\mathtt{x}}{:}\overline{\mathtt{T}} \vdash \mathtt{t}_0 : \mathtt{T}_0$ and $TT \vdash \mathtt{T}_0\mathtt{<:T}$. By induction we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{t}_0 : \mathtt{T}_0'$ and $TT \vdash \mathtt{T}_0'\mathtt{<:}\mathtt{T}_0$. By S-Trans also $TT \vdash \mathtt{T}_0'\mathtt{<:T}$, so by T-UCast we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{(T)t}_0 : \mathtt{T}$. Finally, by S-Ref we have $TT \vdash \mathtt{T<:T}$.

- Case T-DCast: Then $\mathtt{t}$ has the form $\mathtt{(T)t}_0$ and $\Gamma,\overline{\mathtt{x}}{:}\overline{\mathtt{T}} \vdash \mathtt{t}_0 : \mathtt{T}_0$ and $TT \vdash \mathtt{T<:}\mathtt{T}_0$ and $\mathtt{T} \neq \mathtt{T}_0$. By induction we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{t}_0 : \mathtt{T}_0'$ and $TT \vdash \mathtt{T}_0'\mathtt{<:}\mathtt{T}_0$. If $TT \vdash \mathtt{T}_0'\mathtt{<:T}$, then by T-UCast we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{(T)t}_0 : \mathtt{T}$. Otherwise if $TT \vdash \mathtt{T<:}\mathtt{T}_0'$, then by T-DCast we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{(T)t}_0 : \mathtt{T}$. Otherwise we have $TT \vdash \mathtt{T}_0' \not<: \mathtt{T}$ and $TT \vdash \mathtt{T} \not<: \mathtt{T}_0'$, so by T-SCast we have $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{s}}]\mathtt{(T)t}_0 : \mathtt{T}$ and a *stupid warning* is generated. Finally, by S-Ref we have $TT \vdash \mathtt{T<:T}$.

- Case T-SCast: Then $\mathtt{t}$ has the form $\mathtt{(T)t_0}$ and $\Gamma\mathtt{,\overline{x}:\overline{T}} \vdash \mathtt{t_0} : \mathtt{T_0}$ and $TT \vdash \mathtt{T_0}\not<\mathtt{:T}$ and $TT \vdash \mathtt{T}\not<\mathtt{:T_0}$. By induction we have $\Gamma \vdash \mathtt{[\overline{x} \mapsto \overline{s}]t_0} : \mathtt{T'_0}$ and $TT \vdash \mathtt{T'_0{<:}T_0}$. If $TT \vdash \mathtt{T'_0{<:}T}$, then by T-UCast we have $\Gamma \vdash \mathtt{[\overline{x} \mapsto \overline{s}](T)t_0} : \mathtt{T}$. Otherwise if $TT \vdash \mathtt{T{<:}T'_0}$, then by S-Trans we have $TT \vdash \mathtt{T{<:}T_0}$, which contradicts the fact that $TT \vdash \mathtt{T}\not<\mathtt{:T_0}$, so it is not possible that $TT \vdash \mathtt{T{<:}T'_0}$. Otherwise we have $TT \vdash \mathtt{T'_0}\not<\mathtt{:T}$ and $TT \vdash \mathtt{T}\not<\mathtt{:T'_0}$, so by T-SCast we have $\Gamma \vdash \mathtt{[\overline{x} \mapsto \overline{s}](T)t_0} : \mathtt{T}$. Finally, by S-Ref we have $TT \vdash \mathtt{T{<:}T}$.

□

Lemma 6.6 (Weakening) If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{x} \notin \mathrm{dom}(\Gamma)$, then $\Gamma\mathtt{,x:S} \vdash \mathtt{t} : \mathtt{T}$.
Proof By induction on the depth of the derivation of $\Gamma \vdash \mathtt{t} : \mathtt{T}$. Case analysis of the last rule in the derivation.

- Case T-Var: Then $\mathtt{t}$ has the form $\mathtt{y}$ and $\mathtt{y:T} \in \Gamma$. Since $\mathtt{x} \notin \mathrm{dom}(\Gamma)$, we have that $\mathtt{x} \neq \mathtt{y}$, so also $\mathtt{y:T} \in \Gamma\mathtt{,x:S}$. Therefore by T-Var we have $\Gamma\mathtt{,x:S} \vdash \mathtt{y} : \mathtt{T}$.

- Case T-Field: Then $\mathtt{t}$ has the form $\mathtt{s.f}_i$ and $\mathtt{T}$ has the form $\mathtt{T}_i$ and $\Gamma \vdash \mathtt{s} : \mathtt{C}$ and *fields*($\mathtt{C}$) $= \mathtt{\overline{T}\ \overline{f}}$. By induction we have $\Gamma\mathtt{,x:S} \vdash \mathtt{s} : \mathtt{C}$, so by T-Field also $\Gamma\mathtt{,x:S} \vdash \mathtt{s.f}_i : \mathtt{T}_i$.

- Case T-Invk: Then $\mathtt{t}$ has the form $\mathtt{t_0.m(\overline{t})}$ and $\Gamma \vdash \mathtt{t_0} : \mathtt{T_0}$ and *mtype*($\mathtt{m,T_0}$) $= \mathtt{\overline{T}}\rightarrow\mathtt{T}$ and $\Gamma \vdash \mathtt{\overline{t}} : \mathtt{\overline{S}}$ and $TT \vdash \mathtt{\overline{S}{<:}\overline{T}}$. By induction we have $\Gamma\mathtt{,x:S} \vdash \mathtt{t_0} : \mathtt{T_0}$ and $\Gamma\mathtt{,x:S} \vdash \mathtt{\overline{t}} : \mathtt{\overline{S}}$, so by T-Invk also $\Gamma\mathtt{,x:S} \vdash \mathtt{t_0.m(\overline{t})} : \mathtt{T}$.

- Case T-New: Then $\mathtt{t}$ has the form $\mathtt{new\ C(\overline{t})}$ and $\mathtt{T} = \mathtt{C}$ and *fields*($\mathtt{C}$) $= \mathtt{\overline{T}\ \overline{f}}$ and $\Gamma \vdash \mathtt{\overline{t}} : \mathtt{\overline{S}}$ and $TT \vdash \mathtt{\overline{S}{<:}\overline{T}}$. By induction we have $\Gamma\mathtt{,x:S} \vdash \mathtt{\overline{t}} : \mathtt{\overline{S}}$, so by T-New also $\Gamma\mathtt{,x:S} \vdash \mathtt{new\ C(\overline{t})} : \mathtt{C}$.

- Case T-UCast: Then $\mathtt{t}$ has the form $\mathtt{(T)t_0}$ and $\Gamma \vdash \mathtt{t_0} : \mathtt{T_0}$ and $TT \vdash \mathtt{T_0{<:}T}$. By induction we have $\Gamma\mathtt{,x:S} \vdash \mathtt{t_0} : \mathtt{T_0}$, so by T-UCast also $\Gamma\mathtt{,x:S} \vdash \mathtt{(T)t_0} : \mathtt{T}$.

- Case T-DCast: Then $\mathtt{t}$ has the form $\mathtt{(T)t_0}$ and $\Gamma \vdash \mathtt{t_0} : \mathtt{T_0}$ and $TT \vdash \mathtt{T{<:}T_0}$ and $\mathtt{T} \neq \mathtt{T_0}$. By induction we have $\Gamma\mathtt{,x:S} \vdash \mathtt{t_0} : \mathtt{T_0}$, so by T-DCast also $\Gamma\mathtt{,x:S} \vdash \mathtt{(T)t_0} : \mathtt{T}$.

- Case T-SCast: Then $\mathtt{t}$ has the form $\mathtt{(T)t_0}$ and $\Gamma \vdash \mathtt{t_0} : \mathtt{T_0}$ and $TT \vdash \mathtt{T_0}\not<\mathtt{:T}$ and $TT \vdash \mathtt{T}\not<\mathtt{:T_0}$ and a *stupid warning* is generated. By induction we have $\Gamma\mathtt{,x:S} \vdash \mathtt{t_0} : \mathtt{T_0}$, so by T-SCast also $\Gamma\mathtt{,x:S} \vdash \mathtt{(T)t_0} : \mathtt{T}$.

□

Lemma 6.7 If *mbody*($\mathtt{m}$,$\mathtt{C}$,$\mathtt{\overline{D}}$) = ($\mathtt{\overline{x}}$, $\mathtt{t}$) then there exist $\mathtt{\overline{T}}$ and $\mathtt{T}$ such that *mtype*($\mathtt{m}$,$\mathtt{C}$) $= \mathtt{\overline{T}}\rightarrow\mathtt{T}$.
Proof By induction on the depth of the derivation of *mbody*($\mathtt{m}$,$\mathtt{C}$,$\mathtt{\overline{D}}$) = ($\mathtt{\overline{x}}$, $\mathtt{t}$). Case analysis of the last rule in the derivation.

- Case MBody1: Then $TT(\mathtt{C})$ = $\mathtt{class\ C\ extends\ E\ implements\ \overline{I}}$ $\{\mathtt{\overline{S}\ \overline{f};\ K\ \overline{M}}\}$ and $\mathtt{U\ m(\overline{U}\ \overline{x})\ when\ \overline{P}}$ $\{\mathtt{return\ \overline{t};}\} \in \mathtt{\overline{M}}$, and the result follows by MType-C1.

- Case MBody2: Then $TT(\mathtt{C})$ = $\mathtt{class\ C\ extends\ E\ implements\ \overline{I}}$ $\{\mathtt{\overline{S}\ \overline{f};\ K\ \overline{M}}\}$ and $\mathtt{U\ m(\overline{U}\ \overline{x})\ when\ \overline{P}}$ $\{\mathtt{return\ \overline{t};}\} \in \mathtt{\overline{M}}$, and the result follows by MType-C1.

- Case MBody3: Then $TT(\mathtt{C})$ = $\mathtt{class\ C\ extends\ E\ implements\ \overline{I}}$ $\{\mathtt{\overline{S}\ \overline{f};\ K\ \overline{M}}\}$ and $\mathtt{m}$ is not defined in $\mathtt{\overline{M}}$ and *mbody*($\mathtt{m}$,$\mathtt{C}$,$\mathtt{\overline{D}}$) = *mbody*($\mathtt{m}$,$\mathtt{E}$,$\mathtt{\overline{D}}$). By induction there exist $\mathtt{\overline{T}}$ and $\mathtt{T}$ such that *mtype*($\mathtt{m}$,$\mathtt{E}$) $= \mathtt{\overline{T}}\rightarrow\mathtt{T}$, and the result follows by MType-C2.

□

Lemma 6.8 If *mbody*($\mathtt{m}$,$\mathtt{C}$,$\overline{\mathtt{D}}$) = ($\overline{\mathtt{x}}$, $\mathtt{t}$) and *mtype*($\mathtt{m}$,$\mathtt{C}$) = $\overline{\mathtt{T}}\to\mathtt{T}$, then there exists a class $\mathtt{D}$ and a type $\mathtt{S}$ such that $TT \vdash \mathtt{C}\texttt{<:}\mathtt{D}$ and $TT \vdash \mathtt{S}\texttt{<:}\mathtt{T}$ and $\overline{\mathtt{x}}\texttt{:}\overline{\mathtt{T}}\texttt{,this:}\mathtt{D} \vdash \mathtt{t} : \mathtt{S}$.
Proof By induction on the depth of the derivation of *mbody*($\mathtt{m}$,$\mathtt{C}$,$\overline{\mathtt{D}}$) = ($\overline{\mathtt{x}}$, $\mathtt{t}$). Case analysis of the last rule in the derivation.

- Case MBody1: Then $\mathtt{t} = \mathtt{t}_i$ and $TT(\mathtt{C})$ = `class C extends E implements` $\overline{\mathtt{I}}$ $\{\overline{\mathtt{S}}\ \overline{\mathtt{f}}\texttt{;}\ \mathtt{K}\ \overline{\mathtt{M}}\}$ and $\mathtt{U}\ \mathtt{m}(\overline{\mathtt{U}}\ \overline{\mathtt{x}})$ **`when`** $\overline{\mathtt{P}}$ $\{$**`return`** $\overline{\mathtt{t}}\texttt{;}\} \in \overline{\mathtt{M}}$. Since *mtype*($\mathtt{m}$,$\mathtt{C}$) = $\overline{\mathtt{T}}\to\mathtt{T}$, by MType-C1 we have that $\overline{\mathtt{U}} = \overline{\mathtt{T}}$ and $\mathtt{U} = \mathtt{T}$. By T-Class we have $\overline{\mathtt{M}}$ `OK in C`, so by T-Meth we have $\overline{\mathtt{x}}\texttt{:}\overline{\mathtt{T}}\texttt{,this:}\mathtt{C} \vdash \mathtt{t} : \mathtt{S}$ and $TT \vdash \mathtt{S}\texttt{<:}\mathtt{T}$. Finally, by S-Ref we have $TT \vdash \mathtt{C}\texttt{<:}\mathtt{C}$.

- Case MBody2: Then $TT(\mathtt{C})$ = `class C extends E implements` $\overline{\mathtt{I}}$ $\{\overline{\mathtt{S}}\ \overline{\mathtt{f}}\texttt{;}\ \mathtt{K}\ \overline{\mathtt{M}}\}$ and $\mathtt{U}\ \mathtt{m}(\overline{\mathtt{U}}\ \overline{\mathtt{x}})$ **`when`** $\overline{\mathtt{P}}$ $\{$**`return`** $\overline{\mathtt{t}}\texttt{;}\} \in \overline{\mathtt{M}}$ and *mbody*($\mathtt{m}$,$\mathtt{C}$,$\overline{\mathtt{D}}$) = *mbody*($\mathtt{m}$,$\mathtt{E}$,$\overline{\mathtt{D}}$). Since *mtype*($\mathtt{m}$,$\mathtt{C}$) = $\overline{\mathtt{T}}\to\mathtt{T}$, by MType-C1 we have that $\overline{\mathtt{U}} = \overline{\mathtt{T}}$ and $\mathtt{U} = \mathtt{T}$. By Lemma 6.7 there exist $\overline{\mathtt{T}_0}$ and $\mathtt{T}_0$ such that *mtype*($\mathtt{m}$,$\mathtt{E}$) = $\overline{\mathtt{T}_0}\to\mathtt{T}_0$. By S-CExt we have $TT \vdash \mathtt{C}\texttt{<:}\mathtt{E}$, so by Lemma 6.4 we have $\overline{\mathtt{T}_0} = \overline{\mathtt{T}}$ and $\mathtt{T}_0 = \mathtt{T}$. Therefore, by induction there exists a class $\mathtt{D}$ and a type $\mathtt{S}$ such that $TT \vdash \mathtt{E}\texttt{<:}\mathtt{D}$ and $TT \vdash \mathtt{S}\texttt{<:}\mathtt{T}$ and $\overline{\mathtt{x}}\texttt{:}\overline{\mathtt{T}}\texttt{,this:}\mathtt{D} \vdash \mathtt{t} : \mathtt{S}$. By S-Trans we have $TT \vdash \mathtt{C}\texttt{<:}\mathtt{D}$, so the result follows.

- Case MBody3: Then $TT(\mathtt{C})$ = `class C extends E implements` $\overline{\mathtt{I}}$ $\{\overline{\mathtt{S}}\ \overline{\mathtt{f}}\texttt{;}\ \mathtt{K}\ \overline{\mathtt{M}}\}$ and $\mathtt{m}$ is not defined in $\overline{\mathtt{M}}$ and *mbody*($\mathtt{m}$,$\mathtt{C}$,$\overline{\mathtt{D}}$) = *mbody*($\mathtt{m}$,$\mathtt{E}$,$\overline{\mathtt{D}}$). Since *mtype*($\mathtt{m}$,$\mathtt{C}$) = $\overline{\mathtt{T}}\to\mathtt{T}$, by MType-C2 we have that *mtype*($\mathtt{m}$,$\mathtt{E}$) = $\overline{\mathtt{T}}\to\mathtt{T}$ as well. Therefore, by induction there exists a class $\mathtt{D}$ and a type $\mathtt{S}$ such that $TT \vdash \mathtt{E}\texttt{<:}\mathtt{D}$ and $TT \vdash \mathtt{S}\texttt{<:}\mathtt{T}$ and $\overline{\mathtt{x}}\texttt{:}\overline{\mathtt{T}}\texttt{,this:}\mathtt{D} \vdash \mathtt{t} : \mathtt{S}$. By S-Trans we have $TT \vdash \mathtt{C}\texttt{<:}\mathtt{D}$, so the result follows.

□

Theorem 6.1 (Subject Reduction) If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{s}$, then there exists some type $\mathtt{S}$ such that $\Gamma \vdash \mathtt{s} : \mathtt{S}$ and $TT \vdash \mathtt{S}\texttt{<:}\mathtt{T}$.
Proof By induction on the depth of the derivation of $\mathtt{t} \longrightarrow \mathtt{s}$. Case analysis of the last rule in the derivation.

- Case E-ProjNew: Then $\mathtt{t}$ has the form $\texttt{(new C(}\overline{\mathtt{v}}\texttt{)).f}_i$ and $\mathtt{s}$ has the form $\mathtt{v}_i$ and *fields*($\mathtt{C}$) = $\overline{\mathtt{T}}\ \overline{\mathtt{f}}$. Since $\Gamma \vdash \mathtt{t} : \mathtt{T}$, by T-Field and T-New we have that $\Gamma \vdash \texttt{new C(}\overline{\mathtt{v}}\texttt{)} : \mathtt{C}$ and $\Gamma \vdash \mathtt{v}_i : \mathtt{S}_i$ and $TT \vdash \mathtt{S}_i\texttt{<:}\mathtt{T}_i$ and $\mathtt{T} = \mathtt{T}_i$, so the result follows.

- Case E-InvkNew: Then $\mathtt{t}$ has the form $\texttt{new C(}\overline{\mathtt{v}}\texttt{).m(}\overline{\mathtt{u}}\texttt{)}$ and $\mathtt{s}$ has the form $[\overline{\mathtt{x}} \mapsto \overline{\mathtt{u}}\texttt{,this} \mapsto \texttt{new C(}\overline{\mathtt{v}}\texttt{)}]\mathtt{t}_0$ and $\overline{\mathtt{u}} = \texttt{new}\ \overline{\mathtt{D}}(\ldots)$ and *mbody*($\mathtt{m}$,$\mathtt{C}$,$\overline{\mathtt{D}}$) = ($\overline{\mathtt{x}}$, $\mathtt{t}_0$). Since $\Gamma \vdash \mathtt{t} : \mathtt{T}$, by T-Invk we have $\Gamma \vdash \texttt{new C(}\overline{\mathtt{v}}\texttt{)} : \mathtt{S}'$ and *mtype*($\mathtt{m}$,$\mathtt{S}'$) = $\overline{\mathtt{T}}\to\mathtt{T}$ and $\Gamma \vdash \overline{\mathtt{u}} : \overline{\mathtt{S}}$ and $TT \vdash \overline{\mathtt{S}}\texttt{<:}\overline{\mathtt{T}}$. By T-New we have that $\mathtt{S}' = \mathtt{C}$. Therefore by Lemma 6.8 there exists a class $\mathtt{D}$ and a type $\mathtt{U}$ such that $TT \vdash \mathtt{C}\texttt{<:}\mathtt{D}$ and $TT \vdash \mathtt{U}\texttt{<:}\mathtt{T}$ and $\overline{\mathtt{x}}\texttt{:}\overline{\mathtt{T}}\texttt{,this:}\mathtt{D} \vdash \mathtt{t}_0 : \mathtt{U}$. Then by Lemma 6.5 we have that $\bullet \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{u}}\texttt{,this} \mapsto \texttt{new C(}\overline{\mathtt{v}}\texttt{)}]\mathtt{t}_0 : \mathtt{S}$, where $TT \vdash \mathtt{S}\texttt{<:}\mathtt{U}$. Then by Lemma 6.6 also $\Gamma \vdash [\overline{\mathtt{x}} \mapsto \overline{\mathtt{u}}\texttt{,this} \mapsto \texttt{new C(}\overline{\mathtt{v}}\texttt{)}]\mathtt{t}_0 : \mathtt{S}$. Finally, by S-Trans we have $TT \vdash \mathtt{S}\texttt{<:}\mathtt{T}$.

- Case E-CastNew: Then $\mathtt{t}$ has the form $(\mathtt{T}_0)\texttt{(new C(}\overline{\mathtt{v}}\texttt{))}$ and $\mathtt{s}$ has the form $\texttt{new C(}\overline{\mathtt{v}}\texttt{)}$ and $TT \vdash \mathtt{C}\texttt{<:}\mathtt{T}_0$. There are three subcases, depending on the last rule in the derivation of $\Gamma \vdash \mathtt{t} : \mathtt{T}$.

  – Case T-UCast: Then $\Gamma \vdash \texttt{new C(}\overline{\mathtt{v}}\texttt{)} : \mathtt{S}_0$ and $TT \vdash \mathtt{S}_0\texttt{<:}\mathtt{T}_0$ and $\mathtt{T} = \mathtt{T}_0$.

  - Case T-DCast: Then $\Gamma \vdash \texttt{new C}(\overline{\texttt{v}}) : \texttt{S}_0$ and $TT \vdash \texttt{T}_0\texttt{<:}\texttt{S}_0$ and $\texttt{T}_0 \neq \texttt{S}_0$ and $\texttt{T} = \texttt{T}_0$. Then by T-New we have that $\texttt{S}_0 = \texttt{C}$, but then $TT \vdash \texttt{T}_0\texttt{<:C}$ and $\texttt{T}_0 \neq \texttt{C}$, which contradicts the fact that $TT \vdash \texttt{C<:T}_0$ (by the assumption that the subtyping relation has no nontrivial cycles). Therefore, T-DCast cannot be the last rule used in the derivation.
  - Case T-SCast: Then $\Gamma \vdash \texttt{new C}(\overline{\texttt{v}}) : \texttt{S}_0$ and $TT \vdash \texttt{S}_0\not\texttt{<:}\texttt{T}_0$ and $TT \vdash \texttt{T}_0\not\texttt{<:}\texttt{S}_0$ and $\texttt{T} = \texttt{T}_0$. Then by T-New we have that $\texttt{S}_0 = \texttt{C}$, but then $TT \vdash \texttt{C}\not\texttt{<:}\texttt{T}_0$, which contradicts the fact that $TT \vdash \texttt{C<:T}_0$. Therefore, T-SCast cannot be the last rule used in the derivation.

- Case E-Field: Then $\texttt{t}$ has the form $\texttt{t}_1\texttt{.f}$ and $\texttt{s}$ has the form $\texttt{t}_2\texttt{.f}$ and $\texttt{t}_1 \longrightarrow \texttt{t}_2$. Since $\Gamma \vdash \texttt{t} : \texttt{T}$, by T-Field we have that $\texttt{f}$ has the form $\texttt{f}_i$ and $\texttt{T}$ has the form $\texttt{T}_i$ and $\Gamma \vdash \texttt{t}_1 : \texttt{C}$ and $\textit{fields}(\texttt{C}) = \overline{\texttt{T}}\ \overline{\texttt{f}}$. By induction, there exists some type $\texttt{T}_0$ such that $\Gamma \vdash \texttt{t}_2 : \texttt{T}_0$ and $TT \vdash \texttt{T}_0\texttt{<:C}$. Then by Lemma 6.1 $\texttt{T}_0$ is some class $\texttt{D}$, and by Lemma 6.2 we have $\textit{fields}(\texttt{C}) \subseteq \textit{fields}(\texttt{D})$. Then by T-Field we have $\Gamma \vdash \texttt{t}_2\texttt{.f} : \texttt{T}$ and by S-Ref we have $TT \vdash \texttt{T<:T}$.

- Case E-Invk-Recv: Then $\texttt{t}$ has the form $\texttt{s}_1\texttt{.m}(\overline{\texttt{t}})$ and $\texttt{s}$ has the form $\texttt{s}_2\texttt{.m}(\overline{\texttt{t}})$ and $\texttt{s}_1 \longrightarrow \texttt{s}_2$. Since $\Gamma \vdash \texttt{t} : \texttt{T}$, by T-Invk we have $\Gamma \vdash \texttt{s}_1 : \texttt{S}'$ and $\textit{mtype}(\texttt{m},\texttt{S}') = \overline{\texttt{T}}\rightarrow\texttt{T}$ and $\Gamma \vdash \overline{\texttt{t}} : \overline{\texttt{S}}$ and $TT \vdash \overline{\texttt{S}}\texttt{<:}\overline{\texttt{T}}$. By induction we have $\Gamma \vdash \texttt{s}_2 : \texttt{S}''$ and $TT \vdash \texttt{S}''\texttt{<:}\texttt{S}'$. Then by Lemma 6.4 we have $\textit{mtype}(\texttt{m},\texttt{S}'') = \overline{\texttt{T}}\rightarrow\texttt{T}$. Then by T-Invk we have $\Gamma \vdash \texttt{s}_2\texttt{.m}(\overline{\texttt{t}}) : \texttt{T}$ and by S-Ref we have $TT \vdash \texttt{T<:T}$.

- Case E-Invk-Arg: Then $\texttt{t}$ has the form $\texttt{v.m}(\overline{\texttt{v}},\texttt{s}_1,\overline{\texttt{s}_0})$ and $\texttt{s}$ has the form $\texttt{v.m}(\overline{\texttt{v}},\texttt{s}_2,\overline{\texttt{s}_0})$ and $\texttt{s}_1 \longrightarrow \texttt{s}_2$. Since $\Gamma \vdash \texttt{t} : \texttt{T}$, by T-Invk we have $\Gamma \vdash \texttt{v} : \texttt{S}'$ and $\textit{mtype}(\texttt{m},\texttt{S}') = \overline{\texttt{T}}\rightarrow\texttt{T}$ and $\overline{\texttt{v}},\texttt{s}_1,\overline{\texttt{s}_0} = \overline{\texttt{t}}$ and $\Gamma \vdash \overline{\texttt{t}} : \overline{\texttt{S}}$ and $TT \vdash \overline{\texttt{S}}\texttt{<:}\overline{\texttt{T}}$. Assume that $\texttt{s}_1$ is the ith element of $\overline{\texttt{t}}$. By induction we have that $\Gamma \vdash \texttt{s}_2 : \texttt{S}'_i$ and $TT \vdash \texttt{S}'_i\texttt{<:}\texttt{S}_i$. Then by S-Trans also $TT \vdash \texttt{S}'_i\texttt{<:}\texttt{T}_i$, so by T-Invk we have $\Gamma \vdash \texttt{v.m}(\overline{\texttt{v}},\texttt{s}_2,\overline{\texttt{s}_0}) : \texttt{T}$ and by S-Ref we have $TT \vdash \texttt{T<:T}$.

- Case E-New-Arg: Then $\texttt{t}$ has the form $\texttt{new C}(\overline{\texttt{v}},\texttt{s}_1,\overline{\texttt{s}_0})$ and $\texttt{s}$ has the form $\texttt{new C}(\overline{\texttt{v}},\texttt{s}_2,\overline{\texttt{s}_0})$ and $\texttt{s}_1 \longrightarrow \texttt{s}_2$. Since $\Gamma \vdash \texttt{t} : \texttt{T}$, by T-New we have $\textit{fields}(\texttt{C}) = \overline{\texttt{T}}\ \overline{\texttt{f}}$ and $\overline{\texttt{v}},\texttt{s}_1,\overline{\texttt{s}_0} = \overline{\texttt{t}}$ and $\Gamma \vdash \overline{\texttt{t}} : \overline{\texttt{S}}$ and $TT \vdash \overline{\texttt{S}}\texttt{<:}\overline{\texttt{T}}$ and $\texttt{T} = \texttt{C}$. Assume that $\texttt{s}_1$ is the ith element of $\overline{\texttt{t}}$. By induction we have that $\Gamma \vdash \texttt{s}_2 : \texttt{S}'_i$ and $TT \vdash \texttt{S}'_i\texttt{<:}\texttt{S}_i$. Then by S-Trans also $TT \vdash \texttt{S}'_i\texttt{<:}\texttt{T}_i$, so by T-New we have $\Gamma \vdash \texttt{new C}(\overline{\texttt{v}},\texttt{s}_2,\overline{\texttt{s}_0}) : \texttt{C}$ and by S-Ref we have $TT \vdash \texttt{C<:C}$.

- Case E-Cast: Then $\texttt{t}$ has the form $(\texttt{T}_0)\texttt{s}_1$ and $\texttt{s}$ has the form $(\texttt{T}_0)\texttt{s}_2$ $\texttt{s}_1 \longrightarrow \texttt{s}_2$. There are three subcases, depending on the last rule in the derivation of $\Gamma \vdash \texttt{t} : \texttt{T}$.
  - Case T-UCast: Then $\Gamma \vdash \texttt{s}_1 : \texttt{S}_0$ and $TT \vdash \texttt{S}_0\texttt{<:}\texttt{T}_0$ and $\texttt{T} = \texttt{T}_0$. By induction we have $\Gamma \vdash \texttt{s}_2 : \texttt{S}'_0$ and $TT \vdash \texttt{S}'_0\texttt{<:}\texttt{S}_0$. Then by S-Trans also $TT \vdash \texttt{S}'_0\texttt{<:}\texttt{T}_0$, so by T-UCast we have $\Gamma \vdash (\texttt{T}_0)\texttt{s}_2 : \texttt{T}_0$ and by S-Ref we have $TT \vdash \texttt{T}_0\texttt{<:}\texttt{T}_0$.
  - Case T-DCast: Then $\Gamma \vdash \texttt{s}_1 : \texttt{S}_0$ and $TT \vdash \texttt{T}_0\texttt{<:}\texttt{S}_0$ and $\texttt{T}_0 \neq \texttt{S}_0$ and and $\texttt{T} = \texttt{T}_0$. By induction we have $\Gamma \vdash \texttt{s}_2 : \texttt{S}'_0$ and $TT \vdash \texttt{S}'_0\texttt{<:}\texttt{S}_0$. If $TT \vdash \texttt{S}'_0\texttt{<:}\texttt{T}_0$ then by T-UCast we have $\Gamma \vdash (\texttt{T}_0)\texttt{s}_2 : \texttt{T}_0$. Otherwise, if $TT \vdash \texttt{T}_0\texttt{<:}\texttt{S}'_0$ then by T-DCast we have $\Gamma \vdash (\texttt{T}_0)\texttt{s}_2 : \texttt{T}_0$. Otherwise we have $TT \vdash \texttt{S}'_0\not\texttt{<:}\texttt{T}_0$ and $TT \vdash \texttt{T}_0\not\texttt{<:}\texttt{S}'_0$, so by T-SCast we have $\Gamma \vdash (\texttt{T}_0)\texttt{s}_2 : \texttt{T}_0$ along with the generation of a *stupid warning*. Finally, by S-Ref we have $TT \vdash \texttt{T}_0\texttt{<:}\texttt{T}_0$.
  - Case T-SCast: Then $\Gamma \vdash \texttt{s}_1 : \texttt{S}_0$ and $TT \vdash \texttt{S}_0\not\texttt{<:}\texttt{T}_0$ and $TT \vdash \texttt{T}_0\not\texttt{<:}\texttt{S}_0$ and a *stupid warning* is generated and $\texttt{T} = \texttt{T}_0$. By induction we have $\Gamma \vdash \texttt{s}_2 : \texttt{S}'_0$ and $TT \vdash \texttt{S}'_0\texttt{<:}\texttt{S}_0$. If $TT \vdash \texttt{S}'_0\texttt{<:}\texttt{T}_0$ then by T-UCast we have $\Gamma \vdash (\texttt{T}_0)\texttt{s}_2 : \texttt{T}_0$. Otherwise, if $TT \vdash \texttt{T}_0\texttt{<:}\texttt{S}'_0$ then by S-Trans also $TT \vdash \texttt{T}_0\texttt{<:}\texttt{S}_0$, contradicting the fact that $TT \vdash \texttt{T}_0\not\texttt{<:}\texttt{S}_0$, so it is not possible that $TT \vdash \texttt{T}_0\texttt{<:}\texttt{S}'_0$. Otherwise we have $TT \vdash \texttt{S}'_0\not\texttt{<:}\texttt{T}_0$ and $TT \vdash \texttt{T}_0\not\texttt{<:}\texttt{S}'_0$, so by T-SCast we have $\Gamma \vdash (\texttt{T}_0)\texttt{s}_2 : \texttt{T}_0$. Finally, by S-Ref we have $TT \vdash \texttt{T}_0\texttt{<:}\texttt{T}_0$.

□

6.2 Progress

Definition 6.1 We say that a class $\mathtt{C}$ *covers* $\overline{\mathtt{D}}$ *for* $\mathtt{m}$ *via* ($\mathtt{P}$,$\overline{\mathtt{P}}$,$\overline{\mathtt{x}}$) if the following conditions hold:

1. $TT(\mathtt{C}) = \mathtt{class\ C} \cdots \{\cdots\ \overline{\mathtt{M}}\}$
2. $\mathtt{S\ m(\overline{S}\ \overline{x})\ when\ \overline{P}}\ \{\cdots\} \in \overline{\mathtt{M}}$
3. $\mathtt{P} \in \overline{\mathtt{P}}$
4. $TT;\ \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \mathtt{P}$

Lemma 6.9 $TT \supseteq$ *restrict*$(TT, \mathtt{P})$
Proof By RESTRICT *restrict*$(TT,\mathtt{P})$ has the form $(\overline{\mathtt{T}}, TT(\overline{\mathtt{T}}))$, so the result follows. □

Lemma 6.10 If $TT;\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \bigvee \overline{\mathtt{P}}$ then there is some $\mathtt{P} \in \overline{\mathtt{P}}$ such that $TT;\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \mathtt{P}$.
Proof By induction on the length of $\overline{\mathtt{P}}$.

- Case the length of $\overline{\mathtt{P}}$ is 0. Then $\overline{\mathtt{P}} = \bullet$, so by definition $\bigvee \overline{\mathtt{P}}$ is $\neg\mathtt{true}$. Since $TT;\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \bigvee \overline{\mathtt{P}}$, by P-NOT we have $TT;\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \not\models \mathtt{true}$. But by P-TRUE also $TT;\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \mathtt{true}$, so we have a contradiction. Therefore, the length of $\overline{\mathtt{P}}$ cannot be 0.
- Case the length of $\overline{\mathtt{P}}$ is 1. Then $\overline{\mathtt{P}}$ is some predicate $\mathtt{P}$, and also $\bigvee \overline{\mathtt{P}}$ is $\mathtt{P}$. Since $TT;\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \bigvee \overline{\mathtt{P}}$, we have $TT;\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \mathtt{P}$, so the result follows.
- Case the length of $\overline{\mathtt{P}}$ is some integer $d > 1$. Then $\overline{\mathtt{P}}$ can be expressed as $\mathtt{Q},\overline{\mathtt{Q}}$ where $\overline{\mathtt{Q}}$ is non-empty, and $\bigvee \overline{\mathtt{P}}$ can be expressed as $\mathtt{Q} \vee \bigvee \overline{\mathtt{Q}}$. Since $TT;\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \bigvee \overline{\mathtt{P}}$, by P-OR1 and P-OR2 we have that either $TT;\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \mathtt{Q}$ or $TT;\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \bigvee \overline{\mathtt{Q}}$. If the former, then the result follows with $\mathtt{P} = \mathtt{Q}$. If the latter, then by induction there is some $\mathtt{P} \in \overline{\mathtt{Q}}$ such that $TT;\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{D}} \models \mathtt{P}$. Then also $\mathtt{P} \in \overline{\mathtt{P}}$, so the result follows.

□

Lemma 6.11 If *mpreds*($\mathtt{m}$,$\mathtt{C}$) $= \overline{\mathtt{Q}}$ and $\mathtt{P} \in \overline{\mathtt{Q}}$, then there exists a class $\mathtt{D}$ such that $TT \vdash \mathtt{C<:D}$ and $TT(\mathtt{D}) = \mathtt{class\ D} \cdots \{\cdots\ \overline{\mathtt{M}}\}$ and $\mathtt{S\ m(\overline{S}\ \overline{y})\ when\ \overline{P}}\ \{\cdots\} \in \overline{\mathtt{M}}$ and $\mathtt{P} \in \overline{\mathtt{P}}$.
Proof By induction on the depth of the derivation of *mpreds*($\mathtt{m}$,$\mathtt{C}$) $= \overline{\mathtt{Q}}$. Case analysis of the last rule used in the derivation.

- Case MPREDS1. Then $\overline{\mathtt{Q}} = \bullet$, contradicting the fact that $\mathtt{P} \in \overline{\mathtt{Q}}$, so MPREDS1 cannot be the last rule in the derivation.
- Case MPREDS2. Then $\overline{\mathtt{Q}} = \overline{\mathtt{Q}}_1,\overline{\mathtt{Q}}_2$. Since $\mathtt{P} \in \overline{\mathtt{Q}}$, there are two subcases.
  - Case $\mathtt{P} \in \overline{\mathtt{Q}}_1$. By MPREDS2 we have $TT(\mathtt{C}) = \mathtt{class\ C} \cdots \{\cdots\ \overline{\mathtt{M}}\}$ and $\mathtt{S\ m(\overline{S}\ \overline{y})\ when}\ \overline{\mathtt{Q}}_1\ \{\cdots\} \in \overline{\mathtt{M}}$. Finally, by S-REF we have $TT \vdash \mathtt{C<:C}$. Therefore the result follows.
  - Case $\mathtt{P} \in \overline{\mathtt{Q}}_2$. By MPREDS2 we have $TT(\mathtt{C}) = \mathtt{class\ C\ extends\ E} \cdots \{\cdots\}$ and *mpreds*($\mathtt{m}$,$\mathtt{E}$) $= \overline{\mathtt{Q}}_2$. By induction there exists a class $\mathtt{D}$ such that $TT \vdash \mathtt{E<:D}$ and $TT(\mathtt{D}) = \mathtt{class\ D} \cdots \{\cdots\ \overline{\mathtt{M}}\}$ and $\mathtt{S\ m(\overline{S}\ \overline{y})\ when\ \overline{P}}\ \{\cdots\} \in \overline{\mathtt{M}}$ and $\mathtt{P} \in \overline{\mathtt{P}}$. Since $TT(\mathtt{C}) = \mathtt{class\ C\ extends\ E} \cdots \{\cdots\}$, by S-CEXT we have $TT \vdash \mathtt{C<:E}$. Then by S-TRANS also $TT \vdash \mathtt{C<:D}$, so the result follows.

□

Lemma 6.12 If *mformals*(`m`,`C`) $=$ $\overline{\texttt{x}}$ and *allMethodNames*(`C`) $=$ $\overline{\texttt{m}}$, then `m` $\in$ $\overline{\texttt{m}}$.
Proof By induction on the depth of the derivation of *mformals*(`m`,`C`) $=$ $\overline{\texttt{x}}$. Case analysis of the last rule in the derivation.

- Case MFormals1. Then TT(`C`) $=$ **`class C extends D implements`** $\overline{\texttt{I}}$ $\{\overline{\texttt{T}}$ $\overline{\texttt{f}}$; `K` $\overline{\texttt{M}}\}$ and `S m(`$\overline{\texttt{S}}$ $\overline{\texttt{x}}$`)` **`when`** $\overline{\texttt{P}}$ $\{$**`return`** $\overline{\texttt{t}}$;$\}$ $\in$ $\overline{\texttt{M}}$. Since *allMethodNames*(`C`) $=$ $\overline{\texttt{m}}$, by MNames-Cls we have *mname*($\overline{\texttt{M}}$) $\subseteq$ $\overline{\texttt{m}}$, and by MName-Cls that means `m` $\in$ $\overline{\texttt{m}}$.

- Case MFormals2. Then TT(`C`) $=$ **`class C extends D implements`** $\overline{\texttt{I}}$ $\{\overline{\texttt{T}}$ $\overline{\texttt{f}}$; `K` $\overline{\texttt{M}}\}$ and *mformals*(`m`,`D`) $=$ $\overline{\texttt{x}}$. By T-Class we have that *allMethodNames*(`D`) $=$ $\overline{\texttt{m}_0}$, so by induction we have `m` $\in$ $\overline{\texttt{m}_0}$. Then by MNames-Cls also `m` $\in$ $\overline{\texttt{m}}$.

□

Lemma 6.13 If $TT \vdash$ `C<:D` and *mformals*(`m`,`D`) $=$ $\overline{\texttt{x}}$, then *mformals*(`m`,`C`) $=$ $\overline{\texttt{x}}$.
Proof By induction on the depth of the derivation of $TT \vdash$ `C<:D`. Case analysis of the last rule in the derivation.

- Case S-Ref. Then `C` $=$ `D`, so the result follows.

- Case S-Trans. Then $TT \vdash$ `C<:T` and $TT \vdash$ `T<:D`. Then by Lemma 6.1 `T` is some class `C`$'$. Then by induction we have *mformals*(`m`,`C`$'$) $=$ $\overline{\texttt{x}}$, and by induction again we have *mformals*(`m`,`C`) $=$ $\overline{\texttt{x}}$.

- Case S-CExt. Then TT(`C`) $=$ **`class C extends D implements`** $\overline{\texttt{I}}$ $\{\overline{\texttt{T}}$ $\overline{\texttt{f}}$; `K` $\overline{\texttt{M}}\}$. Since *mformals*(`m`,`D`) $=$ $\overline{\texttt{x}}$, if `m` is not defined in $\overline{\texttt{M}}$ then by M-Formals2 also *mformals*(`m`,`C`) $=$ $\overline{\texttt{x}}$, so the result follows. Otherwise `m` is defined in $\overline{\texttt{M}}$, so by M-Formals1 there exists some $\overline{\texttt{y}}$ such that *mformals*(`m`,`C`) $=$ $\overline{\texttt{y}}$. By T-Class we have *allMethodNames*(`C`) $=$ $\overline{\texttt{m}}$, so by Lemma 6.12 we have `m` $\in$ $\overline{\texttt{m}}$. Then also by T-Class we have *override*(`m`,`C`), so by Over-Cls we have *override*(`m`, `D`, $\overline{\texttt{S}}\to$`S`, $\overline{\texttt{y}}$). Finally, by T-OverCls we have that $\overline{\texttt{x}}$ $=$ $\overline{\texttt{y}}$, so the result follows.

□

Lemma 6.14 If *mtype*(`m`,`C`) $=$ $\overline{\texttt{T}}\to$`T`, then there exists $\overline{\texttt{x}}$ of the same length as $\overline{\texttt{T}}$ such that *mformals*(`m`,`C`) $=$ $\overline{\texttt{x}}$.
Proof By induction on the depth of the derivation of *mtype*(`m`,`C`) $=$ $\overline{\texttt{T}}\to$`T`. Case analysis of the last rule in the derivation.

- Case MType-C1. Then TT(`C`) $=$ **`class C extends D implements`** $\overline{\texttt{I}}$ $\{\overline{\texttt{T}}$ $\overline{\texttt{f}}$; `K` $\overline{\texttt{M}}\}$ and `S m(`$\overline{\texttt{S}}$ $\overline{\texttt{x}}$`)` **`when`** $\overline{\texttt{P}}$ $\{$**`return`** $\overline{\texttt{t}}$;$\}$ $\in$ $\overline{\texttt{M}}$. Then the result follows by MFormals1.

- Case MType-C2. Then TT(`C`) $=$ **`class C extends D implements`** $\overline{\texttt{I}}$ $\{\overline{\texttt{T}}$ $\overline{\texttt{f}}$; `K` $\overline{\texttt{M}}\}$ and `m` is not defined in $\overline{\texttt{M}}$ and *mtype*(`m`,`D`) $=$ $\overline{\texttt{T}}\to$`T`. By induction there exists $\overline{\texttt{x}}$ of the same length as $\overline{\texttt{T}}$ such that *mformals*(`m`,`D`) $=$ $\overline{\texttt{x}}$, and the result follows by MFormals2.

□

Lemma 6.15 (Exhaustiveness) If *mtype*(`m`,`C`) $=$ $\overline{\texttt{T}}\to$`T` and *mformals*(`m`,`C`) $=$ $\overline{\texttt{x}}$ and $\overline{\texttt{D}}$ has the same length as $\overline{\texttt{x}}$ and $\overline{\texttt{D}}$ $\subseteq$ dom(TT), then there exist `D`, `P`, and $\overline{\texttt{P}}$ such that $TT \vdash$ `C<:D` and `D` covers $\overline{\texttt{D}}$ for `m` via (`P`,$\overline{\texttt{P}}$,$\overline{\texttt{x}}$).

Proof Since *mtype*($\mathtt{m}$,$\mathtt{C}$) = $\overline{\mathtt{T}}\rightarrow\mathtt{T}$, by MTYPE-C1 and MTYPE-C2 we have that $\mathtt{C} \in \mathrm{dom}(TT)$. Then by T-CLASS we have *allMethodNames*($\mathtt{C}$) = $\overline{\mathtt{m}}$ and *exhaustive*($\overline{\mathtt{m}}$,$\mathtt{C}$). By Lemma 6.3 we have $\mathtt{m} \in \overline{\mathtt{m}}$, so by T-EXHAUST we have *mpreds*($\mathtt{m}$,$\mathtt{C}$) = $\overline{\mathtt{Q}}$ and $\overline{\mathtt{x}} \models \bigvee \overline{\mathtt{Q}}$. By Lemma 6.9 we know $TT \supseteq$ *restrict*$(TT, \bigvee \overline{\mathtt{Q}})$, and we are given that $\overline{\mathtt{D}} \subseteq \mathrm{dom}(TT)$, so by VALID we have $TT; \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \bigvee \overline{\mathtt{Q}}$. Then by Lemma 6.10 there is some $\mathtt{P} \in \overline{\mathtt{Q}}$ such that $TT; \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{P}$. Then by Lemma 6.11 there exists a class $\mathtt{D}$ such that $TT \vdash \mathtt{C}{<:}\mathtt{D}$ and $TT(\mathtt{D}) = \mathtt{class\ D} \cdots \{\cdots\ \overline{\mathtt{M}}\}$ and $\mathtt{S\ m}(\overline{\mathtt{S}}\ \overline{\mathtt{y}})\ \mathtt{when}\ \overline{\mathtt{P}}\ \{\cdots\} \in \overline{\mathtt{M}}$ and $\mathtt{P} \in \overline{\mathtt{P}}$. Then by MFORMALS1 we have *mformals*($\mathtt{m}$,$\mathtt{D}$) = $\overline{\mathtt{y}}$, and by Lemma 6.13 we have $\overline{\mathtt{x}} = \overline{\mathtt{y}}$. Finally, by Definition 6.1 we have shown that $\mathtt{D}$ covers $\overline{\mathtt{D}}$ for $\mathtt{m}$ via ($\mathtt{P}$,$\overline{\mathtt{P}}$,$\overline{\mathtt{x}}$). □

Lemma 6.16 $TT'; \Gamma \models \mathtt{P} \Rightarrow \mathtt{P}$
Proof By definition of $\Rightarrow$ we must prove that $TT'; \Gamma \models \neg\mathtt{P} \vee \mathtt{P}$. We have two subcases.

- Case $TT'; \Gamma \models \mathtt{P}$. Then the result follows by P-OR2.
- Case $TT'; \Gamma \not\models \mathtt{P}$. Then by P-NOT also $TT'; \Gamma \models \neg\mathtt{P}$, and the result follows by P-OR1.

□

Lemma 6.17 If $TT'; \Gamma \models \mathtt{P} \Rightarrow \mathtt{Q}$ and $TT'; \Gamma \models \mathtt{Q} \Rightarrow \mathtt{R}$, then $TT'; \Gamma \models \mathtt{P} \Rightarrow \mathtt{R}$.
Proof By definition of $\Rightarrow$ we have $TT'; \Gamma \models \neg\mathtt{P} \vee \mathtt{Q}$ and $TT'; \Gamma \models \neg\mathtt{Q} \vee \mathtt{R}$, and we must prove that $TT'; \Gamma \models \neg\mathtt{P} \vee \mathtt{R}$. Since $TT'; \Gamma \models \neg\mathtt{P} \vee \mathtt{Q}$, by P-OR1 and P-OR2 we have two subcases.

- Case $TT'; \Gamma \models \neg\mathtt{P}$. Then by P-OR1 also $TT'; \Gamma \models \neg\mathtt{P} \vee \mathtt{R}$.
- Case $TT'; \Gamma \models \mathtt{Q}$. Since $TT'; \Gamma \models \neg\mathtt{Q} \vee \mathtt{R}$, by P-OR1 and P-OR2 we have two subcases.
  - Case $TT'; \Gamma \models \neg\mathtt{Q}$. Then by P-NOT also $TT'; \Gamma \not\models \mathtt{Q}$, so we have a contradiction.
  - Case $TT'; \Gamma \models \mathtt{R}$. Then by P-OR2 also $TT'; \Gamma \models \neg\mathtt{P} \vee \mathtt{R}$.

□

Lemma 6.18 If $\mathtt{P}{\preceq_{\overline{\mathtt{x}}}}\mathtt{Q}$ and $\mathtt{Q}{\preceq_{\overline{\mathtt{x}}}}\mathtt{R}$, then $\mathtt{P}{\preceq_{\overline{\mathtt{x}}}}\mathtt{R}$.
Proof By MORESPECIFIC we have $\overline{\mathtt{x}} \models \mathtt{P}{\Rightarrow}\mathtt{Q}$ and $\overline{\mathtt{x}} \models \mathtt{Q}{\Rightarrow}\mathtt{R}$. Then by VALID we have that for all $TT' \supseteq$ *restrict*$(TT, \mathtt{P})$, for all $\overline{\mathtt{D}} \subseteq \mathrm{dom}(TT')$ of the same length as $\overline{\mathtt{x}}$, $TT'; \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{P}{\Rightarrow}\mathtt{Q}$. Similarly, for all $TT' \supseteq$ *restrict*$(TT, \mathtt{P})$, for all $\overline{\mathtt{D}} \subseteq \mathrm{dom}(TT')$ of the same length as $\overline{\mathtt{x}}$, $TT'; \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{Q}{\Rightarrow}\mathtt{R}$. Consider some $TT' \supseteq$ *restrict*$(TT, \mathtt{P})$ and $\overline{\mathtt{D}} \subseteq \mathrm{dom}(TT')$ of the same length as $\overline{\mathtt{x}}$. So we have $TT'; \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{P}{\Rightarrow}\mathtt{Q}$ and $TT'; \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{Q}{\Rightarrow}\mathtt{R}$. Then by Lemma 6.17 also $TT'; \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{P}{\Rightarrow}\mathtt{R}$. Therefore we have shown that for all $TT' \supseteq$ *restrict*$(TT, \mathtt{P})$, for all $\overline{\mathtt{D}} \subseteq \mathrm{dom}(TT')$ of the same length as $\overline{\mathtt{x}}$, $TT'; \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{P}{\Rightarrow}\mathtt{R}$, so by VALID we have $\overline{\mathtt{x}} \models \mathtt{P}{\Rightarrow}\mathtt{R}$. Finally, the result follows by MORESPECIFIC. □

Lemma 6.19 (Unambiguity) If $\overline{\mathtt{D}} \subseteq \mathrm{dom}(TT)$ and $\mathtt{D}$ covers $\overline{\mathtt{D}}$ for $\mathtt{m}$ via ($\mathtt{P}$,$\overline{\mathtt{P}}$,$\overline{\mathtt{x}}$) then there exists $\mathtt{Q} \in \overline{\mathtt{P}}$ such that $TT;\ \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{Q}$ and *overridesIfApplicable*($\mathtt{Q}$,$\overline{\mathtt{P}}$,$\overline{\mathtt{x}}$,$\overline{\mathtt{D}}$).
Proof By Definition 6.1 we have $TT(\mathtt{D}) = \mathtt{class\ D} \cdots \{\cdots\ \overline{\mathtt{M}}\}$ and $\mathtt{S\ m}(\overline{\mathtt{S}}\ \overline{\mathtt{x}})\ \mathtt{when}\ \overline{\mathtt{P}}\ \{\cdots\} \in \overline{\mathtt{M}}$ and $\mathtt{P} \in \overline{\mathtt{P}}$ and $TT;\ \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{P}$. We prove this lemma by induction on the number of predicates $\mathtt{P}' \in \overline{\mathtt{P}}$ such that $TT;\ \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{P}'$ and $\mathtt{P}{\not\prec_{\overline{\mathtt{x}}}}\mathtt{P}'$.

- Case there are zero such predicates. By Lemma 6.16 we have $TT'; \Gamma \models \mathtt{P} \Rightarrow \mathtt{P}$ for all TT' and Γ. Then by VALID we have $\overline{\mathtt{x}} \models \mathtt{P} \Rightarrow \mathtt{P}$, so by MORESPECIFIC also $\mathtt{P}{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}$. Therefore by definition of $\prec$ also $\mathtt{P}{\not\prec_{\overline{\mathtt{x}}}}\mathtt{P}$. Since $TT;\ \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{P}$, we have found a $\mathtt{Q}$, namely $\mathtt{P}$ itself, such that $TT;\ \overline{\mathtt{x}}{:}\overline{\mathtt{D}} \models \mathtt{Q}$ and $\mathtt{P}{\not\prec_{\overline{\mathtt{x}}}}\mathtt{Q}$. Therefore we have a contradiction.

- Case there is one such predicate. As shown above $\mathtt{P}$ is such a predicate, so it must be the only one. Therefore, for each other predicate $\mathtt{P}' \in \overline{\mathtt{P}}$, it must be the case that either TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \not\models \mathtt{P}'$ or $\mathtt{P}{\prec_{\overline{\mathtt{x}}}}\mathtt{P}'$. Therefore by OverApp we have *overridesIfApplicable*(P,P$'$,$\overline{\mathtt{x}}$,$\overline{\mathtt{D}}$). Since $\mathtt{P}{\doteq}\mathtt{P}$, by OverApp also *overridesIfApplicable*(P,P,$\overline{\mathtt{x}}$,$\overline{\mathtt{D}}$). So we have shown that *overridesIfApplicable*(P,$\overline{\mathtt{P}}$,$\overline{\mathtt{x}}$,$\overline{\mathtt{D}}$). Therefore the result follows with $\mathtt{P} = \mathtt{Q}$.

- Case there are $i > 1$ such predicates. Then there exists a predicate $\mathtt{R}$ such that $\mathtt{R}{\not\doteq}\mathtt{P}$ and TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models \mathtt{R}$ and $\mathtt{P}{\not\prec_{\overline{\mathtt{x}}}}\mathtt{R}$. By definition of $\prec$ we have that either $\mathtt{P}{\not\preceq_{\overline{\mathtt{x}}}}\mathtt{R}$ or $\mathtt{R}{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}$, so there are a few subcases.

  - Case $\mathtt{P}{\preceq_{\overline{\mathtt{x}}}}\mathtt{R}$ and $\mathtt{R}{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}$. By T-Class and T-Meth we have *unambiguous*(P,R,$\overline{\mathtt{x}}$,$\overline{\mathtt{P}}$). Therefore by Unamb we have $\mathtt{P}{\doteq}\mathtt{R}$. We saw above that $\mathtt{R}{\not\doteq}\mathtt{P}$, so we have a contradiction.
  - Case $\mathtt{P}{\not\preceq_{\overline{\mathtt{x}}}}\mathtt{R}$. By T-Class and T-Meth we have *unambiguous*(P,R,$\overline{\mathtt{x}}$,$\overline{\mathtt{P}}$). Therefore by Unamb we have $\overline{\mathtt{Q}} = [\mathtt{Q} \mid \mathtt{Q} \in \overline{\mathtt{P}}$ and $\mathtt{Q} \preceq_{\overline{\mathtt{x}}} \mathtt{P}$ and $\mathtt{Q} \preceq_{\overline{\mathtt{x}}} \mathtt{R}]$ and $\overline{\mathtt{x}} \models (\mathtt{P}{\wedge}\mathtt{R}){\Rightarrow}\bigvee\overline{\mathtt{Q}}$. By Lemma 6.9 we have $TT \supseteq$ *restrict*(TT, P), so by Valid we have TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models (\mathtt{P}{\wedge}\mathtt{R}){\Rightarrow}\bigvee\overline{\mathtt{Q}}$. By the definition of $\Rightarrow$ we have TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models \neg(\mathtt{P}{\wedge}\mathtt{R}){\vee}\bigvee\overline{\mathtt{Q}}$. By P-Or1 and P-Or2 this means that either TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models \neg(\mathtt{P}{\wedge}\mathtt{R})$ or TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models \bigvee\overline{\mathtt{Q}}$. If the former, then by P-Not we have TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \not\models \mathtt{P}{\wedge}\mathtt{R}$, and by P-And this means that either TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \not\models \mathtt{P}$ or TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \not\models \mathtt{R}$. But we've already shown that TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models \mathtt{P}$ and TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models \mathtt{R}$, so we have a contradiction.

    Therefore, it must be the case that TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models \bigvee\overline{\mathtt{Q}}$, and by Lemma 6.10 there is some $\mathtt{P}_0 \in \overline{\mathtt{Q}}$ such that TT;$\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models \mathtt{P}_0$. By Definition 6.1 we have D covers $\overline{\mathtt{D}}$ for m via ($\mathtt{P}_0$,$\overline{\mathtt{P}}$,$\overline{\mathtt{x}}$), so the result follows by induction if we can show that the number j of predicates $\mathtt{P}' \in \overline{\mathtt{P}}$ such that TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models \mathtt{P}'$ and $\mathtt{P}_0{\not\prec_{\overline{\mathtt{x}}}}\mathtt{P}'$ is smaller than i. First we show that $j \leq i$. Consider some $\mathtt{P}' \in \overline{\mathtt{P}}$ such that TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models \mathtt{P}'$ and $\mathtt{P}_0{\not\prec_{\overline{\mathtt{x}}}}\mathtt{P}'$. Then we will show that also $\mathtt{P}{\not\prec_{\overline{\mathtt{x}}}}\mathtt{P}'$. Suppose not, so $\mathtt{P}{\prec_{\overline{\mathtt{x}}}}\mathtt{P}'$. Therefore $\mathtt{P}{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}'$ and $\mathtt{P}'{\not\preceq_{\overline{\mathtt{x}}}}\mathtt{P}$. Since $\mathtt{P}_0 \in \overline{\mathtt{Q}}$, we have $\mathtt{P}_0{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}$, and since $\mathtt{P}{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}'$, by Lemma 6.18 we have $\mathtt{P}_0{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}'$. Also, since $\mathtt{P}_0{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}$ and $\mathtt{P}'{\not\preceq_{\overline{\mathtt{x}}}}\mathtt{P}$ then we have $\mathtt{P}'{\not\preceq_{\overline{\mathtt{x}}}}\mathtt{P}_0$ (since if $\mathtt{P}'{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}_0$ then by Lemma 6.18 also $\mathtt{P}'{\prec_{\overline{\mathtt{x}}}}\mathtt{P}$, which is a contradiction). Therefore we have $\mathtt{P}_0{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}'$ and $\mathtt{P}'{\not\preceq_{\overline{\mathtt{x}}}}\mathtt{P}_0$, so $\mathtt{P}_0{\prec_{\overline{\mathtt{x}}}}\mathtt{P}'$, and we have a contradiction.

    As shown above, every predicate of the appropriate form for $\mathtt{P}_0$ is also of the appropriate form for P. To end the proof, we show that converse does not hold. We saw earlier that TT; $\overline{\mathtt{x}}$:$\overline{\mathtt{D}} \models \mathtt{P}$, and as argued in the first case of the proof, we have $\mathtt{P}{\not\prec_{\overline{\mathtt{x}}}}\mathtt{P}$. Therefore, P is one of the predicates of the appropriate form for P. We're done if we can show that $\mathtt{P}_0{\prec_{\overline{\mathtt{x}}}}\mathtt{P}$, since this means that P is not one of the predicates of the appropriate form for $\mathtt{P}_0$. Since $\mathtt{P} \in \overline{\mathtt{Q}}$, we have $\mathtt{P}_0{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}$ and $\mathtt{P}_0{\preceq_{\overline{\mathtt{x}}}}\mathtt{R}$. We assumed above that $\mathtt{P}{\not\preceq_{\overline{\mathtt{x}}}}\mathtt{R}$, so also $\mathtt{P}{\not\preceq_{\overline{\mathtt{x}}}}\mathtt{P}_0$ (since if $\mathtt{P}{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}_0$ then by Lemma 6.18 also $\mathtt{P}{\prec_{\overline{\mathtt{x}}}}\mathtt{R}$, which is a contradiction). Therefore we have shown that $\mathtt{P}_0{\preceq_{\overline{\mathtt{x}}}}\mathtt{P}$ and $\mathtt{P}{\not\preceq_{\overline{\mathtt{x}}}}\mathtt{P}_0$, so $\mathtt{P}_0{\prec_{\overline{\mathtt{x}}}}\mathtt{P}$.

□

Lemma 6.20 If $\mathtt{C} \in \mathrm{dom}(TT)$ and $TT \vdash$ C<:D and $\overline{\mathtt{D}} \subseteq \mathrm{dom}(TT)$ and D covers $\overline{\mathtt{D}}$ for m via (P,$\overline{\mathtt{P}}$,$\overline{\mathtt{x}}$), then there exists t such that *mbody*(m,C,$\overline{\mathtt{D}}$) = ($\overline{\mathtt{x}}$, t).
Proof We prove this lemma by induction on the number of classes E such that $TT \vdash$ C<:E and $TT \vdash$ E<:D.

- Case there are zero such classes. By S-Ref we have $TT \vdash$ C<:C, and we are given $TT \vdash$ C<:D, so C is such a class and we have a contradiction.

- Case there is one such class. Since by S-Ref we have $TT \vdash$ C<:C and $TT \vdash$ D<:D, and since we are given $TT \vdash$ C<:D, we know that both C and D are such classes. Therefore it must be the case that $\texttt{C} = \texttt{D}$. Since $\texttt{D}$ covers $\overline{\texttt{D}}$ for $\texttt{m}$ via ($\texttt{P}$,$\overline{\texttt{P}}$,$\overline{\texttt{x}}$), by Definition 6.1 we have $TT(\texttt{C}) = \texttt{class C} \cdots \{\cdots\ \overline{\texttt{M}}\}$ and $\texttt{S m(}\overline{\texttt{S}}\ \overline{\texttt{x}}\texttt{) when}\ \overline{\texttt{P}}\ \{\cdots\} \in \overline{\texttt{M}}$ and $\texttt{P} \in \overline{\texttt{P}}$ and $TT;\ \overline{\texttt{x}}\texttt{:}\overline{\texttt{D}} \models \texttt{P}$. Also, by Lemma 6.19 there exists $\texttt{Q} \in \overline{\texttt{P}}$ such that $TT;\ \overline{\texttt{x}}\texttt{:}\overline{\texttt{D}} \models \texttt{Q}$ and *overridesIfApplicable*($\texttt{Q}$,$\overline{\texttt{P}}$,$\overline{\texttt{x}}$,$\overline{\texttt{D}}$). Then the result follows by MBody1.

- Case there are $i > 1$ such classes. Then there is some $\texttt{E}$ such that $\texttt{E} \neq \texttt{C}$ and $TT \vdash \texttt{C<:E}$ and $TT \vdash \texttt{E<:D}$. Therefore also $\texttt{C} \neq \texttt{D}$, or else we would have a nontrivial cycle in the subtyping relation. There are two subcases.
  - Case there exist $\texttt{P}_0$ and $\overline{\texttt{P}_0}$ such that $\texttt{C}$ covers $\overline{\texttt{D}}$ for $\texttt{m}$ via ($\texttt{P}_0$,$\overline{\texttt{P}_0}$,$\overline{\texttt{x}}$). Then by Definition 6.1 we have $TT(\texttt{C}) = \texttt{class C} \cdots \{\cdots\ \overline{\texttt{M}}\}$ and $\texttt{S m(}\overline{\texttt{S}}\ \overline{\texttt{x}}\texttt{) when}\ \overline{\texttt{P}_0}\ \{\cdots\} \in \overline{\texttt{M}}$ and $\texttt{P}_0 \in \overline{\texttt{P}_0}$ and $TT;\ \overline{\texttt{x}}\texttt{:}\overline{\texttt{D}} \models \texttt{P}_0$. Also, by Lemma 6.19 there exists $\texttt{Q} \in \overline{\texttt{P}_0}$ such that $TT;\ \overline{\texttt{x}}\texttt{:}\overline{\texttt{D}} \models \texttt{Q}$ and *overridesIfApplicable*($\texttt{Q}$,$\overline{\texttt{P}_0}$,$\overline{\texttt{x}}$,$\overline{\texttt{D}}$). Then the result follows by MBody1.
  - Case there does not exist $\texttt{P}_0$ and $\overline{\texttt{P}_0}$ such that $\texttt{C}$ covers $\overline{\texttt{D}}$ for $\texttt{m}$ via ($\texttt{P}_0$,$\overline{\texttt{P}_0}$,$\overline{\texttt{x}}$). Since $\texttt{C} \in \text{dom}(TT)$, we have that $TT(\texttt{C}) = \texttt{class C extends C}_0 \cdots \{\cdots\ \overline{\texttt{M}}\}$. Since $\texttt{C} \neq \texttt{D}$ and $TT \vdash \texttt{C<:D}$, it must be the case that $TT \vdash \texttt{C}_0\texttt{<:D}$ as well. Further, there are strictly fewer than i classes $\texttt{E}$ such that $TT \vdash \texttt{C}_0\texttt{<:E}$ and $TT \vdash \texttt{E<:D}$. Therefore by induction there exists $\texttt{t}$ such that *mbody*($\texttt{m}$,$\texttt{C}_0$,$\overline{\texttt{D}}$) = ($\overline{\texttt{x}}$, $\texttt{t}$). Since there does not exist $\texttt{P}_0$ and $\overline{\texttt{P}_0}$ such that $\texttt{C}$ covers $\overline{\texttt{D}}$ for $\texttt{m}$ via ($\texttt{P}_0$,$\overline{\texttt{P}_0}$,$\overline{\texttt{x}}$), by Definition 6.1, there are several subcases.
    - $\texttt{C} \notin \text{dom}(TT)$. But we are given that $\texttt{C} \in \text{dom}(TT)$, so we have a contradiction.
    - $TT(\texttt{C}) = \texttt{class C extends C}_0 \cdots \{\cdots\ \overline{\texttt{M}}\}$ but $\texttt{m}$ is not defined in $\overline{\texttt{M}}$. Then the result follows by MBody3.
    - $TT(\texttt{C}) = \texttt{class C extends C}_0 \cdots \{\cdots\ \overline{\texttt{M}}\}$ and $\texttt{S m(}\overline{\texttt{S}}\ \overline{\texttt{x}}\texttt{) when}\ \overline{\texttt{P}}\ \{\cdots\} \in \overline{\texttt{M}}$, but there is no $\texttt{P}_i$ such that $TT;\overline{\texttt{x}}\texttt{:}\overline{\texttt{D}} \models \texttt{P}_i$. Then the result follows by MBody2.

□

Theorem 6.2 (Progress) If $\bullet \vdash \texttt{t} : \texttt{T}$, then either $\texttt{t}$ is a value, $\texttt{t}$ contains a subexpression of the form $\texttt{(U)(new C(}\overline{\texttt{u}}\texttt{))}$ where $TT \vdash \texttt{C}\not\texttt{<}\texttt{:U}$, or there exists some term $\texttt{s}$ such that $\texttt{t} \longrightarrow \texttt{s}$.
Proof By induction on the depth of the derivation of $\bullet \vdash \texttt{t} : \texttt{T}$. Case analysis of the last rule in the derivation.

- Case T-Var: Then $\texttt{t}$ has the form $\texttt{x}$ and $\texttt{x:T} \in \bullet$, which is a contradiction. Therefore, T-Var cannot be the last rule in the derivation.

- Case T-Field: Then $\texttt{t}$ has the form $\texttt{t}_0\texttt{.f}_i$ and $\texttt{T}$ has the form $\texttt{T}_i$ and $\bullet \vdash \texttt{t}_0 : \texttt{C}_0$ and *fields*($\texttt{C}_0$) = $\overline{\texttt{T}}\ \overline{\texttt{f}}$. By induction, there are three subcases.
  - Case $\texttt{t}_0$ is a value. Then $\texttt{t}_0$ has the form $\texttt{new D}_0\texttt{(}\overline{\texttt{v}}\texttt{)}$. Since $\bullet \vdash \texttt{t}_0 : \texttt{C}_0$, by T-New $\texttt{D}_0$ is $\texttt{C}_0$ and $\overline{\texttt{v}}$ has the same length as $\overline{\texttt{f}}$. Then by E-ProjNew we have $\texttt{t}_0\texttt{.f}_i \longrightarrow \texttt{v}_i$.
  - Case $\texttt{t}_0$ contains a subexpression of the form $\texttt{(U)(new C(}\overline{\texttt{u}}\texttt{))}$ where $TT \vdash \texttt{C}\not\texttt{<}\texttt{:U}$. Then also $\texttt{t}$ contains a subexpression of the form $\texttt{(U)(new C(}\overline{\texttt{u}}\texttt{))}$ where $TT \vdash \texttt{C}\not\texttt{<}\texttt{:U}$.
  - Case there exists some term $\texttt{s}_0$ such that $\texttt{t}_0 \longrightarrow \texttt{s}_0$. Then by E-Field we have $\texttt{t}_0\texttt{.f}_i \longrightarrow \texttt{s}_0\texttt{.f}_i$.

- Case T-Invk: Then $\texttt{t}$ has the form $\texttt{t}_0\texttt{.m(}\overline{\texttt{t}}\texttt{)}$ and $\bullet \vdash \texttt{t}_0 : \texttt{T}_0$ and *mtype*($\texttt{m}$,$\texttt{T}_0$) = $\overline{\texttt{T}}\rightarrow\texttt{T}$ and $\bullet \vdash \overline{\texttt{t}} : \overline{\texttt{S}}$ and $TT \vdash \overline{\texttt{S}}\texttt{<:}\overline{\texttt{T}}$. By induction, there are three subcases.
  - Case $\texttt{t}_0$ is a value. Then $\texttt{t}_0$ has the form $\texttt{new C}_0\texttt{(}\overline{\texttt{v}}\texttt{)}$. Since $\bullet \vdash \texttt{t}_0 : \texttt{T}_0$, by T-New $\texttt{T}_0$ is $\texttt{C}_0$. By induction, there are three subcases.

    * Case all terms in $\overline{\mathtt{t}}$ are values. Then $\overline{\mathtt{t}}$ has the form $\mathtt{new}\ \overline{\mathtt{D}}(\ldots)$, and by the sanity conditions we know that $\overline{\mathtt{D}} \subseteq \mathrm{dom}(TT)$. Since *mtype*($\mathtt{m}$,$\mathtt{C}_0$) $= \overline{\mathtt{T}}\rightarrow\mathtt{T}$, by Lemma 6.14 there exists $\overline{\mathtt{x}}$ of the same length as $\overline{\mathtt{T}}$ such that *mformals*($\mathtt{m}$,$\mathtt{C}_0$) $= \overline{\mathtt{x}}$. Therefore, $\overline{\mathtt{x}}$ also has the same length as $\overline{\mathtt{D}}$. Then by Lemma 6.15 there exist $\mathtt{D}$, $\mathtt{P}$, and $\overline{\mathtt{P}}$ such that $TT \vdash \mathtt{C}_0\mathtt{<:D}$ and $\mathtt{D}$ covers $\overline{\mathtt{D}}$ for $\mathtt{m}$ via ($\mathtt{P}$,$\overline{\mathtt{P}}$,$\overline{\mathtt{x}}$). Also since *mtype*($\mathtt{m}$,$\mathtt{C}_0$) $= \overline{\mathtt{T}}\rightarrow\mathtt{T}$, by MType-C1 and MType-C2 we have $\mathtt{C}_0 \in \mathrm{dom}(TT)$. Then by Lemma 6.20 there exists a term $\mathtt{s}_0$ such that *mbody*($\mathtt{m}$,$\mathtt{C}_0$,$\overline{\mathtt{D}}$) $=$ ($\overline{\mathtt{x}}$, $\mathtt{s}_0$). Then by E-InvkNew we have $\mathtt{t}_0\mathtt{.m(}\overline{\mathtt{t}}\mathtt{)} \longrightarrow [\overline{\mathtt{x}} \mapsto \overline{\mathtt{t}},\ \mathtt{this} \mapsto \mathtt{new}\ \mathtt{C}_0(\overline{\mathtt{v}})]\mathtt{s}_0$.
    * Case some term in $\overline{\mathtt{t}}$ contains a subexpression of the form $\mathtt{(U)(new\ C(}\overline{\mathtt{u}}\mathtt{))}$ where $TT \vdash \mathtt{C}\not<\mathtt{:U}$. Then also $\mathtt{t}$ contains a subexpression of the form $\mathtt{(U)(new\ C(}\overline{\mathtt{u}}\mathtt{))}$ where $TT \vdash \mathtt{C}\not<\mathtt{:U}$.
    * Case no term in $\overline{\mathtt{t}}$ contains a subexpression of the form $\mathtt{(U)(new\ C(}\overline{\mathtt{u}}\mathtt{))}$ where $TT \vdash \mathtt{C}\not<\mathtt{:U}$. Further, there is some $\mathtt{t}_i \in \overline{\mathtt{t}}$ for which there exists a term $\mathtt{s}_i$ such that $\mathtt{t}_i \longrightarrow \mathtt{s}_i$. Further, all $\mathtt{t}_j$ such that $1 \leq j < i$ are values. Then by E-Invk-Arg we have $\mathtt{t}_0\mathtt{.m(}\overline{\mathtt{t}}\mathtt{)} \longrightarrow \mathtt{t}_0\mathtt{.m(t}_1\mathtt{,}\ldots\mathtt{,t}_{i-1}\mathtt{,s}_i\mathtt{,t}_{i+1}\mathtt{,}\ldots\mathtt{,t}_n\mathtt{)}$.
  - Case $\mathtt{t}_0$ contains a subexpression of the form $\mathtt{(U)(new\ C(}\overline{\mathtt{u}}\mathtt{))}$ where $TT \vdash \mathtt{C}\not<\mathtt{:U}$. Then also $\mathtt{t}$ contains a subexpression of the form $\mathtt{(U)(new\ C(}\overline{\mathtt{u}}\mathtt{))}$ where $TT \vdash \mathtt{C}\not<\mathtt{:U}$.
  - Case there exists some term $\mathtt{s}_0$ such that $\mathtt{t}_0 \longrightarrow \mathtt{s}_0$. Then by E-Invk-Recv we have $\mathtt{t}_0\mathtt{.m(}\overline{\mathtt{t}}\mathtt{)} \longrightarrow \mathtt{s}_0\mathtt{.m(}\overline{\mathtt{t}}\mathtt{)}$.

- Case T-New: Then $\mathtt{t}$ has the form $\mathtt{new}\ \mathtt{C}_0(\overline{\mathtt{t}})$ and $\mathtt{T}$ is $\mathtt{C}_0$ and *fields*($\mathtt{C}_0$) $= \overline{\mathtt{T}}\ \overline{\mathtt{f}}$ and $\bullet \vdash \overline{\mathtt{t}} : \overline{\mathtt{S}}$ and $TT \vdash \overline{\mathtt{S}}\mathtt{<:}\overline{\mathtt{T}}$. By induction, there are three subcases.
  - Case all terms in $\overline{\mathtt{t}}$ are values. Then also $\mathtt{t}$ is a value.
  - Case some term in $\overline{\mathtt{t}}$ contains a subexpression of the form $\mathtt{(U)(new\ C(}\overline{\mathtt{u}}\mathtt{))}$ where $TT \vdash \mathtt{C}\not<\mathtt{:U}$. Then also $\mathtt{t}$ contains a subexpression of the form $\mathtt{(U)(new\ C(}\overline{\mathtt{u}}\mathtt{))}$ where $TT \vdash \mathtt{C}\not<\mathtt{:U}$.
  - Case no term in $\overline{\mathtt{t}}$ contains a subexpression of the form $\mathtt{(U)(new\ C(}\overline{\mathtt{u}}\mathtt{))}$ where $TT \vdash \mathtt{C}\not<\mathtt{:U}$. Further, there is some $\mathtt{t}_i \in \overline{\mathtt{t}}$ for which there exists a term $\mathtt{s}_i$ such that $\mathtt{t}_i \longrightarrow \mathtt{s}_i$. Further, all $\mathtt{t}_j$ such that $1 \leq j < i$ are values. Then by E-New-Arg we have $\mathtt{new}\ \mathtt{C}_0(\overline{\mathtt{t}}) \longrightarrow \mathtt{new}\ \mathtt{C}_0\mathtt{(t}_1\mathtt{,}\ldots\mathtt{,t}_{i-1}\mathtt{,s}_i\mathtt{,t}_{i+1}\mathtt{,}\ldots\mathtt{,t}_n\mathtt{)}$.

- Case T-UCast: Then $\mathtt{t}$ has the form $\mathtt{(T)t}_0$ and $\bullet \vdash \mathtt{t}_0 : \mathtt{S}$ and $TT \vdash \mathtt{S<:T}$. By induction, there are three subcases.
  - Case $\mathtt{t}_0$ is a value. Then $\mathtt{t}_0$ has the form $\mathtt{new}\ \mathtt{C}_0(\overline{\mathtt{v}})$ and by T-New $\mathtt{S}$ is $\mathtt{C}_0$. Then by E-CastNew we have $\mathtt{(T)t}_0 \longrightarrow \mathtt{t}_0$.
  - Case $\mathtt{t}_0$ contains a subexpression of the form $\mathtt{(U)(new\ C(}\overline{\mathtt{u}}\mathtt{))}$ where $TT \vdash \mathtt{C}\not<\mathtt{:U}$. Then also $\mathtt{t}$ contains a subexpression of the form $\mathtt{(U)(new\ C(}\overline{\mathtt{u}}\mathtt{))}$ where $TT \vdash \mathtt{C}\not<\mathtt{:U}$.
  - Case there exists some term $\mathtt{s}_0$ such that $\mathtt{t}_0 \longrightarrow \mathtt{s}_0$. Then by E-Cast we have $\mathtt{(T)t}_0 \longrightarrow \mathtt{(T)s}_0$.

- Case T-DCast: Then $\mathtt{t}$ has the form $\mathtt{(T)t}_0$ and $\bullet \vdash \mathtt{t}_0 : \mathtt{S}$ and $TT \vdash \mathtt{T<:S}$ and $\mathtt{T} \neq \mathtt{S}$. By induction, there are three subcases.
  - Case $\mathtt{t}_0$ is a value. Then $\mathtt{t}_0$ has the form $\mathtt{new}\ \mathtt{C}_0(\overline{\mathtt{v}})$ and by T-New $\mathtt{S}$ is $\mathtt{C}_0$. If $TT \vdash \mathtt{C}_0\mathtt{<:T}$ then by E-CastNew we have $\mathtt{(T)t}_0 \longrightarrow \mathtt{t}_0$. Otherwise $TT \vdash \mathtt{C}_0\not<\mathtt{:T}$, so $\mathtt{t}$ contains a subexpression of the form $\mathtt{(U)(new\ C(}\overline{\mathtt{u}}\mathtt{))}$ where $TT \vdash \mathtt{C}\not<\mathtt{:U}$.

  - Case $\mathtt{t}_0$ contains a subexpression of the form `(U)(new C(`$\overline{\mathtt{u}}$`))` where $TT \vdash \mathtt{C} \not<: \mathtt{U}$. Then also `t` contains a subexpression of the form `(U)(new C(`$\overline{\mathtt{u}}$`))` where $TT \vdash \mathtt{C} \not<: \mathtt{U}$.
  - Case there exists some term $\mathtt{s}_0$ such that $\mathtt{t}_0 \longrightarrow \mathtt{s}_0$. Then by E-Cast we have $(\mathtt{T})\mathtt{t}_0 \longrightarrow (\mathtt{T})\mathtt{s}_0$.

- Case T-SCast: Then `t` has the form $(\mathtt{T})\mathtt{t}_0$ and $\bullet \vdash \mathtt{t}_0 : \mathtt{S}$ and $TT \vdash \mathtt{S} \not<: \mathtt{T}$ and $TT \vdash \mathtt{T} \not<: \mathtt{S}$ and a *stupid warning* is generated. By induction, there are three subcases.
  - Case $\mathtt{t}_0$ is a value. Then $\mathtt{t}_0$ has the form `new` $\mathtt{C}_0(\overline{\mathtt{v}})$ and by T-New `S` is $\mathtt{C}_0$. Then `t` contains a subexpression of the form `(U)(new C(`$\overline{\mathtt{u}}$`))` where $TT \vdash \mathtt{C} \not<: \mathtt{U}$.
  - Case $\mathtt{t}_0$ contains a subexpression of the form `(U)(new C(`$\overline{\mathtt{u}}$`))` where $TT \vdash \mathtt{C} \not<: \mathtt{U}$. Then also `t` contains a subexpression of the form `(U)(new C(`$\overline{\mathtt{u}}$`))` where $TT \vdash \mathtt{C} \not<: \mathtt{U}$.
  - Case there exists some term $\mathtt{s}_0$ such that $\mathtt{t}_0 \longrightarrow \mathtt{s}_0$. Then by E-Cast we have $(\mathtt{T})\mathtt{t}_0 \longrightarrow (\mathtt{T})\mathtt{s}_0$.

□

References

[1] K. Bruce, L. Cardelli, G. Castagna, T. H. O. Group, G. T. Leavens, and B. Pierce. On binary methods. *Theory and Practice of Object Systems*, 1(3):217–238, 1995.

[2] G. Castagna. Covariance and contravariance: conflict without a cause. *ACM Trans. Program. Lang. Syst.*, 17(3):431–447, 1995.

[3] C. Chambers. Object-oriented multi-methods in Cecil. In O. L. Madsen, editor, *Proceedings ECOOP '92*, LNCS 615, pages 33–56. Springer-Verlag, June 1992.

[4] A. Igarashi, B. C. Pierce, and P. Wadler. Featherweight Java: a minimal core calculus for Java and GJ. *ACM Transactions on Programming Languages and Systems*, 23(3):396–450, May 2001.

[5] T. Millstein. Practical predicate dispatch. In *OOPSLA 2004 Conference on Object-Oriented Programming, Systems, Languages, and Applications*, Oct. 2004.