Modular Typechecking for Hierarchically Extensible Datatypes and Functions

Todd Millstein Craig Chambers

Department of Computer Science and Engineering
University of Washington
{todd,chambers}@cs.washington.edu

Technical Report UW-CSE-01-07-02
July 2001, revised March 2002

Abstract

This technical report provides the formal details of Mini-Eml, a core language for Eml. Eml is an ML-like language containing hierarchically, extensible datatypes and functions while retaining modular typechecking. Section 1 presents the syntax of Mini-Eml. Section 2 presents its dynamic semantics and section 3 presents its static semantics. Section 4 gives the subject reduction proof, and section 5 gives the progress proof.

$$
\begin{array}{rcl}
T & ::= & Tn \mid Ct \mid T_1 \to T_2 \mid T_1 * \cdots * T_k \\
Mt & ::= & \texttt{\#}\ Ct \mid T_1 * \cdots * T_{i-1} * Mt * T_{i+1} * \cdots * T_k \\
E & ::= & I \mid Fv \mid E_1\ E_2 \mid Ct(\overline{E}) \mid (\overline{E}) \mid Ct\ \{\overline{V} = \overline{E}\} \\
Pat & ::= & \_ \mid I\ \texttt{as}\ Pat \mid C\ \{\overline{V} = \overline{Pat}\} \mid (\overline{Pat}) \\
Ct & ::= & \overline{T}\ C \qquad\qquad Fv ::= \overline{T}\ F \\
C & ::= & Bn.Cn \qquad\quad V ::= Bn.Vn \\
F & ::= & Bn.Fn
\end{array}
$$

(a)

$$
\begin{array}{rcl}
B & ::= & \texttt{block}\ \ Bn = \texttt{blk extends}\ \overline{Bn}\ \overline{Ood}\ \texttt{end} \\
Ood & ::= & \langle\texttt{abstract}\rangle\ \texttt{class}\ \overline{Tn}\ Cn(\overline{I} : \overline{T}) \\
 & & \quad \langle\langle\texttt{extends}\ Ct(\overline{E})\rangle\rangle\ \texttt{of}\ \{\overline{Vn} : \overline{T_0} = \overline{E_0}\} \\
 & \mid & \texttt{fun}\ \overline{Tn}\ Fn : Mt \to T \\
 & \mid & \texttt{extend fun}_{Mn}\ \overline{Tn}\ F\ \ Pat = E
\end{array}
$$

(b)

Figure 1: (a) MINI-EML types, expressions, and patterns; (b) MINI-EML blocks. Metavariable Tn ranges over type variable names, I over identifier names, Cn over class names, Vn over instance variable names, Fn over function names, and Mn over case names. $\overline{D}$ denotes a comma-separated list of elements (and is independent of any variable named D). Angle brackets ($<>$) and double angle brackets ($<<>>$) denote independent optional pieces of syntax. The notation $\overline{V} = \overline{E}$ abbreviates $V_1 = E_1, \ldots, V_k = E_k$ where $\overline{V}$ is $V_1, \ldots, V_k$ and $\overline{E}$ is $E_1, \ldots, V_k$ for some $k \geq 0$, and similarly for $\overline{V} = \overline{Pat}$, $\overline{Vn} : \overline{T_0} = \overline{E_0}$, and $\overline{I} : \overline{T}$.

1 Syntax

1.1 Types, Expressions, and Patterns

Figure 1a defines the syntax of types, expressions, and patterns in MINI-EML. MINI-EML types include type variables, class types, function types, and tuple types. The domain Mt represents *marked types*, which contain a `#` mark on a single component class type. Marked types are used to implement our modular type system discussed in section 3.

Expressions include identifiers, function values, function application, constructor calls, tuples, and instance expressions. The instance expression $Ct\ \{\overline{V} = \overline{E}\}$ is not available at the source level, as instances may only be created via a constructor call. Patterns include the wildcard pattern, identifier binding, class patterns, and tuple patterns. We assume that all identifiers bound in a given pattern are distinct.

The subset of expressions that are MINI-EML values is described by the following grammar:

$v ::= Ct\ \{\overline{V} = \overline{v}\} \mid Fv \mid (\overline{v})$

Values include class instances, function values, and tuple values.

1.2 Declarations, Blocks and Programs

The syntax of MINI-EML blocks and declarations is shown in figure 1(b). A block consists of a sequence of class, extensible function, and function case declarations. The class (function, case) names introduced in a given block are assumed to be distinct. The type variables parameterizing a given OO declaration are assumed to be distinct. The instance variable names introduced in a given class declaration are assumed to be distinct.

A MINI-EML program is a pair of a *block table* and an expression. A block table is a finite function from block names to blocks. The semantics assumes a fixed block table denoted BT. The domain of a block table BT is denoted $\mathrm{dom}(BT)$. The block table is assumed to satisfy some sanity conditions: (1) $BT(Bn) = \texttt{block}\ Bn = \texttt{blk} \ldots$ for every $Bn \in \mathrm{dom}(BT)$; (2) for every block name Bn appearing anywhere in the program, we have $Bn \in \mathrm{dom}(BT)$.

2 Dynamic Semantics

2.1 Preliminaries

MINI-EML's dynamic semantics is defined as a mostly standard small-step operational semantics. The block table BT is accessed when information about a given OO declaration is required in the evaluation of an expression. In addition, several side judgments are necessary to express the function-case lookup semantics.

The metavariable e ranges over environments, which are finite functions from identifiers to values. We use $|\overline{D}|$ to denote the length of the sequence $\overline{D}$. The notation $[I_1 \mapsto E_1, \ldots, I_k \mapsto E_k]D$ denotes the expression resulting from the simultaneous substitution of E_i for each occurrence of I_i in D, for $1 \leq i \leq k$, and similarly for $[Tn_1 \mapsto T_1, \ldots, Tn_k \mapsto T_k]D$. We use $[\overline{I} \mapsto \overline{v}]D$ as a shorthand for $[I_1 \mapsto v_1, \ldots, I_k \mapsto v_k]D$, where $\overline{I} = I_1, \ldots, I_k$ and $\overline{v} = v_1, \ldots, v_k$, and similarly for $[\overline{Tn} \mapsto \overline{T}]D$. In a given inference rule, fragments enclosed in <> must either be all present or all absent, and similarly for <<>>. We sometimes treat sequences as if they were sets. For example, $Ood \in \overline{Ood}$ means that Ood is one of the declarations in $\overline{Ood}$. We use $Ood \in BT(Bn)$ as shorthand for $BT(Bn) = \texttt{block}\ Bn = \texttt{blk extends}\ \overline{Bn}\ \overline{Ood}\ \texttt{end}$ and $Ood \in \overline{Ood}$.

2.2 Expressions

$$\boxed{E \longrightarrow E'}$$

$$\frac{Ct = (\overline{T}\ C) \qquad \text{concrete}(C) \qquad \text{rep}(Ct(\overline{E_0})) = \{\overline{V} = \overline{E_1}\}}{Ct(\overline{E_0}) \longrightarrow Ct\ \{\overline{V} = \overline{E_1}\}}\ \text{E-NEW}$$

$$\frac{E \longrightarrow E'}{Ct\ \{\overline{V_0} = \overline{E_0}, V = E, \overline{V_1} = \overline{E_1}\} \longrightarrow \quad Ct\ \{\overline{V_0} = \overline{E_0}, V = E', \overline{V_1} = \overline{E_1}\}}\ \text{E-REP}$$

$$\frac{E \longrightarrow E'}{(\overline{E_0}, E, \overline{E_1}) \longrightarrow (\overline{E_0}, E', \overline{E_1})}\ \text{E-TUP}$$

$$\frac{E_1 \longrightarrow E_1'}{E_1\ E_2 \longrightarrow E_1'\ E_2}\ \text{E-APP1} \qquad \frac{E_2 \longrightarrow E_2'}{E_1\ E_2 \longrightarrow E_1\ E_2'}\ \text{E-APP2}$$

$$\frac{\text{most-specific-case-for}(Fv, v) = (\{(\overline{I}, \overline{v})\}, E)}{Fv\ v \longrightarrow [\overline{I} \mapsto \overline{v}]E}\ \text{E-APPRED}$$

Rule E-APPRED: The notation $(\overline{I}, \overline{v})$ abbreviates $(I_1, v_1), \ldots, (I_k, v_k)$.

2.3 Function Application

These auxiliary judgments are used to specify the function-case lookup semantics. Some of these judgments are used by the static semantics as well.

$$\boxed{\text{most-specific-case-for } (Fv, v) = (e, E)}$$

$$\frac{\begin{array}{c}(\texttt{extend fun}_{Mn}\ \overline{Tn}\ F\ Pat = E) \in BT(Bn) \qquad \text{match}(v, Pat) = e \\ \forall Bn' \in \text{dom}(BT).\forall(\texttt{extend fun}_{Mn'}\ \overline{Tn'}\ F\ Pat' \ldots) \in BT(Bn').\forall e'. \\ (\text{match}(v, Pat') = e' \wedge Bn.Mn \neq Bn'.Mn' \Rightarrow Pat \leq Pat' \wedge Pat' \not\leq Pat)\end{array}}{\text{most-specific-case-for } ((\overline{T}\ F), v) = (e, [\overline{Tn} \mapsto \overline{T}]E)}\ \text{LOOKUP}$$

$$\boxed{\text{match}(v, \mathit{Pat}) = e}$$

$$\frac{}{\text{match}(v, \_) = \{\}}\ \textsc{E-MatchWild}$$

$$\frac{\text{match}(v, \mathit{Pat}) = e}{\text{match}(v, I \texttt{ as } \mathit{Pat}) = e \cup \{(I, v)\}}\ \textsc{E-MatchBind}$$

$$\frac{C \leq C' \qquad \text{match}(\overline{v}, \overline{\mathit{Pat}}) = \overline{e}}{\text{match}(\overline{T}\ C\ \{\overline{V} = \overline{v}, \overline{V_1} = \overline{v_1}\}, C'\ \{\overline{V} = \overline{\mathit{Pat}}\}) = \bigcup \overline{e}}\ \textsc{E-MatchClass}$$

Rule E-MatchClass: The notation $\text{match}(\overline{v}, \overline{\mathit{Pat}}) = \overline{e}$ abbreviates $\text{match}(v_1, \mathit{Pat}_1) = e_1 \cdots \text{match}(v_k, \mathit{Pat}_k) = e_k$.

$$\frac{\text{match}(\overline{v}, \overline{\mathit{Pat}}) = \overline{e}}{\text{match}((\overline{v}), (\overline{\mathit{Pat}})) = \bigcup \overline{e}}\ \textsc{E-MatchTup}$$

$$\boxed{\mathit{Pat} \leq \mathit{Pat}'}$$

$$\frac{}{\mathit{Pat} \leq \_}\ \textsc{SpecWild}$$

$$\frac{\mathit{Pat}_1 \leq \mathit{Pat}_2}{I \texttt{ as } \mathit{Pat}_1 \leq \mathit{Pat}_2}\ \textsc{SpecBind1} \qquad \frac{\mathit{Pat}_1 \leq \mathit{Pat}_2}{\mathit{Pat}_1 \leq I \texttt{ as } \mathit{Pat}_2}\ \textsc{SpecBind2}$$

$$\frac{C \leq C' \qquad \overline{\mathit{Pat}_1} \leq \overline{\mathit{Pat}_2}}{C\ \{\overline{V} = \overline{\mathit{Pat}_1}, \overline{V_3} = \overline{\mathit{Pat}_3}\} \leq C'\ \{\overline{V} = \overline{\mathit{Pat}_2}\}}\ \textsc{SpecClass}$$

Rule SpecClass: The notation $\overline{\mathit{Pat}} \leq \overline{\mathit{Pat}'}$ abbreviates $\mathit{Pat}_1 \leq \mathit{Pat}'_1 \cdots \mathit{Pat}_k \leq \mathit{Pat}'_k$.

$$\frac{\overline{\mathit{Pat}_1} \leq \overline{\mathit{Pat}_2}}{(\overline{\mathit{Pat}_1}) \leq (\overline{\mathit{Pat}_2})}\ \textsc{SpecTup}$$

$$\boxed{C \leq C'}$$

$$\frac{}{C \leq C}\ \textsc{SubRef}$$

$$\frac{C_1 \leq C_2 \qquad C_2 \leq C_3}{C_1 \leq C_3}\ \textsc{SubTrans}$$

$$\frac{(\texttt{<abstract>}\ \texttt{class}\ (\overline{Tn}\ Cn)(\overline{I_1} : \overline{T_1})\ \texttt{extends}\ (\overline{T}\ C)\ \ldots) \in BT(Bn)}{Bn.Cn \leq C}\ \textsc{SubExt}$$

2.4 Auxiliary Judgments

$$\boxed{\text{concrete}(C)}$$

$$\frac{(\texttt{class}\ \overline{Tn}\ Cn\ \ldots) \in BT(Bn)}{\text{concrete}(Bn.Cn)}\ \textsc{Concrete}$$

$$\boxed{\text{rep}(Ct(\overline{E_0})) = \{\overline{V} = \overline{E}\}}$$

$$\frac{\begin{array}{c}(\texttt{<<abstract>>}\ \texttt{class}\ \overline{Tn}\ Cn(\overline{I}:\overline{T_1})\ \texttt{<extends}\ Ct(\overline{E_0})\ \texttt{>of}\ \{\overline{Vn}:\overline{T_2}=\overline{E_2}\}) \in BT(Bn)\\ <\text{rep}(Ct(\overline{E_0}))=\{\overline{V}=\overline{E_1}\}>\end{array}}{\text{rep}((\overline{T}\ Bn.Cn)(\overline{E})) = [\overline{I}\mapsto\overline{E}][\overline{Tn}\mapsto\overline{T}]\{<\overline{V}=\overline{E_1},>Bn.\overline{Vn}=\overline{E_2}\}}\ \textsc{Rep}$$

Rule Rep: The notation $Bn.\overline{Vn} = \overline{E}$ abbreviates $Bn.Vn_1 = E_1, \ldots, Bn.Vn_k = E_k$.

3 Static Semantics

3.1 Preliminaries

Γ is a type environment, mapping identifiers to types. The metavariable Tm ranges over both types and marked types. The notation $\hat{Mt}$ denotes the type T equivalent to Mt, but with the # mark removed.

3.2 Blocks

$$\boxed{B\ \text{OK}}$$

$$\frac{\overline{Bn} \vdash \overline{Ood}\ \text{OK in}\ Bn}{\texttt{block}\ \ Bn = \texttt{blk extends}\ \overline{Bn}\ \overline{Ood}\ \texttt{end}\ \text{OK}}\ \textsc{BlockOK}$$

Rule BlockOK: The notation $\overline{Bn} \vdash \overline{Ood}$ OK in Bn abbreviates $\overline{Bn} \vdash Ood_1$ OK in $Bn \cdots \overline{Bn} \vdash Ood_k$ OK in Bn.

3.3 OO Declarations

$$\boxed{\overline{Bn} \vdash Ood\ \text{OK in}\ Bn}$$

$$\frac{\begin{array}{c}<Ct = \overline{Tn}\ Bn.Cn> \qquad <\Gamma;\overline{Tn} \vdash Ct(\overline{E})\ \text{OK}>\\ \overline{Tn} \vdash \overline{T}\ \text{OK} \qquad \overline{Tn} \vdash \overline{T_0}\ \text{OK} \qquad \Gamma = \{(\overline{I},\overline{T})\} \qquad \Gamma;\overline{Tn} \vdash \overline{E_0}:\overline{T_1} \qquad \overline{T_1} \leq \overline{T_0}\\ \overline{Bn} \vdash Bn.Cn\ \text{transExtended} \qquad \text{concrete}(Bn.Cn) \Rightarrow \overline{Bn} \vdash \text{funs-have-ldefault-for}\ Bn.Cn\end{array}}{\overline{Bn} \vdash \texttt{<<abstract>>}\ \texttt{class}\ \overline{Tn}\ Cn(\overline{I}:\overline{T})\ \texttt{<extends}\ Ct(\overline{E})\ \texttt{>}\ \texttt{of}\ \{\overline{Vn}:\overline{T_0}=\overline{E_0}\}\ \text{OK in}\ Bn}\ \textsc{ClassOK}$$

Rule ClassOK: The notation $\overline{Tn} \vdash \overline{T}$ OK abbreviates $\overline{Tn} \vdash T_1$ OK $\cdots$ $\overline{Tn} \vdash T_k$ OK. The notation $(\overline{I},\overline{T})$ abbreviates $(I_1,T_1),\ldots,(I_k,T_k)$. The notation $\Gamma;\overline{Tn} \vdash \overline{E}:\overline{T}$ abbreviates $\Gamma;\overline{Tn} \vdash E_1 : T_1 \cdots \Gamma;\overline{Tn} \vdash E_k : T_k$. The notation $\overline{T_1} \leq \overline{T_0}$ abbreviates $T_{11} \leq T_{01} \cdots T_{1k} \leq T_{0k}$.

$$\frac{\overline{Tn} \vdash \hat{Mt}\ \text{OK} \qquad \overline{Tn} \vdash T\ \text{OK} \qquad \text{CP}(Bn.Fn) = Bn'.Cn \qquad Bn = Bn' \lor \overline{Bn} \vdash Bn.Fn\ \text{has-gdefault}}{\overline{Bn} \vdash \texttt{fun}\ \overline{Tn}\ Fn : Mt \rightarrow T\ \text{OK in}\ Bn}\ \textsc{FunOK}$$

$$\frac{\begin{array}{c}(\texttt{fun}\ \overline{Tn'}\ Fn : Mt \to T) \in BT(Bn') \\ \text{matchType}([\overline{Tn'} \mapsto \overline{Tn}]\hat{M}t, Pat) = (\Gamma, T_0) \qquad \Gamma; \overline{Tn} \vdash E : T' \qquad T' \leq [\overline{Tn'} \mapsto \overline{Tn}]T \\ \overline{Bn} \vdash Bn'.Fn \text{ extended} \qquad Bn; \overline{Bn} \vdash \texttt{extend fun}_{Mn}\ \overline{Tn}\ Bn'.Fn\ Pat = E \text{ unambiguous}\end{array}}{\overline{Bn} \vdash \texttt{extend fun}_{Mn}\ \overline{Tn}\ Bn'.Fn\ Pat = E \text{ OK in } Bn} \text{ CaseOK}$$

3.4 Types

$$\boxed{\overline{Tn} \vdash T \text{ OK}}$$

$$\frac{Tn \in \overline{Tn}}{\overline{Tn} \vdash Tn \text{ OK}} \text{ TVarOK}$$

$$\frac{(\texttt{<abstract> class}\ \overline{Tn_0}\ Cn \ldots) \in BT(Bn) \qquad \overline{Tn} \vdash \overline{T} \text{ OK} \qquad |\overline{Tn_0}| = |\overline{T}|}{\overline{Tn} \vdash \overline{T}\ Bn.Cn \text{ OK}} \text{ ClassTypeOK}$$

$$\frac{\overline{Tn} \vdash T_1 \text{ OK} \qquad \overline{Tn} \vdash T_2 \text{ OK}}{\overline{Tn} \vdash T_1 \to T_2 \text{ OK}} \text{ FunTypeOK}$$

$$\frac{\overline{Tn} \vdash T_1 \text{ OK} \qquad \cdots \qquad \overline{Tn} \vdash T_k \text{ OK}}{\overline{Tn} \vdash T_1 * \cdots * T_k \text{ OK}} \text{ TupTypeOK}$$

3.5 Subtyping

$$\boxed{T \leq T'}$$

$$\frac{}{T \leq T} \text{ SubTRef}$$

$$\frac{T_1 \leq T_2 \qquad T_2 \leq T_3}{T_1 \leq T_3} \text{ SubTTrans}$$

$$\frac{(\texttt{<abstract> class}\ \overline{Tn}\ Cn(\overline{I_1} : \overline{T_1})\ \texttt{extends}\ Ct \ldots) \in BT(Bn)}{\overline{T}\ Bn.Cn \leq [\overline{Tn} \mapsto \overline{T}]Ct} \text{ SubTExt}$$

$$\frac{T_1' \leq T_1 \qquad T_2 \leq T_2'}{T_1 \to T_2 \leq T_1' \to T_2'} \text{ SubTFun}$$

$$\frac{T_1 \leq T_1' \qquad \cdots \qquad T_k \leq T_k'}{T_1 * \cdots * T_k \leq T_1' * \cdots * T_k'} \text{ SubTTup}$$

3.6 Patterns

$$\boxed{\text{matchType}(T, Pat) = (\Gamma, T')}$$

$$\frac{}{\text{matchType}(T, \_) = (\{\}, T)} \text{ T-MatchWild}$$

$$\frac{\text{matchType}(T, Pat) = (\Gamma, T')}{\text{matchType}(T, I \texttt{ as } Pat) = (\Gamma \cup \{(I, T')\}, T')} \text{ T-MatchBind}$$

$$\frac{C \le C' \qquad \text{repType}(\overline{T}\ C) = \{\overline{V} : \overline{T_0}\} \qquad \text{matchType}(\overline{T_0}, \overline{Pat}) = (\overline{\Gamma}, \overline{T_1})}{\text{matchType}((\overline{T}\ C'), C\ \{\overline{V} = \overline{Pat}\}) = (\bigcup \overline{\Gamma}, (\overline{T}\ C))} \text{ T-MatchClass}$$

Rule T-MatchClass: The notation $\text{matchType}(\overline{T_0}, \overline{Pat}) = (\overline{\Gamma}, \overline{T_1})$ abbreviates $\text{matchType}(T_1, Pat_1) = (\Gamma_1, T_1') \cdots \text{matchType}(T_k, Pat_k) = (\Gamma_k, T_k')$.

$$\frac{\text{matchType}(T_1, Pat_1) = (\Gamma_1, T_1') \qquad \cdots \qquad \text{matchType}(T_k, Pat_k) = (\Gamma_k, T_k')}{\text{matchType}(T_1 * \cdots * T_k, (Pat_1, \ldots, Pat_k)) = \qquad (\Gamma_1 \cup \ldots \cup \Gamma_k, T_1' * \cdots * T_k')} \text{ T-MatchTup}$$

3.7 Expressions

$$\boxed{\Gamma; \overline{Tn} \vdash E : T}$$

$$\frac{(I, T) \in \Gamma}{\Gamma; \overline{Tn} \vdash I : T} \text{ T-Id}$$

$$\frac{(\texttt{fun}\ \overline{Tn_0}\ Fn : Mt \to T) \in BT(Bn) \qquad \overline{Tn} \vdash \overline{T_0}\ \text{OK}}{\Gamma; \overline{Tn} \vdash \overline{T_0}\ Bn.Fn : [\overline{Tn_0} \mapsto \overline{T_0}](\hat{Mt} \to T)} \text{ T-Fun}$$

$$\frac{\Gamma; \overline{Tn} \vdash E_1 : T_2 \to T \qquad \Gamma; \overline{Tn} \vdash E_2 : T_2' \qquad T_2' \le T_2}{\Gamma; \overline{Tn} \vdash E_1\ E_2 : T} \text{ T-App}$$

$$\frac{\Gamma; \overline{Tn} \vdash Ct(\overline{E})\ \text{OK} \qquad Ct = (\overline{T}\ C) \qquad \text{concrete}(C)}{\Gamma; \overline{Tn} \vdash Ct(\overline{E}) : Ct} \text{ T-New}$$

$$\frac{\Gamma; \overline{Tn} \vdash E_1 : T_1 \qquad \cdots \qquad \Gamma; \overline{Tn} \vdash E_k : T_k}{\Gamma; \overline{Tn} \vdash (E_1, \ldots, E_k) : T_1 * \cdots * T_k} \text{ T-Tup}$$

$$\frac{\begin{array}{c}\overline{Tn} \vdash Ct\ \text{OK} \\ Ct = (\overline{T_0}\ C) \qquad \text{concrete}(C) \qquad \text{repType}(Ct) = \{\overline{V} : \overline{T}\} \qquad \Gamma; \overline{Tn} \vdash \overline{E} : \overline{T_1} \qquad \overline{T_1} \le \overline{T}\end{array}}{\Gamma; \overline{Tn} \vdash Ct\ \{\overline{V} = \overline{E}\} : Ct} \text{ T-Rep}$$

3.8 Constructor Calls

$$\boxed{\Gamma; \overline{Tn} \vdash Ct(\overline{E}) \text{ OK}}$$

$$\frac{\begin{array}{c}\overline{Tn} \vdash Ct \text{ OK} \qquad Ct = (\overline{T_0}\ Bn.Cn) \\ (\texttt{<abstract> class}\ \overline{Tn_0}\ Cn(\overline{I} : \overline{T})\ \ldots) \in BT(Bn) \qquad \Gamma; \overline{Tn} \vdash \overline{E} : \overline{T_1} \qquad \overline{T_1} \leq [\overline{Tn_0} \mapsto \overline{T_0}]\overline{T}\end{array}}{\Gamma; \overline{Tn} \vdash Ct(\overline{E}) \text{ OK}} \text{ T-SUPER}$$

3.9 Class Representation Types

$$\boxed{\text{repType}(Ct) = \{\overline{V} : \overline{T}\}}$$

$$\frac{\begin{array}{c}(\texttt{<<abstract>> class}\ \overline{Tn}\ Cn(\overline{I} : \overline{T_1})\ \texttt{<extends}\ Ct(\overline{E_0}) > \texttt{of}\ \{\overline{Vn} : \overline{T_2} = \overline{E_2}\}) \in BT(Bn) \\ < \text{repType}(Ct) = \{\overline{V} : \overline{T_3}\} > \end{array}}{\text{repType}(\overline{T}\ Bn.Cn) = [\overline{Tn} \mapsto \overline{T}]\{< \overline{V} : \overline{T_3}, > Bn.\overline{Vn} : \overline{T_2}\}} \text{ REPTYPE}$$

Rule REPTYPE: The notation $Bn.\overline{Vn} : \overline{T}$ abbreviates $Bn.Vn_1 : T_1, \ldots, Bn.Vn_k : T_k$.

3.10 Completeness Checking

3.10.1 Checking for Local and Global Default Cases

$$\boxed{\overline{Bn} \vdash \text{funs-have-ldefault-for } C}$$

$$\frac{\forall F, C'.[(\overline{Bn} \vdash F \text{ extended} \wedge \text{CP}(F) = C' \wedge C \leq C') \Rightarrow \overline{Bn} \vdash F \text{ has-default-for } C]}{\overline{Bn} \vdash \text{funs-have-ldefault-for } C} \text{ LDEFAULT}$$

$$\boxed{\overline{Bn} \vdash F \text{ has-gdefault}}$$

$$\frac{\text{CP}(F) = C \qquad \overline{Bn} \vdash F \text{ has-default-for } C}{\overline{Bn} \vdash F \text{ has-gdefault}} \text{ GDEFAULT}$$

$$\boxed{\overline{Bn} \vdash F \text{ has-default-for } C}$$

$$\frac{\begin{array}{c}(\texttt{fun}\ \overline{Tn}\ Fn : Mt \to T) \in BT(Bn) \qquad \text{defaultPat}(Mt, C) = Pat \\ (\texttt{extend fun}_{Mn}\ \overline{Tn_0}\ Bn.Fn\ Pat' = E) \in BT(Bn') \qquad Pat \leq Pat' \qquad Bn' \in \overline{Bn}\end{array}}{\overline{Bn} \vdash Bn.Fn \text{ has-default-for } C} \text{ DEFAULT}$$

3.10.2 Generating the Default Pattern

$$\boxed{\text{defaultPat}(Mt, C) = Pat}$$

$$\frac{\text{defaultPat}(Mt, C, d) = Pat}{\text{defaultPat}(Mt, C) = Pat} \text{ DEFPAT}$$

Rule DEFPAT: The metavariable d ranges over nonnegative integers. It represents the "depth" of the resulting default pattern. For example, a default pattern of depth 0 is simply the wildcard, while a default pattern of depth 1 for a class type has the form $C\_$. The higher the depth, the more precise the check for local/global defaults is. This type system does not compute the best depth to use, instead choosing it non-deterministically. It is straightforward to find the appropriately precise depth — it is the maximum depth of any pattern in an available case of the function being checked.

$$\boxed{\mathrm{defaultPat}(Tm, C, d) = Pat}$$

The metavariable Tm ranges over both types and marked types.

$$\frac{}{\mathrm{defaultPat}(Tm, C, 0) = \_}\ \textsc{DefZero}$$

$$\frac{d > 0}{\mathrm{defaultPat}(Tn, C, d) = \_}\ \textsc{DefTypeVar}$$

$$\frac{\mathrm{repType}(\overline{T}\ C') = \{\overline{V} : \overline{T_0}\} \qquad \mathrm{defaultPat}(\overline{T_0}, C, d-1) = \overline{Pat} \qquad d > 0}{\mathrm{defaultPat}((\overline{T}\ C'), C, d) = (C'\ \{\overline{V} = \overline{Pat}\})}\ \textsc{DefClassType}$$

Rule DefClassType: The notation $\mathrm{defaultPat}(\overline{T_0}, C, d-1) = \overline{Pat}$ abbreviates $\mathrm{defaultPat}(T_1, C, d-1) = Pat_1 \cdots \mathrm{defaultPat}(T_k, C, d-1) = Pat_k$.

$$\frac{\mathrm{repType}(\overline{T}\ C) = \{\overline{V} : \overline{T_0}\} \qquad \mathrm{defaultPat}(\overline{T_0}, C, d-1) = \overline{Pat} \qquad d > 0}{\mathrm{defaultPat}(\#(\overline{T}\ C'), C, d) = (C\ \{\overline{V} = \overline{Pat}\})}\ \textsc{DefCPClassType}$$

$$\frac{\mathrm{defaultPat}(Tm_1, C, d-1) = Pat_1 \qquad \cdots \qquad \mathrm{defaultPat}(Tm_k, C, d-1) = Pat_k \qquad d > 0}{\mathrm{defaultPat}(Tm_1 * \ldots * Tm_k, C, d) = (Pat_1, \ldots, Pat_k)}\ \textsc{DefTupType}$$

$$\frac{d > 0}{\mathrm{defaultPat}(T_1 \to T_2, C, d) = \_}\ \textsc{DefFunType}$$

3.11 Ambiguity Checking

3.11.1 The Top-Level Rule

$$\boxed{Bn; \overline{Bn} \vdash \texttt{extend fun} \ldots \text{ unambiguous}}$$

$$\frac{\begin{array}{c}\overline{Bn} \vdash \texttt{extend fun}_{Mn}\ \overline{Tn}\ Bn'.Fn\ Pat = E \text{ unambiguous in } Bn \\ (\texttt{fun}\ \overline{Tn'}\ Fn : Mt \to T) \in BT(Bn') \qquad \mathrm{CP}(Mt, Pat) = Bn''.Cn \qquad Bn = Bn' \vee Bn = Bn''\end{array}}{Bn; \overline{Bn} \vdash \texttt{extend fun}_{Mn}\ \overline{Tn}\ Bn'.Fn\ Pat = E \text{ unambiguous}}\ \textsc{Amb}$$

3.11.2 Ambiguity With Available Cases

$$\boxed{\overline{Bn} \vdash \texttt{extend fun} \ldots \text{ unambiguous in } Bn}$$

$$\frac{\begin{array}{c}\forall Bn' \in \overline{Bn}.\forall(\texttt{extend fun}_{Mn'}\ \overline{Tn_1}\ F\ Pat' = E') \in BT(Bn'). \\ \forall Pat_0.[(Pat \cap Pat' = Pat_0 \wedge Bn.Mn \neq Bn'.Mn') \Rightarrow \\ \exists Bn'' \in \overline{Bn}.\exists(\texttt{extend fun}_{Mn''}\ \overline{Tn_2}\ F\ Pat'' = E'') \in BT(Bn''). \\ (Pat_0 \leq Pat'' \wedge Pat'' \leq Pat \wedge Pat'' \leq Pat' \wedge (Pat \not\leq Pat'' \vee Pat' \not\leq Pat''))]\end{array}}{\overline{Bn} \vdash \texttt{extend fun}_{Mn}\ \overline{Tn}\ F\ Pat = E \text{ unambiguous in } Bn} \text{ BLAMB}$$

Rule BLAMB: This rule ensures that a function case is not ambiguous with any other function cases declared in $\overline{Bn}$: for each such case that has a non-empty *intersection* with the current case's pattern, there must exist a *resolving* case. The resolving case must cover the intersection, be at least as specific as the other two cases, and be strictly more specific than one of them.

3.11.3 Pattern Intersection

$$\boxed{Pat_1 \cap Pat_2 = Pat}$$

$$\frac{}{\_ \cap Pat = Pat} \text{ PATINTWILD}$$

$$\frac{Pat_1 \cap Pat_2 = Pat}{I \texttt{ as } Pat_1 \cap Pat_2 = Pat} \text{ PATINTBIND}$$

$$\frac{C \leq C' \qquad \overline{Pat_1} \cap \overline{Pat_2} = \overline{Pat}}{C\ \{\overline{V} = \overline{Pat_1}, \overline{V_3} = \overline{Pat_3}\} \cap C'\ \{\overline{V} = \overline{Pat_2}\} = C\ \{\overline{V} = \overline{Pat}, \overline{V_3} = \overline{Pat_3}\}} \text{ PATINTCLASS}$$

Rule PATINTCLASS: The notation $\overline{Pat_1} \cap \overline{Pat_2} = \overline{Pat}$ abbreviates $Pat'_1 \cap Pat''_1 = Pat_1 \cdots Pat'_k \cap Pat''_k = Pat_k$.

$$\frac{\overline{Pat_1} \cap \overline{Pat_2} = \overline{Pat}}{(\overline{Pat_1}) \cap (\overline{Pat_2}) = (\overline{Pat})} \text{ PATINTTUP}$$

$$\frac{Pat_2 \cap Pat_1 = Pat}{Pat_1 \cap Pat_2 = Pat} \text{ PATINTREV}$$

3.12 Block Extension

$$\boxed{\overline{Bn} \vdash Bn.Cn \text{ transExtended}}$$

$$\frac{\begin{array}{c}(\texttt{<<abstract>>}\ \texttt{class}\ (\overline{Tn}\ Cn)(\overline{I} : \overline{T})\texttt{<extends}\ (\overline{T_0}\ C)(\overline{E}) > \ldots) \in BT(Bn) \\ Bn \in \overline{Bn} \qquad < \overline{Bn} \vdash C \text{ transExtended} >\end{array}}{\overline{Bn} \vdash Bn.Cn \text{ transExtended}} \text{ CLASSTRANSEXT}$$

$$\boxed{\overline{Bn} \vdash F \text{ extended}}$$

$$\frac{Bn \in \overline{Bn}}{\overline{Bn} \vdash Bn.Fn \text{ extended}} \text{ FUNEXT}$$

3.13 Accessing the CP

3.13.1 The CP of a Function's Argument Type

$$\boxed{\mathrm{CP}(F) = C}$$

$$\frac{(\texttt{fun}\ \overline{Tn}\ Fn : Mt \to T) \in BT(Bn) \qquad \mathrm{CP}(Mt) = C}{\mathrm{CP}(Bn.Fn) = C}\ \textsc{CPFun}$$

$$\boxed{\mathrm{CP}(Mt) = C}$$

$$\frac{}{\mathrm{CP}(\texttt{\#}\ \overline{T}\ C) = C}\ \textsc{CPClass}$$

$$\frac{\mathrm{CP}(Mt) = C}{\mathrm{CP}(T_1 * \cdots * T_{i-1} * Mt * T_{i+1} * \cdots * T_k) = C}\ \textsc{CPTup}$$

3.13.2 The CP of a Pattern

$$\boxed{\mathrm{CP}(Mt, Pat) = C}$$

$$\frac{\mathrm{CP}(Mt, Pat) = C}{\mathrm{CP}(Mt, I\ \texttt{as}\ Pat) = C}\ \textsc{CPBindPat}$$

$$\frac{\mathrm{CP}(Mt, Pat_i) = C}{\mathrm{CP}(T_1 * \cdots * T_{i-1} * Mt * T_{i+1} * \cdots * T_k, (Pat_1, \ldots, Pat_k)) = C}\ \textsc{CPTupPat}$$

$$\frac{}{\mathrm{CP}(\# Ct, C\ \{\overline{V} = \overline{Pat}\}) = C}\ \textsc{CPClassPat}$$

3.13.3 The CP of a Value

$$\boxed{\mathrm{CP}(Mt, v) = C}$$

These rules are used only in the proof of progress.

$$\frac{\mathrm{CP}(Mt, v_i) = C}{\mathrm{CP}(T_1 * \cdots * T_{i-1} * Mt * T_{i+1} * \cdots * T_k, (v_1, \ldots, v_k)) = C}\ \textsc{CPTupVal}$$

$$\frac{}{\mathrm{CP}(\# Ct, (\overline{T}\ C)\ \{\overline{V} = \overline{v}\}) = C}\ \textsc{CPInstance}$$

4 Subject Reduction

4.1 Shared Preliminaries and Lemmas

These preliminaries and lemmas are also used in the progress proof in section 5.

As in the inference rules, we assume a global block table BT. We further assume that for each $Bn \in \mathrm{dom}(BT)$ we have $BT(Bn)$ OK. The empty sequence is denoted $\bullet$. The notation $\vdash E : T$ is shorthand for $\{\}; \bullet \vdash E : T$.

Lemma 4.1 If $\overline{Tn} \vdash T$ OK, then all type variables in T are in $\overline{Tn}$.
Proof By (strong) induction on the depth of the derivation of $\overline{Tn} \vdash T$ OK. Case analysis on the last rule used in the derivation. For TVarOK, T has the form Tn and the premise ensures that $Tn \in \overline{Tn}$. All other cases are easily proven by induction.

Lemma 4.2 If $\overline{Tn} \vdash T$ OK and $|\overline{Tn}| = |\overline{T}|$ and $\overline{Tn'} \vdash \overline{T}$ OK, then $\overline{Tn'} \vdash [\overline{Tn} \mapsto \overline{T}]T$ OK.
Proof By (strong) induction on the depth of the derivation of $\overline{Tn} \vdash T$ OK. Case analysis on the last rule used in the derivation. For TVarOK, T has the form Tn and the premise ensures that $Tn \in \overline{Tn}$. Therefore $[\overline{Tn} \mapsto \overline{T}]T$ is some T_0 in $\overline{T}$. By assumption $\overline{Tn'} \vdash T_0$ OK so the result follows. All other cases are easily proven by induction.

Lemma 4.3 If $(\overline{T}\ C) \leq T$, then T has the form $(\overline{T_1}\ C')$.
Proof By (strong) induction on the depth of the derivation of $(\overline{T}\ C) \leq T$. Case analysis of the last rule used in the derivation.

- Case SubTRef. Then $T = (\overline{T}\ C)$.
- Case SubTTrans. Then $(\overline{T}\ C) \leq T'$ and $T' \leq T$. By induction T' has the form $(\overline{T_2}\ C'')$. Then by induction again, T has the form $(\overline{T_1}\ C')$.
- Case SubTExt. Then T has the form $[\overline{Tn} \mapsto \overline{T}]Ct$, which is also of the form $(\overline{T_1}\ C')$.

Lemma 4.4 If $(\overline{T}\ C) \leq (\overline{T_1}\ C')$, then $\overline{T} = \overline{T_1}$.
Proof By (strong) induction on the depth of the derivation of $(\overline{T}\ C) \leq (\overline{T_1}\ C')$. Case analysis of the last rule used in the derivation.

- Case SubTRef. Then $(\overline{T}\ C) = (\overline{T_1}\ C')$, so $\overline{T} = \overline{T_1}$.
- Case SubTTrans. Then $(\overline{T}\ C) \leq T$ and $T \leq (\overline{T_1}\ C')$. By Lemma 4.3, T has the form $(\overline{T_2}\ C'')$. Then by induction we have $\overline{T} = \overline{T_2}$ and $\overline{T_2} = \overline{T_1}$, so $\overline{T} = \overline{T_1}$.
- Case SubTExt. Then $C = Bn.Cn$ and $(\overline{T_1}\ C') = [\overline{Tn} \mapsto \overline{T}](\overline{T_2}\ C')$ and (`<abstract> class` $\overline{Tn}$ $Cn(I_1 : T_1, \ldots, I_m : T_m)$ `extends` $(\overline{T_2}\ C')\ \ldots) \in BT(Bn)$. By ClassOK, we have $\overline{T_2} = \overline{Tn}$. Therefore $(\overline{T_1}\ C') = [\overline{Tn} \mapsto \overline{T}](\overline{Tn}\ C') = (\overline{T}\ C')$. Therefore $\overline{T} = \overline{T_1}$.

Lemma 4.5 If $(\overline{T}\ C) \leq (\overline{T_1}\ C')$ then $C \leq C'$.
Proof By (strong) induction on the depth of the derivation of $(\overline{T}\ C) \leq (\overline{T_1}\ C')$. Case analysis of the last rule used in the derivation.

- Case SubTRef. Then $(\overline{T}\ C) = (\overline{T_1}\ C')$, so $C = C'$. Then the result holds by SubRef.
- Case SubTTrans. Then $(\overline{T}\ C) \leq T$ and $T \leq (\overline{T_1}\ C')$. By Lemma 4.3 T has the form $(\overline{T_2}\ C'')$. Then by induction we have that $C \leq C''$ and $C'' \leq C'$. Therefore the result follows by SubTrans.

- Case SubTExt. Then $C = Bn.Cn$ and (<abstract> `class` $\overline{Tn}$ $Cn(\overline{I_0} : \overline{T_0})$ `extends` $(\overline{T_2}\ C')\ \ldots) \in BT(Bn)$. Then the result follows by SubExt.

Lemma 4.6 If $T \leq T_1 * \cdots * T_k$, then T has the form $T'_1 * \cdots * T'_k$, where for all $1 \leq i \leq k$ we have $T'_i \leq T_i$.
Proof By (strong) induction on the depth of the derivation of $T \leq T_1 * \cdots * T_k$. Case analysis of the last rule used in the derivation.

- Case SubTRef. Then $T = T_1 * \cdots * T_k$. By SubTRef, for all $1 \leq i \leq k$ we have $T_i \leq T_i$, so the result follows.

- Case SubTTrans. Then $T \leq T'$ and $T' \leq T_1 * \cdots * T_k$. By induction T' has the form $T''_1 * \cdots * T''_k$, where for all $1 \leq i \leq k$ we have $T''_i \leq T_i$. Then by induction again, T has the form $T'_1 * \cdots * T'_k$, where for all $1 \leq i \leq k$ we have $T'_i \leq T''_i$. Then by SubTTrans, for all $1 \leq i \leq k$ we have $T'_i \leq T_i$.

- Case SubTTup. Then T has the form $T'_1 * \cdots * T'_k$, where for all $1 \leq i \leq k$ we have $T'_i \leq T_i$.

Lemma 4.7 If $Bn.Cn \leq Bn'.Cn'$ and $\overline{Tn_0} \vdash (\overline{T}\ Bn.Cn)$ OK then (1) $(\overline{T}\ Bn.Cn) \leq (\overline{T}\ Bn'.Cn')$; and (2) $\overline{Tn_0} \vdash (\overline{T}\ Bn'.Cn')$ OK.
Proof By (strong) induction on the depth of the derivation of $Bn.Cn \leq Bn'.Cn'$. Case analysis of the last rule used in the derivation.

- Case SubRef. Then $Bn'.Cn' = Bn.Cn$. Then condition 1 follows from SubTRef, and condition 2 follows by assumption.

- Case SubTrans. Then $Bn.Cn \leq Bn''.Cn''$ and $Bn''.Cn'' \leq Bn'.Cn'$. By induction we have $(\overline{T}\ Bn.Cn) \leq (\overline{T}\ Bn''.Cn'')$ and $\overline{Tn_0} \vdash (\overline{T}\ Bn''.Cn'')$ OK. Then by induction again we have $(\overline{T}\ Bn''.Cn'') \leq (\overline{T}\ Bn'.Cn')$ and $\overline{Tn_0} \vdash (\overline{T}\ Bn'.Cn')$ OK. Therefore condition 2 is shown, and condition 1 follows from SubTTrans.

- Case SubExt. Then (<abstract> `class` $\overline{Tn}$ $Cn(\overline{I_0} : \overline{T_0})$ `extends` $(\overline{T'}\ Bn'.Cn')(\overline{E})\ \ldots) \in BT(Bn)$. Then by ClassOK we have $\overline{T'} = \overline{Tn}$. Since $\overline{Tn_0} \vdash (\overline{T}\ Bn.Cn)$ OK, by ClassTypeOK we have $|\overline{Tn}| = |\overline{T}|$ and $\overline{Tn_0} \vdash \overline{T}$ OK. Therefore by SubTExt we have $(\overline{T}\ Bn.Cn) \leq [\overline{Tn} \mapsto \overline{T}](\overline{Tn}\ Bn'.Cn')$. Since $[\overline{Tn} \mapsto \overline{T}](\overline{Tn}\ Bn'.Cn') = (\overline{T}\ Bn'.Cn')$, condition 1 is shown. Also by ClassOK $\overline{Tn} \vdash (\overline{Tn}\ Bn'.Cn')(\overline{E})$ OK, so by T-Super we have have $\overline{Tn} \vdash (\overline{Tn}\ Bn'.Cn')$ OK. Therefore by Lemma 4.2 we have $\overline{Tn_0} \vdash (\overline{T}\ Bn'.Cn')$ OK, so condition 2 is shown.

Lemma 4.8 If $\overline{Tn} \vdash Ct$ OK then repType(Ct) is well-defined and has the form $\{\overline{V_0} : \overline{T_0}\}$.
Proof Let $Ct = (\overline{T}\ Bn.Cn)$. We prove this lemma by induction on the length of the longest path in the superclass graph from $Bn.Cn$ (in other words, the number of non-trivial superclasses of $Bn.Cn$). By ClassTypeOK we have $\overline{Tn} \vdash \overline{T}$ OK and (<abstract> `class` $\overline{Tn_0}$ $Cn(\overline{I_1} : \overline{T_1})$ <<`extends` $Ct'(\overline{E})$ >> `of` $\{\overline{Vn} : \overline{T_2} = \overline{E_2}\}) \in BT(Bn)$ and $|\overline{Tn_0}| = |\overline{T}|$. There are two cases to consider.

- The length of the longest path in the superclass graph from $Bn.Cn$ is 0. Then $Bn.Cn$ has no non-trivial superclasses, so the `extends` clause in the declaration of $Bn.Cn$ is absent. Then by RepType we have repType(Ct) $= [\overline{Tn_0} \mapsto \overline{T}]\{Bn.\overline{Vn} : \overline{T_2}\}$, so the result follows.

- The length of the longest path in the superclass graph from $Bn.Cn$ is $i > 0$. Then $Bn.Cn$ has at least one non-trivial superclass, so the `extends` clause in the declaration of $Bn.Cn$ is present. Then by ClassOK we have $\overline{Tn_0} \vdash Ct'(\overline{E})$ OK, so by T-Super we have $\overline{Tn_0} \vdash Ct'$ OK. Since Ct' must have the form $(\overline{T_1}\ Bn'.Cn')$, where the length of the longest path in the superclass graph from $Bn'.Cn'$ is $i-1$, by induction we have that repType(Ct') has the form $\{\overline{V_0} : \overline{T_0}\}$. Then by RepType we have repType(Ct) $= [\overline{Tn_0} \mapsto \overline{T}]\{\overline{V_0} : \overline{T_0}, Bn.\overline{Vn} : \overline{T_2}\}$, so the result follows.

Lemma 4.9 If $\overline{Tn} \vdash Ct$ OK and $Ct \leq Ct'$, then $\overline{Tn} \vdash Ct'$ OK.
Proof By (strong) induction on the depth of the derivation of $Ct \leq Ct'$. Case analysis of the last rule used in the derivation.

- Case SubTRef. Then $Ct = Ct'$, so the result follows by assumption.
- Case SubTTrans. Then $Ct \leq T$ and $T \leq Ct'$. By Lemma 4.3 T has the form Ct''. Therefore by induction we have $\overline{Tn} \vdash Ct''$ OK, and by induction again we have $\overline{Tn} \vdash Ct'$ OK.
- Case SubTExt. Then $Ct = (\overline{T}\ Bn.Cn)$ and $Ct' = [\overline{Tn_0} \mapsto \overline{T}]Ct''$ and (<abstract> `class` $\overline{Tn_0}$ $Cn(\overline{I_0} : \overline{T_0})$ `extends` $Ct''(\overline{E})$ ...) $\in BT(Bn)$. By ClassOK we have $\overline{Tn_0} \vdash Ct''(\overline{E})$ OK, so by T-Super we have $\overline{Tn_0} \vdash Ct''$ OK. Since $\overline{Tn} \vdash Ct$ OK, by ClassTypeOK we have $\overline{Tn} \vdash \overline{T}$ OK. Therefore by Lemma 4.2 we have $\overline{Tn} \vdash [\overline{Tn_0} \mapsto \overline{T}]Ct''$ OK.

Lemma 4.10 If repType$(Ct) = \{\overline{V} : \overline{T}\}$ and $\overline{Tn} \vdash Ct$ OK, then $\overline{Tn} \vdash \overline{T}$ OK.
Proof By induction on the depth of the derivation of repType$(Ct) = T$. Then by RepType $Ct = (\overline{T_0}\ Bn.Cn)$ and $\{\overline{V} : \overline{T}\} = [\overline{Tn_0} \mapsto \overline{T_0}]\{< \overline{V_1} : \overline{T_1}, > Bn.\overline{Vn} : \overline{T_2}\}$ and (<<abstract>> `class` $\overline{Tn_0}$ $Cn(\overline{I_0} : \overline{T_0})$ <`extends` $Ct'(\overline{E})$ > `of` $\{\overline{Vn} : \overline{T_2} = \overline{E_2}\}) \in BT(Bn)$ and < repType$(Ct') = \{\overline{V_1} : \overline{T_1}\}$. By ClassOK we have < $\overline{Tn_0} \vdash Ct'(\overline{E})$ OK >, so by T-Super we have < $\overline{Tn_0} \vdash Ct'$ OK >. Then by induction we have have < $\overline{Tn_0} \vdash \overline{T_1}$ OK. Also by ClassOK we have $\overline{Tn_0} \vdash \overline{T_2}$ OK. Since $\overline{Tn} \vdash Ct$ OK, by ClassTypeOK we have that $\overline{Tn} \vdash \overline{T_0}$ OK. Therefore by Lemma 4.2 we have < $\overline{Tn} \vdash [\overline{Tn_0} \mapsto \overline{T_0}]\overline{T_1}$ OK > and $\overline{Tn} \vdash [\overline{Tn_0} \mapsto \overline{T_0}]\overline{T_2}$ OK, so the result follows.

Lemma 4.11 If repType$(Ct) = \{\overline{V} : \overline{T}\}$ and $|\overline{Tn}| = |\overline{T}|$, then repType$([\overline{Tn} \mapsto \overline{T}]Ct) = [\overline{Tn} \mapsto \overline{T}]\{\overline{V} : \overline{T}\}$.
Proof By induction on the depth of the derivation of repType$(Ct) = \{\overline{V} : \overline{T}\}$. Then by RepType $Ct = (\overline{T_0}\ Bn.Cn)$ and $\{\overline{V} : \overline{T}\} = [\overline{Tn_0} \mapsto \overline{T_0}]\{< \overline{V_1} : \overline{T_1}, > Bn.\overline{Vn} : \overline{T_2}\}$ and (<<abstract>> `class` $\overline{Tn_0}$ $Cn(\overline{I_4} : \overline{T_4})$ <`extends` $Ct'(\overline{E})$ > `of` $\{\overline{Vn} : \overline{T_2} = \overline{E_2}\}) \in BT(Bn)$ and < repType$(Ct') = \{\overline{V_1} : \overline{T_1}\}$ >. Therefore by RepType we have repType$([\overline{Tn} \mapsto \overline{T}](\overline{T_0}\ Bn.Cn)) = [\overline{Tn_0} \mapsto [\overline{Tn} \mapsto \overline{T}]\overline{T_0}]\{< \overline{V_1} : \overline{T_1}, > Bn.\overline{Vn} : \overline{T_2}\}$. By ClassOK we have < $\overline{Tn_0} \vdash Ct'(\overline{E})$ OK >, so by T-Super we have < $\overline{Tn_0} \vdash Ct'$ OK >. Then by Lemma 4.10 we have < $\overline{Tn_0} \vdash \overline{T_1}$ OK >, so by Lemma 4.1 all type variables $\overline{T_1}$ are in $\overline{Tn_0}$. Also by ClassOK we have $\overline{Tn_0} \vdash \overline{T_2}$ OK, so by Lemma 4.1 all type variables in $\overline{T_2}$ are in $\overline{Tn_0}$. Therefore $[\overline{Tn_0} \mapsto [\overline{Tn} \mapsto \overline{T}]\overline{T_0}]\{\overline{V_1} : \overline{T_1}, Bn.\overline{Vn} : \overline{T_2}\}$ is equivalent to $[\overline{Tn} \mapsto \overline{T}][\overline{Tn_0} \mapsto \overline{T_0}]\{\overline{V_1} : \overline{T_1}, Bn.\overline{Vn} : \overline{T_2}\}$, so the result follows.

Lemma 4.12 If $\bullet \vdash Ct$ OK and $Ct \leq Ct'$ then repType$(Ct) = \{\overline{V_1} : \overline{T_1}, \overline{V_2} : \overline{T_2}\}$ and repType$(Ct') = \{\overline{V_1} : \overline{T_1}\}$.
Proof By induction on the depth of the derivation of $Ct \leq Ct'$. Case analysis of the last rule used in the derivation.

- Case SubTRef. Then $Ct = Ct'$. Since $\bullet \vdash Ct$ OK, by Lemma 4.8 we have that repType(Ct) is well-defined and has the form $\{\overline{V} : \overline{T}\}$. Therefore, repType$(Ct') = \{\overline{V} : \overline{T}\}$ as well, so the result follows.
- Case SubTTrans. Then $Ct \leq T$ and $T \leq Ct'$. By Lemma 4.3 T has the form Ct''. Then by Lemma 4.9 we have $\bullet \vdash Ct''$ OK and $\bullet \vdash Ct'$ OK. Therefore by induction we have repType(Ct) $= \{\overline{V_1} : \overline{T_1}, \overline{V_3} : \overline{T_3}, \overline{V_4} : \overline{T_4}\}$ and repType$(Ct'') = \{\overline{V_1} : \overline{T_1}, \overline{V_3} : \overline{T_3}\}$. By induction again we have repType$(Ct') = \{\overline{V_1} : \overline{T_1}\}$, so the result is shown.
- Case SubTExt. Then $Ct = (\overline{T}\ Bn.Cn)$ and $Ct' = [\overline{Tn} \mapsto \overline{T}]Ct''$ and (<abstract> `class` $\overline{Tn}$ $Cn(\overline{I_0} : \overline{T_0})$ `extends` $Ct''(\overline{E})$ `of` $\{\overline{Vn} : \overline{T_2} = \overline{E_2}\}) \in BT(Bn)$. Since $\bullet \vdash Ct$ OK, by Lemma 4.8 we have that repType(Ct) is well defined and has the form $\{\overline{V_3} : \overline{T_3}\}$. Then by RepType we have $\{\overline{V_3} : \overline{T_3}\} = [\overline{Tn} \mapsto \overline{T}]\{\overline{V_1} : \overline{T_1}, Bn.\overline{Vn} : \overline{T_2}\}$ and repType$(Ct'') = \{\overline{V_1} : \overline{T_1}\}$. Then by Lemma 4.11 we have repType$(Ct') = [\overline{Tn} \mapsto \overline{T}]\{\overline{V_1} : \overline{T_1}\}$, so the result follows.

4.2 Simple Lemmas

Lemma 4.13 If $T \leq T_1 \to T_2$, then T has the form $T_1' \to T_2'$, where $T_1 \leq T_1'$ and $T_2' \leq T_2$.
Proof By (strong) induction on the depth of the derivation of $T \leq T_1 \to T_2$. Case analysis on the last rule used in the derivation.

- Case SubTRef. Therefore $T = T_1 \to T_2$, so $T_1' = T_1$ and $T_2' = T_2$. By SubTRef we have $T_1 \leq T_1'$ and $T_2' \leq T_2$.
- Case SubTTrans. Therefore $T \leq T'$ and $T' \leq T_1 \to T_2$. By induction T' has the form $T_1'' \to T_2''$, where $T_1 \leq T_1''$ and $T_2'' \leq T_2$. Therefore, again by induction T has the form $T_1' \to T_2'$, where $T_1'' \leq T_1'$ and $T_2' \leq T_2''$. By SubTTrans we have $T_1 \leq T_1'$ and $T_2' \leq T_2$.
- Case SubTFun. Then T has the form $T_1' \to T_2'$, where $T_1 \leq T_1'$ and $T_2' \leq T_2$.

Lemma 4.14 If rep$(Ct(\overline{E})) = \{\overline{V_1} = \overline{E_1}\}$ and repType$(Ct) = \{\overline{V_2} : \overline{T_2}\}$ then $\overline{V_1} = \overline{V_2}$.
Proof By induction on the depth of the derivation of rep$(Ct(\overline{E})) = \{\overline{V_1} = \overline{E_1}\}$. By Rep we have $Ct = (\overline{T}\ Bn.Cn)$ and (<<`abstract`>> `class` $\overline{Tn}\ Cn(\overline{I_0} : \overline{T_0})$ <`extends` $Ct'(\overline{E_0})$ > `of` $\{\overline{Vn} : \overline{T_2} = \overline{E_2}\}) \in BT(Bn)$ and <rep$(Ct'(\overline{E_0})) = \{\overline{V_3} = \overline{E_3}\}$ > and $\overline{V_1}$ is equivalent to $< \overline{V_3}, > Bn.\overline{Vn}$. Since repType$(Ct) = \{\overline{V_2} : \overline{T_2}\}$, by RepType we have <repType$(Ct') = \{\overline{V_4} : \overline{T_4}\}$ >, so by induction $< \overline{V_3} = \overline{V_4} >$. Then by RepType $\overline{V_2}$ is equivalent to $< \overline{V_3}, > Bn.\overline{Vn}$.

4.3 Type Substitution

Lemma 4.15 If $T \leq T'$ and $|\overline{Tn}| = |\overline{T}|$, then $[\overline{Tn} \mapsto \overline{T}]T \leq [\overline{Tn} \mapsto \overline{T}]T'$.
Proof By (strong) induction on the depth of the derivation of $T \leq T'$. Case analysis of the last rule used in the derivation. The only interesting case is SubTExt.

- Case SubTExt. Then T has the form $\overline{T_0}\ Bn.Cn$ and T' has the form $[\overline{Tn_0} \mapsto \overline{T_0}]Ct$ and (<`abstract`> `class` $\overline{Tn_0}\ Cn(\overline{I_3} : \overline{T_3})$ `extends` $Ct(\overline{E}) \ldots) \in BT(Bn)$. Then by SubTExt we have $([\overline{Tn} \mapsto \overline{T}]\overline{T_0})\ Bn.Cn \leq [\overline{Tn_0} \mapsto [\overline{Tn} \mapsto \overline{T}]\overline{T_0}]Ct$. Note that $([\overline{Tn} \mapsto \overline{T}]\overline{T_0})\ Bn.Cn$ is equivalent to $[\overline{Tn} \mapsto \overline{T}](\overline{T_0}\ Bn.Cn)$. Further, by ClassOK we have that $\overline{Tn_0} \vdash Ct(\overline{E})$ OK, so by T-Super also $\overline{Tn_0} \vdash Ct$ OK. Therefore, by Lemma 4.1 all type variables in Ct are in $\overline{Tn_0}$. Therefore we have that $[\overline{Tn_0} \mapsto [\overline{Tn} \mapsto \overline{T}]\overline{T_0}]Ct$ is equivalent to $[\overline{Tn} \mapsto \overline{T}][\overline{Tn_0} \mapsto \overline{T_0}]Ct$. Therefore the result follows.

Lemma 4.16 If $\Gamma; \overline{Tn} \vdash E : T$ and $|\overline{Tn}| = |\overline{T}|$ and $\overline{Tn_0} \vdash \overline{T}$ OK, then $[\overline{Tn} \mapsto \overline{T}]\Gamma; \overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]E : [\overline{Tn} \mapsto \overline{T}]T$.
Proof By (strong) induction on the depth of the derivation of $\Gamma; \overline{Tn} \vdash E : T$. Case analysis of the last rule used in the derivation.

- Case T-Id. Then $E = I$ and $(I, T) \in \Gamma$. Therefore, $(I, [\overline{Tn} \mapsto \overline{T}]T) \in [\overline{Tn} \mapsto \overline{T}]\Gamma$. Also, $I = [\overline{Tn} \mapsto \overline{T}]I$. So by T-Id we have $[\overline{Tn} \mapsto \overline{T}]\Gamma; \overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]E : [\overline{Tn} \mapsto \overline{T}]T$.
- Case T-New. Then $E = Ct(\overline{E})$ and $T = Ct$ and $\overline{Tn} \vdash Ct(\overline{E})$ OK and $Ct = (\overline{T_1}\ Bn.Cn)$ and concrete$(Bn.Cn)$. By T-Super we have $\overline{Tn} \vdash Ct$ OK and (<`abstract`> `class` $\overline{Tn_1}\ Cn(\overline{I_0} : \overline{T_0}) \ldots) \in BT(Bn)$ and $\Gamma; \overline{Tn} \vdash \overline{E} : \overline{T_0'}$ and $\overline{T_0'} \leq [\overline{Tn_1} \mapsto \overline{T_1}]\overline{T_0}$. By Lemma 4.2 we have $\overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]Ct$ OK. Since $Ct = (\overline{T_1}\ Bn.Cn)$ we have $[\overline{Tn} \mapsto \overline{T}]Ct = [\overline{Tn} \mapsto \overline{T}](\overline{T_1}\ Bn.Cn) = ([\overline{Tn} \mapsto \overline{T}]\overline{T_1}\ Bn.Cn)$, which is of the form $(\overline{T_2}\ Bn.Cn)$. By induction we have $[\overline{Tn} \mapsto \overline{T}]\Gamma; \overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]\overline{E} : [\overline{Tn} \mapsto \overline{T}]\overline{T_0'}$. By Lemma 4.15 we have $[\overline{Tn} \mapsto \overline{T}]\overline{T_0'} \leq [\overline{Tn} \mapsto \overline{T}][\overline{Tn_1} \mapsto \overline{T_1}]\overline{T_0}$. By ClassOK we have $\overline{Tn_1} \vdash \overline{T_0}$ OK, so by Lemma 4.1 all type variables in each $\overline{T_0}$ are in $\overline{Tn_1}$. Therefore $[\overline{Tn} \mapsto \overline{T}][\overline{Tn_1} \mapsto \overline{T_1}]\overline{T_0}$ is equivalent to $[\overline{Tn_1} \mapsto [\overline{Tn} \mapsto \overline{T}]\overline{T_1}]\overline{T_0}$. Therefore by T-Super we have $[\overline{Tn} \mapsto \overline{T}]\Gamma; \overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]E$ OK, and the result follows by T-New.

- Case T-Rep. Then $E = Ct\,\{\overline{V} = \overline{E}\}$ and $T = Ct$ and $\overline{Tn} \vdash Ct$ OK and $Ct = (\overline{T_1}\ Bn.Cn)$ and concrete$(Bn.Cn)$ repType$(Ct) = \{\overline{V_0 : T_0}\}$ and $\Gamma; \overline{Tn} \vdash \overline{E} : \overline{T_0'}$ and $\overline{T_0'} \leq \overline{T_0}$. By Lemma 4.2 we have $\overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]Ct$ OK. Since $Ct = (\overline{T_1}\ Bn.Cn)$ we have $[\overline{Tn} \mapsto \overline{T}]Ct = [\overline{Tn} \mapsto \overline{T}](\overline{T_1}\ Bn.Cn) = ([\overline{Tn} \mapsto \overline{T}]\overline{T_1}\ Bn.Cn)$, which is of the form $(\overline{T_2}\ Bn.Cn)$. By Lemma 4.11 we have repType$([\overline{Tn} \mapsto \overline{T}]Ct)$ $= [\overline{Tn} \mapsto \overline{T}]\{\overline{V_0 : T_0}\}$. By induction we have $[\overline{Tn} \mapsto \overline{T}]\Gamma; \overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]\overline{E} : [\overline{Tn} \mapsto \overline{T}]\overline{T_0'}$. By Lemma 4.15 we have $[\overline{Tn} \mapsto \overline{T}]\overline{T_0'} \leq [\overline{Tn} \mapsto \overline{T}]\overline{T_0}$. Therefore by T-Rep the result follows.

- Case T-Fun. Then $E = \overline{T_1}\ Bn.Fn$ and $T = [\overline{Tn_1} \mapsto \overline{T_1}](\hat{Mt} \to T')$ and $\overline{Tn} \vdash \overline{T_1}$ OK and (`fun` $\overline{Tn_1}$ $Fn : Mt \to T') \in BT(Bn)$. By Lemma 4.2 we have $\overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]\overline{T_1}$ OK. Therefore by T-Fun we have $[\overline{Tn} \mapsto \overline{T}]\Gamma; \overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}](\overline{T_1}\ Bn.Fn) : [\overline{Tn} \mapsto \overline{T}][\overline{Tn_1} \mapsto \overline{T_1}](\hat{Mt} \to T')$. By FunOK we have $\overline{Tn} \vdash \hat{Mt}$ OK and $\overline{Tn} \vdash T'$ OK. Therefore by Lemma 4.1 we have that all type variables in $\hat{Mt}$ and T' are in $\overline{Tn}$. Therefore, $[\overline{Tn} \mapsto \overline{T}][\overline{Tn_1} \mapsto \overline{T_1}](\hat{Mt} \to T')$ is equivalent to $[\overline{Tn_1} \mapsto [\overline{Tn} \mapsto \overline{T}]\overline{T_1}](\hat{Mt} \to T')$, so the result follows.

- Case T-Tup. Then $E = (E_1, \ldots, E_k)$ and $T = T_1 * \cdots * T_k$ and for all $1 \leq i \leq k$ we have $\Gamma; \overline{Tn} \vdash E_i : T_i$. Therefore by induction, for all $1 \leq i \leq k$ we have $[\overline{Tn} \mapsto \overline{T}]\Gamma; \overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]E_i : [\overline{Tn} \mapsto \overline{T}]T_i$, and the result follows by T-Tup.

- Case T-App. Then $E = E_1\ E_2$ and $\Gamma; \overline{Tn} \vdash E_1 : T_2 \to T$ and $\Gamma; \overline{Tn} \vdash E_2 : T_2'$ and $T_2' \leq T_2$. By induction we have $[\overline{Tn} \mapsto \overline{T}]\Gamma; \overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]E_1 : [\overline{Tn} \mapsto \overline{T}](T_2 \to T)$ and $[\overline{Tn} \mapsto \overline{T}]\Gamma; \overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]E_2 : [\overline{Tn} \mapsto \overline{T}]T_2'$. By Lemma 4.15 we have $[\overline{Tn} \mapsto \overline{T}]T_2' \leq [\overline{Tn} \mapsto \overline{T}]T_2$, so the result follows by T-App.

Lemma 4.17 If matchType$(T, Pat) = (\Gamma, T')$ and $|\overline{Tn}| = |\overline{T}|$, then matchType$([\overline{Tn} \mapsto \overline{T}]T, Pat) = ([\overline{Tn} \mapsto \overline{T}]\Gamma, [\overline{Tn} \mapsto \overline{T}]T')$.
Proof By (strong) induction on the depth of the derivation of matchType$(T, Pat) = (\Gamma, T')$. Case analysis of the last rule used in the derivation.

- Case T-MatchWild. Then Pat has the form `_` and $\Gamma = \{\}$ and $T' = T$. Then $[\overline{Tn} \mapsto \overline{T}]T = [\overline{Tn} \mapsto \overline{T}]T'$ and $[\overline{Tn} \mapsto \overline{T}]\Gamma = \{\}$, so the result follows by T-MatchWild.

- Case T-MatchBind. Then Pat has the form I `as` Pat' and $\Gamma = \Gamma' \cup \{(I, T')\}$ and matchType$(T, Pat') = (\Gamma', T')$. By induction we have matchType$([\overline{Tn} \mapsto \overline{T}]T, Pat') = ([\overline{Tn} \mapsto \overline{T}]\Gamma', [\overline{Tn} \mapsto \overline{T}]T')$. Therefore by T-MatchBind we have matchType$([\overline{Tn} \mapsto \overline{T}]T, (I$ `as` $Pat') = [\overline{Tn} \mapsto \overline{T}]\Gamma' \cup \{(I, [\overline{Tn} \mapsto \overline{T}]T')\}, [\overline{Tn} \mapsto \overline{T}]T')$. Since $[\overline{Tn} \mapsto \overline{T}]\Gamma' \cup \{(I, [\overline{Tn} \mapsto \overline{T}]T')\}$ is equivalent to $[\overline{Tn} \mapsto \overline{T}](\Gamma' \cup \{(I, T')\})$, the result follows.

- Case T-MatchTup. Then $T = T_1 * \cdots * T_k$ and Pat has the form $(Pat_1, \ldots, Pat_k)$ and $\Gamma = \Gamma_1 \cup \ldots \cup \Gamma_k$ and $T' = T_1' * \cdots * T_k'$ and for all $1 \leq i \leq k$ we have matchType$(T_i, Pat_i) = (\Gamma_i, T_i')$. By induction, for all $1 \leq i \leq k$ we have matchType$([\overline{Tn} \mapsto \overline{T}]T_i, Pat_i) = ([\overline{Tn} \mapsto \overline{T}]\Gamma_i, [\overline{Tn} \mapsto \overline{T}]T_i')$. Therefore, the result follows by T-MatchTup.

- Case T-MatchClass. Then Pat has the form $C\ \{\overline{V} = \overline{Pat}\}$ and $T = (\overline{T_1}\ C')$ and $T' = (\overline{T_1}\ C)$ and $\Gamma = \bigcup \overline{\Gamma}$ and $C \leq C'$ and repType$(\overline{T_1}\ C) = \{\overline{V : T}\}$ and matchType$(\overline{T}, \overline{Pat}) = (\overline{\Gamma}, \overline{T'})$. By Lemma 4.11 we have repType$([\overline{Tn} \mapsto \overline{T}](\overline{T_1}\ C)) = [\overline{Tn} \mapsto \overline{T}]\{\overline{V : T}\}$. By induction we have matchType$([\overline{Tn} \mapsto \overline{T}]\overline{T}, \overline{Pat}) = ([\overline{Tn} \mapsto \overline{T}]\overline{\Gamma}, [\overline{Tn} \mapsto \overline{T}]\overline{T'})$. Therefore the result follows by T-MatchClass.

4.4 Subject Reduction

Lemma 4.18 If $\vdash v : T''$ and $T'' \leq T$ and match$(v, Pat) = e$ and matchType$(T, Pat) = (\Gamma, T')$, then (1) $T'' \leq T'$; and (2) dom(Γ) = dom(e) and for each $(I_0, T_0) \in \Gamma$, there exists $(I_0, v_0) \in e$ such that $\vdash v_0 : T_0'$,

where $T_0' \leq T_0$.
Proof By (strong) induction on the length of the derivation of $\mathrm{match}(v, Pat) = e$. Case analysis of the last rule used in the derivation:

- Case E-MatchWild. Then Pat has the form _ and $e = \{\}$. By T-MatchWild we have $\Gamma = \{\}$ and $T' = T$. Therefore, condition 1 follows from the assumption that $T'' \leq T$, and condition 2 holds vacuously.

- Case E-MatchBind. Then Pat has the form I `as` Pat' and $e = e' \cup \{(I, v)\}$ and $\mathrm{match}(v, Pat') = e'$. By T-MatchBind we have $\Gamma = \Gamma' \cup \{(I, T')\}$ and $\mathrm{matchType}(T, Pat') = (\Gamma', T')$. By induction we have that $T'' \leq T'$ and $\mathrm{dom}(\Gamma') = \mathrm{dom}(e')$ and for each $(I_0, T_0) \in \Gamma'$, there exists $(I_0, v_0) \in e'$ such that $\vdash v_0 : T_0'$, where $T_0' \leq T_0$. Therefore, we have $T'' \leq T'$ and $\mathrm{dom}(\Gamma' \cup \{(I, T')\}) = \mathrm{dom}(e' \cup \{(I, v)\})$ and for each $(I_0, T_0) \in \Gamma' \cup \{(I, T')\}$, there exists $(I_0, v_0) \in e' \cup \{(I, v)\}$ such that $\vdash v_0 : T_0'$, where $T_0' \leq T_0$.

- Case E-MatchTup. Then $v = (v_1, \ldots, v_k)$ and Pat has the form $(Pat_1, \ldots, Pat_k)$ and $e = e_1 \cup \cdots \cup e_k$ and for all $1 \leq i \leq k$ we have $\mathrm{match}(v_i, Pat_i) = e_i$. By T-MatchTup we have $T = T_1 * \cdots * T_k$ and $\Gamma = \Gamma_1 \cup \ldots \cup \Gamma_k$ and $T' = T_1' \cdots * T_k'$ and for all $1 \leq i \leq k$ we have $\mathrm{match}(T_i, Pat_i) = (\Gamma_i, T_i')$.

  Since we're given that $\vdash v : T''$, by T-Tup we have that $T'' = T_1'' * \cdots * T_k''$ and for all $1 \leq i \leq k$ we have $\vdash v_i : T_i''$. Since we're given that $T'' \leq T$, by Lemma 4.6 we have $T_i'' \leq T_i$ for all $1 \leq i \leq k$. Then by induction, for all $1 \leq i \leq k$ we have $T_i'' \leq T_i'$. Then by SubTTup we have $T_1'' * \cdots * T_k'' \leq T_1' * \ldots * T_k'$, proving condition 1. Also by induction, $\mathrm{dom}(\Gamma_i) = \mathrm{dom}(e_i)$ and for each $(I_0, T_0) \in \Gamma_i$, there exists $(I_0, v_0) \in e_i$ such that $\vdash v_0 : T_0'$, where $T_0' \leq T_0$, so condition 2 follows.

- Case E-MatchClass. Then $v = ((\overline{T}\ C)\ \{\overline{V_1} = \overline{v_1}, \overline{V_2} = \overline{v_2}\})$ and Pat has the form $(C'\ \{\overline{V_1} = \overline{Pat_1})$ and $C \leq C'$ and $e = \bigcup \overline{e_1}$ and $\mathrm{match}(\overline{v_1}, \overline{Pat_1}) = \overline{e_1}$. By T-MatchClass we have $T = (\overline{T'}\ C'')$ and $T' = (\overline{T'}\ C')$ and $\Gamma \bigcup \overline{\Gamma_1}$ and $C' \leq C''$ and $\mathrm{repType}(\overline{T'}\ C') = \{\overline{V_1} : \overline{T_1}\}$ and $\mathrm{matchType}(\overline{T_1}, \overline{Pat_1}) = (\overline{\Gamma_1}, \overline{T_1'})$.

  Since $\vdash v : T''$ and $v = ((\overline{T}\ C)\ \{\overline{V_1} = \overline{v_1}, \overline{V_2} = \overline{v_2}\})$, by T-Rep we have that $T'' = (\overline{T}\ C)$ and $\bullet \vdash (\overline{T}\ C)$ OK and and $\mathrm{repType}(\overline{T}\ C) = \{\overline{V_1} : \overline{T_1''}, \overline{V_2} : \overline{T_2''}\}$ and $\vdash \overline{v_1} : \overline{T_1'''}$ and $\overline{T_1'''} \leq \overline{T_1''}$. Since $T'' \leq T$, we have $(\overline{T}\ C) \leq (\overline{T_1}\ C'')$, so by Lemma 4.4 we have $\overline{T} = \overline{T_1}$. Since $C \leq C'$ and $\bullet \vdash (\overline{T}\ C)$ OK, by Lemma 4.7 we have $(\overline{T}\ C) \leq (\overline{T}\ C')$, and since $\overline{T} = \overline{T_1}$, condition 1 is shown. By Lemma 4.12 we have $\overline{T_1''} = \overline{T_1}$. Therefore $\vdash \overline{v_1} : \overline{T_1'''}$ and $\overline{T_1'''} \leq \overline{T_1}$ and $\mathrm{match}(\overline{v_1}, \overline{Pat_1}) = \overline{e_1}$ and $\mathrm{matchType}(\overline{T_1}, \overline{Pat_1}) = (\overline{\Gamma_1}, \overline{T_1'})$, so by induction we have that $\overline{T_1'''} \leq \overline{T_1'}$ and $\mathrm{dom}(\bigcup \overline{\Gamma_1}) = \mathrm{dom}(\bigcup \overline{e_1})$ and for each $(I_0, T_0) \in \bigcup \overline{\Gamma_1}$, there exists $(I_0, v_0) \in \bigcup \overline{e_1}$ such that $\vdash v_0 : T_0'$, where $T_0' \leq T_0$.

Lemma 4.19 (Substitution) If $\Gamma, \overline{Tn_0} \vdash E : T$ and $\Gamma = \{(\overline{I_0}, \overline{T_0})\}$ and $\Gamma_0; \overline{Tn_0} \vdash \overline{E_0} : \overline{T_0}'$ and $\overline{T_0'} \leq \overline{T_0}$, then $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]E : T'$ and $T' \leq T$.
Proof By (strong) induction on the depth of the derivation of $\Gamma, \overline{Tn_0} \vdash E : T$. Case analysis of the last rule used in the derivation.

- Case T-Id. Then $E = I$ and $(I, T) \in \Gamma$, so $I = I_j$ and $T = T_j$, for some $1 \leq j \leq k$, where $\overline{I_0} = I_1, \ldots, I_k$ and $\overline{T_0} = T_1, \ldots, T_k$ and $\overline{E_0} = E_1, \ldots, E_k$. Therefore $[\overline{I_0} \mapsto \overline{E_0}]E = E_j$. Since we're given that $\Gamma_0; \overline{Tn_0} \vdash E_j : T_j'$ and $T_j' \leq T_j$, the result is shown.

- Case T-New. Then $E = Ct(\overline{E})$ and $T = Ct$ and $\overline{Tn_0} \vdash Ct(\overline{E})$ OK and $Ct = (\overline{T_1}\ Bn.Cn)$ and $\mathrm{concrete}(Bn.Cn)$. Then by T-Super we have $\overline{Tn_0} \vdash Ct$ OK and (`<abstract> class` $\overline{Tn_1}\ Cn(\overline{I} : \overline{T})$ $\ldots) \in BT(Bn)$ and $\Gamma; \overline{Tn_0} \vdash \overline{E} : \overline{T'}$ and $\overline{T'} \leq [\overline{Tn_1} \mapsto \overline{T_1}]\overline{T}$. Since $[\overline{I_0} \mapsto \overline{E_0}]Ct = Ct$ and $[\overline{I_0} \mapsto \overline{E_0}]Bn.Cn = Bn.Cn$, we have $\overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]Ct$ OK and $\mathrm{concrete}([\overline{I_0} \mapsto \overline{E_0}]Bn.Cn)$. By induction we have $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]\overline{E} : \overline{T''}$ and $\overline{T''} \leq \overline{T'}$. Then by SubTTrans we have $\overline{T''} \leq [\overline{Tn_1} \mapsto \overline{T_1}]\overline{T'}$.

Therefore by T-Super we have $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]E$ OK, so by T-New we have $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]E : T$. By SubTRef we have $T \leq T$, so the result is shown.

- Case T-Rep. Then $E = Ct\,\{\overline{V} = \overline{E}\}$ and $T = Ct$ and $\overline{Tn_0} \vdash Ct$ OK and $Ct = (\overline{T_1}\ Bn.Cn)$ and concrete$(Bn.Cn)$ and repType$(Ct) = \{\overline{V} : \overline{T}\}$ and $\Gamma; \overline{Tn_0} \vdash \overline{E} : \overline{T'}$ and $\overline{T'} \leq \overline{T}$. Since $[\overline{I_0} \mapsto \overline{E_0}]\,Ct = Ct$ and $[\overline{I_0} \mapsto \overline{E_0}]Bn.Cn = Bn.Cn$, we have $\overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]\,Ct$ OK and concrete$([\overline{I_0} \mapsto \overline{E_0}]Bn.Cn)$ and and repType$([\overline{I_0} \mapsto \overline{E_0}]\,Ct) = \{\overline{V} : \overline{T}\}$. By induction we have $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]\overline{E} : \overline{T''}$ and $\overline{T''} \leq \overline{T'}$. Then by SubTTrans we have $\overline{T''} \leq \overline{T}$, so by T-Rep we have $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]E : T$. By SubTRef we have $T \leq T$, so the result is shown.

- Case T-Fun. Then since Γ is not used at all in T-Fun and $\Gamma; \overline{Tn_0} \vdash E : T$, also $\Gamma_0; \overline{Tn_0} \vdash E : T$. Further, we have $E = Fv$, so $[\overline{I_0} \mapsto \overline{E_0}]E = E$. Therefore $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]E : T$, and by SubTRef $T \leq T$, so the result is shown.

- Case T-Tup. Then $E = (E_1, \ldots, E_k)$ and $T = T_1 * \cdots * T_k$ and for all $1 \leq j \leq k$ we have $\Gamma; \overline{Tn_0} \vdash E_j : T_j$. Then by induction, for all $1 \leq j \leq k$ we have $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]E_j : T'_j$ and $T'_j \leq T_j$. Then by T-Tup we have $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}](E_1, \ldots, E_k) : T'_1 * \cdots * T'_k$. Finally, by SubTTup we have $T'_1 * \cdots * T'_k \leq T_1 * \cdots * T_k$.

- Case T-App. Then $E = E_1\ E_2$ and $\Gamma; \overline{Tn_0} \vdash E_1 : T_2 \rightarrow T$ and $\Gamma; \overline{Tn_0} \vdash E_2 : T'_2$ and $T'_2 \leq T_2$. By induction we have $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]E_1 : T_0$ and $T_0 \leq T_2 \rightarrow T$. Also by induction we have $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}]E_2 : T''_2$ and $T''_2 \leq T'_2$. Then by SubTTrans we have $T''_2 \leq T_2$. By Lemma 4.13 T_0 has the form $T_{arg} \rightarrow T_{res}$, where $T_2 \leq T_{arg}$ and $T_{res} \leq T$. Therefore by SubTTrans we have $T''_2 \leq T_{arg}$. Therefore by T-Fun we have $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_0} \mapsto \overline{E_0}](E'_1\ E'_2) : T_{res}$. We saw above that $T_{res} \leq T$, so the result is shown.

Lemma 4.20 If $\Gamma_0; \overline{Tn_0} \vdash Ct(\overline{E})$ OK and rep$(Ct(\overline{E})) = \{\overline{V_0} = \overline{E_0}\}$ and repType$(Ct) = \{\overline{V_0} : \overline{T_0}\}$, then $\Gamma_0; \overline{Tn_0} \vdash \overline{E_0} : \overline{T'_0}$ and $\overline{T'_0} \leq \overline{T_0}$.
Proof Since $\Gamma_0; \overline{Tn_0} \vdash Ct(\overline{E})$ OK, by T-Super we have $\overline{Tn_0} \vdash Ct$ OK and $Ct = (\overline{T}\ Bn.Cn)$ and $(\texttt{<abstract>}\ \texttt{class}\ \overline{Tn}\ Cn(\overline{I_1} : \overline{T_1})\ \ldots) \in BT(Bn)$ and $\Gamma_0; \overline{Tn_0} \vdash \overline{E} : \overline{T'_1}$ and $\overline{T'_1} \leq [\overline{Tn} \mapsto \overline{T}]\overline{T_1}$. Since $\overline{Tn_0} \vdash Ct$ OK, by ClassTypeOK we have $\overline{Tn_0} \vdash \overline{T}$ OK and $|\overline{T}| = |\overline{Tn}|$. We prove the lemma by induction on the depth of the derivation of rep$(Ct(\overline{E})) = \{\overline{V_0} = \overline{E_0}\}$.

By Rep we have $(\texttt{<<abstract>>}\ \texttt{class}\ \overline{Tn}\ Cn(\overline{I_1} : \overline{T_1})\ \texttt{<extends}\ Ct'(\overline{E_1})\ \texttt{>}\ \texttt{of}\ \{\overline{Vn} : \overline{T_2} = \overline{E_2}\}) \in BT(Bn)$ and $<$rep$(Ct'(\overline{E_1})) = \{\overline{V_3} = \overline{E_3}\} >$ and $\{\overline{V_0} = \overline{E_0}\}$ is equivalent to $[\overline{I_1} \mapsto \overline{E}][\overline{Tn} \mapsto \overline{T}]\{< \overline{V_3} = \overline{E_3}, > Bn.\overline{Vn} = \overline{E_2}\}$. Since repType$(Ct) = \{\overline{V_0} : \overline{T_0}\}$, by RepType and Lemma 4.14 we have that $<$repType$(Ct') = \{\overline{V_3} : \overline{T_3}\} >$ and $\{\overline{V_0} : \overline{T_0}\}$ is equivalent to $[\overline{Tn} \mapsto \overline{T}]\{< \overline{V_3} : \overline{T_3}, > Bn.\overline{Vn} : \overline{T_2}\}$.

Let $\Gamma = \{(\overline{I_1}, \overline{T_1})\}$. By ClassOK we have $< \Gamma; \overline{Tn} \vdash Ct'(\overline{E_1})$ OK $>$. Therefore by induction we have $< \Gamma; \overline{Tn} \vdash \overline{E_3} : \overline{T'_3} >$ and $< \overline{T'_3} \leq \overline{T_3} >$. Also by ClassOK we have $\Gamma; \overline{Tn} \vdash \overline{E_2} : \overline{T'_2}$ and $\overline{T'_2} \leq \overline{T_2}$. Then by Lemmas 4.16 and 4.15 we have $< [\overline{Tn} \mapsto \overline{T}]\Gamma; \overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]\overline{E_3} : [\overline{Tn} \mapsto \overline{T}]\overline{T'_3} >$ and $< [\overline{Tn} \mapsto \overline{T}]\overline{T'_3} \leq [\overline{Tn} \mapsto \overline{T}]\overline{T_3} >$ and $[\overline{Tn} \mapsto \overline{T}]\Gamma; \overline{Tn_0} \vdash [\overline{Tn} \mapsto \overline{T}]\overline{E_2} : [\overline{Tn} \mapsto \overline{T}]\overline{T'_2}$ and $[\overline{Tn} \mapsto \overline{T}]\overline{T'_2} \leq [\overline{Tn} \mapsto \overline{T}]\overline{T_2}$. Then by Lemma 4.19 we have $< \Gamma_0; \overline{Tn_0} \vdash [\overline{I_1} \mapsto \overline{E}][\overline{Tn} \mapsto \overline{T}]\overline{E_3} : \overline{T''_3} >$ and $< \overline{T''_3} \leq [\overline{Tn} \mapsto \overline{T}]\overline{T'_3} >$ and $\Gamma_0; \overline{Tn_0} \vdash [\overline{I_1} \mapsto \overline{E}][\overline{Tn} \mapsto \overline{T}]\overline{E_2} : \overline{T''_2}$ and $\overline{T''_2} \leq [\overline{Tn} \mapsto \overline{T}]\overline{T'_2}$. By SubTrans we have $< \overline{T''_3} \leq [\overline{Tn} \mapsto \overline{T}]\overline{T_3} >$ and $\overline{T''_2} \leq [\overline{Tn} \mapsto \overline{T}]\overline{T_2}$. Therefore we have shown $\Gamma_0; \overline{Tn_0} \vdash \overline{E_0} : \overline{T'_0}$ and $\overline{T'_0} \leq \overline{T_0}$.

Theorem 4.1 (Subject Reduction) If $\vdash E : T$ and $E \longrightarrow E'$ then $\vdash E' : T'$, for some T' such that $T' \leq T$.
Proof By (strong) induction on the depth of the derivation of $E \longrightarrow E'$. Case analysis of the last rule used in the derivation.

- Case E-New. Then E has the form $Ct(\overline{E})$ and E' has the form $Ct\ \{\overline{V_0} = \overline{E_0}\}$ and $Ct = (\overline{T}\ C)$ and concrete(C) and rep$(Ct(\overline{E})) = \{\overline{V_0} = \overline{E_0}\}$. Since $\vdash E : T$, by T-New we have $T = Ct$ and

$\bullet \vdash Ct(\overline{E})$ OK. Then by T-SUPER we have $\bullet \vdash Ct$ OK. Therefore by Lemmas 4.8 and 4.14 we have repType$(Ct) = \{\overline{V_0 : T_0}\}$. So we have $\vdash Ct(\overline{E})$ OK and rep$(Ct(\overline{E})) = \{\overline{V_0 = E_0}\}$ and repType$(Ct) = \{\overline{V_0 : T_0}\}$, so by Lemma 4.20 we have $\vdash \overline{E_0 : T_0'}$ and $\overline{T_0' \leq T_0}$. Then by T-REP we have $\vdash Ct\ \{\overline{V_0 = E_0}\} : Ct$, and by SUBTREF we have $Ct \leq Ct$.

- Case E-REP. Then E has the form $Ct\ \{\overline{V_0 = E_0}, V_0 = E_0, \overline{V_1 = E_1}\}$ and E' has the form $Ct\ \{\overline{V_0 = E_0}, V_0 = E_0', \overline{V_1 = E_1}\}$ and $E_0 \longrightarrow E_0'$. Since $\vdash E : T$, by T-REP we have $T = Ct$ and $\bullet \vdash Ct$ OK and repType$(Ct) = \{\overline{V_0 : T_0}, V_0 : T_0, \overline{V_1 : T_1}\}$ and $\vdash \overline{E_0 : T_0'}$ and $\overline{T_0' \leq T_0}$ and $\vdash E_0 : T_0'$ and $T_0' \leq T_0$ and $\vdash \overline{E_1 : T_1'}$ and $\overline{T_1' \leq T_1}$. By induction we have $\vdash E_0' : T_0''$, for some T_0'' such that $T_0'' \leq T_0'$. Therefore by SUBTTRANS we have that $T_0'' \leq T_0$. Then by T-REP we have $\vdash Ct\ \{\overline{V_0 = E_0}, V_0 = E_0', \overline{V_1 = E_1}\} : Ct$, and by SUBTREF we have $Ct \leq Ct$.

- Case E-TUP. Then E has the form $(E_1, \ldots, E_k)$ and E' has the form $(E_1, \ldots, E_{i-1}, E_i', E_{i+1}, \ldots, E_k)$ and $E_i \longrightarrow E_i'$, where $1 \leq i \leq k$. Since $\vdash E : T$, by T-TUP we have that T has the form $T_1 * \cdots * T_k$ and $\vdash E_j : T_j$ for all $1 \leq j \leq k$. Therefore by induction we have $\vdash E_i' : T_i'$ for some T_i' such that $T_i' \leq T_i$. Then by T-TUP we have $\vdash (E_1, \ldots, E_{i-1}, E_i', E_{i+1}, \ldots, E_k) : T_1 * \cdots * T_{i-1} * T_i' * T_{i+1} * \cdots * T_k$. Finally, by SUBTREF we have that $T_j \leq T_j$ for all $1 \leq j \leq k$, so by SUBTTUP we have $T_1 * \cdots * T_{i-1} * T_i' * T_{i+1} * \cdots * T_k \leq T_1 * \cdots * T_k$.

- Case E-APP1. Then E has the form $E_1\ E_2$ and E' has the form $E_1'\ E_2$ and $E_1 \longrightarrow E_1'$. Since $\vdash E : T$, by (T-App) we have $\vdash E_1 : T_2 \rightarrow T$ and $\vdash E_2 : T_2'$ and $T_2' \leq T_2$. Therefore by induction we have $\vdash E_1' : T'$, for some T' such that $T' \leq T_2 \rightarrow T$. By Lemma 4.13 T' has the form $T_2'' \rightarrow T''$, where $T_2 \leq T_2''$ and $T'' \leq T$. Therefore by SUBTTRANS we have $T_2' \leq T_2''$, so by T-APP we have $\vdash E_1'\ E_2 : T''$, where $T'' \leq T$.

- Case E-APP2. Then E has the form $E_1\ E_2$ and E' has the form $E_1\ E_2'$ and $E_2 \longrightarrow E_2'$. Since $\vdash E : T$, by T-APP we have $\vdash E_1 : T_2 \rightarrow T$ and $\vdash E_2 : T_2'$ and $T_2' \leq T_2$. Therefore by induction we have $\vdash E_2' : T_2''$, for some T_2'' such that $T_2'' \leq T_2'$. By SUBTTRANS we have $T_2'' \leq T_2$, so by T-APP we have $\vdash E_1\ E_2' : T$, and by SUBTREF we have $T \leq T$.

- Case E-APPRED. Then $E = (\overline{T}\ F)\ v$ and $E' = [\overline{I_0} \mapsto \overline{v_0}]E_0$ and most-specific-case-for$((\overline{T}\ F), v) = (\{(\overline{I_0}, \overline{v_0})\}, E_0)$. Since $\vdash E : T$, by T-APP we have $\vdash (\overline{T}\ F) : T_2 \rightarrow T$ and $\vdash v : T_2'$ and $T_2' \leq T_2$. Then by T-FUN we have and $F = Bn.Fn$ and $T_2 \rightarrow T = [\overline{Tn} \mapsto \overline{T}](\hat{M}t \rightarrow T_0)$ and $(\texttt{fun}\ \overline{Tn}\ Fn : Mt \rightarrow T_0) \in BT(Bn)$ and $\bullet \vdash \overline{T}$ OK. Therefore we have $T_2 = [\overline{Tn} \mapsto \overline{T}]\hat{M}t$ and $T = [\overline{Tn} \mapsto \overline{T}]T_0$. By LOOKUP we have $E_0 = [\overline{Tn_0} \mapsto \overline{T}]E_0'$ and $(\texttt{extend fun}_{Mn}\ \overline{Tn_0}\ F\ Pat = E_0') \in BT(Bn')$ and match$(v, Pat) = \{(\overline{I_0}, \overline{v_0})\}$. Then by CASEOK we have $\overline{Tn_0} \vdash$ matchType$([\overline{Tn} \mapsto \overline{Tn_0}]\hat{M}t, Pat) = (\Gamma, T'')$ and $\Gamma; \overline{Tn_0} \vdash E_0' : T_0'$ and $T_0' \leq [\overline{Tn} \mapsto \overline{Tn_0}]T_0$.

  By Lemma 4.16 we have $[\overline{Tn_0} \mapsto \overline{T}]\Gamma; \bullet \vdash [\overline{Tn_0} \mapsto \overline{T}]E_0' : [\overline{Tn_0} \mapsto \overline{T}]T_0'$. By Lemma 4.15 we have $[\overline{Tn_0} \mapsto \overline{T}]T_0' \leq [\overline{Tn_0} \mapsto \overline{T}][\overline{Tn} \mapsto \overline{Tn_0}]T_0$. By FUNOK we have $\overline{Tn} \vdash T_0$ OK, so by Lemma 4.1 all type variables in T_0 are in $\overline{Tn}$. Therefore $[\overline{Tn_0} \mapsto \overline{T}][\overline{Tn} \mapsto \overline{Tn_0}]T_0$ is equivalent to $[\overline{Tn} \mapsto \overline{T}]T_0 = T$, so we have $[\overline{Tn_0} \mapsto \overline{T}]T_0' \leq T$.

  By Lemma 4.17 we have $\bullet \vdash$ matchType$([\overline{Tn_0} \mapsto \overline{T}][\overline{Tn} \mapsto \overline{Tn_0}]\hat{M}t, Pat) = ([\overline{Tn_0} \mapsto \overline{T}]\Gamma, [\overline{Tn_0} \mapsto \overline{T}]T'')$. By FUNOK we have $\overline{Tn} \vdash \hat{M}t$ OK, so by Lemma 4.1 all type variables in $\hat{M}t$ are in $\overline{Tn}$. Therefore $[\overline{Tn_0} \mapsto \overline{T}][\overline{Tn} \mapsto \overline{Tn_0}]\hat{M}t$ is equivalent to $[\overline{Tn} \mapsto \overline{T}]\hat{M}t = T_2$, so we have $\bullet \vdash$ matchType$(T_2, Pat) = ([\overline{Tn_0} \mapsto \overline{T}]\Gamma, [\overline{Tn_0} \mapsto \overline{T}]T'')$.

  By Lemma 4.18 we have $T_2' \leq [\overline{Tn_0} \mapsto \overline{T}]T''$ and dom$([\overline{Tn_0} \mapsto \overline{T}]\Gamma) =$ dom$(\{(\overline{I_0}, \overline{v_0})\})$ and for each $(I_x, T_x) \in [\overline{Tn_0} \mapsto \overline{T}]\Gamma$, there exists $(I_x, v_x) \in \{(\overline{I_0}, \overline{v_0})\}$ such that $\vdash v_x : T_x'$, where $T_x' \leq T_x$. Then by Lemma 4.19 we have $\vdash [\overline{I_0} \mapsto \overline{v_0}][\overline{Tn_0} \mapsto \overline{T}]E_0' : T_{sub}$ and $T_{sub} \leq [\overline{Tn_0} \mapsto \overline{T}]T_0'$. We saw above that

$[\overline{Tn_0} \mapsto \overline{T}]T_0' \leq T$, so by SubTTrans we have $T_{sub} \leq T$. Therefore we have shown $\vdash E' : T_{sub}$ and $T_{sub} \leq T$.

5 Progress

5.1 Preliminaries and Simple Lemmas

We say that $S \subseteq S'$, where S is either a set or a sequence and similarly for S', if for every element d such that $d \in S$, also $d \in S'$. The notation $Pat < Pat'$ is shorthand for $Pat \leq Pat'$ and $Pat' \not\leq pat$.

Lemma 5.1 If $T \leq (\overline{T}\ C)$, then T has the form $(\overline{T_1}\ C')$.
Proof By (strong) induction on the depth of the derivation of $T \leq (\overline{T}\ C)$. Case analysis of the last rule used in the derivation.

- Case SubTRef. Then $T = (\overline{T}\ C)$.
- Case SubTTrans. Then $T \leq T'$ and $T' \leq (\overline{T}\ C)$. By induction T' has the form $(\overline{T_2}\ C'')$. Then by induction again, T has the form $(\overline{T_1}\ C')$.
- Case SubTExt. Then T has the form $(\overline{T_1} Bn.Cn)$, which is also of the form $(\overline{T_1}\ C')$.

Lemma 5.2 If $T_1 \rightarrow T_2 \leq T$, then T has the form $T_1' \rightarrow T_2'$.
Proof By (strong) induction on the depth of the derivation of $T_1 \rightarrow T_2 \leq T$. Case analysis of the last rule used in the derivation.

- Case SubTRef. Then $T = T_1 \rightarrow T_2$.
- Case SubTTrans. Then $T_1 \rightarrow T_2 \leq T'$ and $T' \leq T$. By induction T' has the form $T_1'' \rightarrow T_2''$. Then by induction again, T has the form $T_1' \rightarrow T_2'$.
- Case SubTFun. Then T has the form $T_1' \rightarrow T_2'$.

Lemma 5.3 If $T_1 * \cdots * T_k \leq T$, then T has the form $T_1' * \cdots * T_k'$, where for all $1 \leq i \leq k$ we have $T_i \leq T_i'$.
Proof By (strong) induction on the depth of the derivation of $T_1 * \cdots * T_k \leq T$. Case analysis of the last rule used in the derivation.

- Case SubTRef. Then $T = T_1 * \cdots * T_k$. By SubTRef, for all $1 \leq i \leq k$ we have $T_i \leq T_i$.
- Case SubTTrans. Then $T_1 * \cdots * T_k \leq T'$ and $T' \leq T$. By induction T' has the form $T_1'' * \cdots * T_k''$, where for all $1 \leq i \leq k$ we have $T_i \leq T_i''$. Then by induction again, T has the form $T_1' * \cdots * T_k'$, where for all $1 \leq i \leq k$ we have $T_i'' \leq T_i'$. By SubTTrans, for all $1 \leq i \leq k$ we have $T_i \leq T_i'$.
- Case SubTTup. Then T has the form $T_1' * \cdots * T_k'$, where for all $1 \leq i \leq k$ we have $T_i \leq T_i'$.

Lemma 5.4 If $C_1 \leq C_2$ and $C_1 \leq C_3$, then either $C_2 \leq C_3$ or $C_3 \leq C_2$.
Proof By induction on the depth of the derivation of $C_1 \leq C_2$. Case analysis of the last rule used in the derivation.

- Case SubRef. Then $C_1 = C_2$. Since $C_1 \leq C_3$, also $C_2 \leq C_3$.
- Case SubTrans. Then $C_1 \leq C_4$ and $C_4 \leq C_2$. So we have $C_1 \leq C_4$ and $C_1 \leq C_3$, and by induction either $C_4 \leq C_3$ or $C_3 \leq C_4$.

- Case $C_4 \leq C_3$. Then we have $C_4 \leq C_2$ and $C_4 \leq C_3$, so by induction either $C_2 \leq C_3$ or $C_3 \leq C_2$.
- Case $C_3 \leq C_4$. Then we have $C_3 \leq C_4$ and $C_4 \leq C_2$, so by SubTrans $C_3 \leq C_2$.

- Case SubExt. Then $C_1 = Bn_1.Cn_1$ and (`<abstract> class` $\overline{Tn}$ $Cn_1(\overline{I_0} : \overline{T_0})$ `extends` $\overline{T}$ C_2 ...) $\in BT(Bn_1)$. Case analysis of the last rule used in the derivation of $C_1 \leq C_3$.
  - Case SubRef. Then $C_1 = C_3$. Since $C_1 \leq C_2$, also $C_3 \leq C_2$.
  - Case SubTrans. Then $C_1 \leq C_4$ and $C_4 \leq C_3$. Assume WLOG that the derivation of $C_1 \leq C_4$ ends with a use of SubExt. Then (`<abstract> class` $\overline{Tn}$ $Cn_1(\overline{I_0} : \overline{T_0})$ `extends` $\overline{T}$ C_4 ...) $\in BT(Bn_1)$, so $C_2 = C_4$. Since $C_4 \leq C_3$, also $C_2 \leq C_3$.
  - Case SubExt. Then (`<abstract> class` $\overline{Tn}$ $Cn_1(\overline{I_0} : \overline{T_0})$ `extends` $\overline{T}$ C_3 ...) $\in BT(Bn_1)$, so $C_2 = C_3$. Then by SubRef $C_2 \leq C_3$.

Lemma 5.5 If $C_1 \leq C_2$, then there is a path in the declared inheritance graph from C_1 to C_2.
Proof By induction on the depth of the derivation of $C_1 \leq C_2$. Case analysis of the last rule used in the derivation.

- Case SubRef. Then $C_1 = C_2$, so there is a trivial path in the inheritance graph from C_1 to C_2.
- Case SubTrans. Then $C_1 \leq C_3$ and $C_3 \leq C_2$. By induction, there is a path in the inheritance graph from C_1 to C_3 and from C_3 to C_2, so the concatenation of these paths is a path from C_1 to C_2.
- Case SubExt. Then $C_1 = Bn_1.Cn_1$ and `<abstract> class` $\overline{Tn_1}$ $Cn_1(\overline{I_0} : \overline{T_0})$ `extends` $\overline{T}$ C_2 ...) $\in BT(Bn_1)$. Therefore there is an edge from C_1 to C_2 in the declared inheritance graph, so there is also a path from C_1 to C_2.

Lemma 5.6 If $C_1 \leq C_2$ and $C_2 \leq C_1$, then $C_1 = C_2$.
Proof By Lemma 5.5, there is a path in the declared inheritance graph from C_1 to C_2 and a path from C_2 to C_1. By assumption, the declared inheritance graph is acyclic, so it must be the case that $C_1 = C_2$.

Lemma 5.7 If $\text{match}(v, Pat) = e$ and $Pat \leq Pat'$, then there exists e' such that $\text{match}(v, Pat') = e'$.
Proof By induction on the depth of the derivation of $Pat \leq Pat'$. Case analysis of the last rule used in the derivation:

- Case SpecWild. Then Pat' has the form _, so by E-MatchWild we have $\text{match}(v, \_) = \{\}$.
- Case SpecBind1.: Then Pat has the form (I `as` Pat_1) and we have $Pat_1 \leq Pat'$. Since we're given that $\text{match}(v,\ I$ `as` $Pat_1) = e$, by E-MatchBind we also have that $\text{match}(v,\ Pat_1) = e - \{(I, v)\}$. Therefore by induction there exists e' such that $\text{match}(v, Pat') = e'$.
- Case SpecBind2.: Then Pat' has the form (I `as` Pat_2) and we have $Pat \leq Pat_2$. Therefore by induction we have that there exists e'' such that $\text{match}(v, Pat_2) = e''$. Then by E-MatchBind we have $\text{match}(v,$ I `as` $Pat_2) = e'' \cup \{I, v\}$.
- Case SpecTup. Then Pat has the form $(\overline{Pat})$ and Pat' has the form $(\overline{Pat'})$ and $\overline{Pat} \leq \overline{Pat'}$. Since we're given that $\text{match}(v,(\overline{Pat})) = e$, by E-MatchTup we have that $v = (\overline{v})$ and $\text{match}(\overline{v}, \overline{Pat}) = \overline{e}$. Therefore by induction we have $\text{match}(\overline{v}, \overline{Pat'}) = \overline{e'}$. Then by E-MatchTup we have $\text{match}((\overline{v}), (\overline{Pat}))$ $= \bigcup \overline{e'}$.

- Case SpecClass. Then Pat has the form $(C_1\ \{\overline{V} = \overline{Pat_1}, \overline{V_3} = \overline{Pat_3}\})$ and Pat' has the form $(C_2\ \{\overline{V} = \overline{Pat_2}\})$ and $C_1 \leq C_2$ and $\overline{Pat_1} \leq \overline{pat_2}$. Since we're given that $\text{match}(v, C_1\ \{\overline{V} = \overline{Pat_1}, \overline{V_3} = \overline{Pat_3}\}) = e$, by E-MatchClass we have that $v = ((\overline{T}\ C_0)\ \{\overline{V} = \overline{v}, \overline{V_3} = \overline{v_3}, \overline{V_4} = \overline{v_4}\})$ and $C_0 \leq C_1$ and $\text{match}(\overline{v}, \overline{Pat_1}) = \overline{e_1}$. Since $C_0 \leq C_1$ and $C_1 \leq C_2$, by SubTrans we have $C_0 \leq C_2$. By induction we have $\text{match}(\overline{v}, \overline{Pat_2}) = \overline{e_2}$. Therefore by E-MatchClass we have $\text{match}((\overline{T}\ C_0)\ \{\overline{V} = \overline{v}, \overline{V_3} = \overline{v_3}, \overline{V_4} = \overline{v_4}\}), C_2\ \{\overline{V} = \overline{Pat_2}\}) = \bigcup \overline{e_2}$.

Lemma 5.8 If $\overline{Bn} \vdash C$ transExtended and $C \leq Bn'.Cn'$, then $Bn' \in \overline{Bn}$.
Proof By induction on the depth of the derivation of $C \leq Bn'.Cn'$. Case analysis of the last rule in the derivation.

- Case SubRef. Then $C = Bn'.Cn'$. Since we're given that $\overline{Bn} \vdash C$ transExtended, by ClassTransExt we have $Bn' \in \overline{Bn}$.

- Case SubTrans. Then $C \leq Bn''.Cn''$ and $Bn''.Cn'' \leq Bn'.Cn'$. Assume WLOG that the derivation of $C \leq Bn''.Cn''$ ends with a use of SubExt. Let $C = Bn.Cn$. Therefore by SubExt we have (<`abstract`> `class` $\overline{Tn}\ Cn(\overline{I_0} : \overline{T_0})$ `extends` $\overline{T_2}\ Bn''.Cn''\ \ldots) \in BT(Bn)$. Since we're given that $\overline{Bn} \vdash C$ transExtended, by ClassTransExt we have $\overline{Bn} \vdash Bn''.Cn''$ transExtended. In addition, we showed above that $Bn''.Cn'' \leq Bn'.Cn'$, so by induction we have $Bn' \in \overline{Bn}$.

- Case SubExt. Then (<`abstract`> `class` $\overline{Tn}\ Cn(\overline{I_0} : \overline{T_0})$ `extends` $\overline{T_1}\ Bn'.Cn'\ \ldots) \in BT(Bn)$. Since we're given that $\overline{Bn} \vdash C$ transExtended, by ClassTransExt we have $\overline{Bn} \vdash Bn'.Cn'$ transExtended. Therefore by ClassTransExt we have $Bn' \in \overline{Bn}$.

Lemma 5.9 If $\overline{Tn} \vdash Ct$ OK and $Ct = (\overline{T}\ Bn.Cn)$ and (<`abstract`> `class` $\overline{Tn_0}\ Cn(\overline{I_0} : \overline{T_0})\ \ldots) \in BT(Bn)$ and $|\overline{E_0}| = |\overline{I_0}|$ then $\text{rep}(Ct(\overline{E_0}))$ is well-defined and has the form $\{\overline{V} = \overline{E}\}$.
Proof We prove this lemma by induction on the length of the longest path in the superclass graph from $Bn.Cn$ (in other words, the number of non-trivial superclasses of $Bn.Cn$). By ClassTypeOK we have $\overline{Tn} \vdash \overline{T}$ OK and (<<`abstract`>> `class` $\overline{Tn_0}\ Cn(\overline{I_0} : \overline{T_0})$ <`extends` $Ct'(\overline{E'})$ > `of` $\overline{Vn} : \overline{T_2} = \overline{E_2}\}) \in BT(Bn)$ and $|\overline{Tn_0}| = |\overline{T}|$. There are two cases to consider.

- The length of the longest path in the superclass graph from $Bn.Cn$ is 0. Then $Bn.Cn$ has no non-trivial superclasses, so the `extends` clause in the declaration of $Bn.Cn$ is absent. Then by Rep we have that $\text{rep}(Ct(\overline{E_0}))$ is well-defined and has the form $\{\overline{V} = \overline{E}\}$.

- The length of the longest path in the superclass graph from $Bn.Cn$ is $i > 0$. Then $Bn.Cn$ has at least one non-trivial superclass, so the `extends` clause in the declaration of $Bn.Cn$ is present. Then by ClassOK we have $\overline{Tn_0} \vdash Ct'(\overline{E'})$ OK, so by T-Super we have $\overline{Tn_0} \vdash Ct'$ OK and $Ct' = (\overline{Tn_1}\ Bn'.Cn')$ and (<`abstract`> `class` $\overline{Tn_0}\ Cn'(\overline{I'_0} : \overline{T'_0})\ \ldots) \in BT(Bn')$ and $|\overline{I'_0}| = |\overline{E'}|$. Since Ct' must have the form $(\overline{T_1}\ Bn'.Cn')$, where the length of the longest path in the superclass graph from $Bn'.Cn'$ is $i-1$, by induction we have that $\text{rep}(Ct'(\overline{E'}))$ is well-defined and has the form $\{\overline{V} = \overline{E}\}$. Then by Rep we have that $\text{rep}(Ct(\overline{E_0}))$ is well-defined and also has the appropriate form.

5.2 Completeness

These lemmas prove that all functions are complete.

Lemma 5.10 If $\vdash v : T'$ and $T' \leq T$ and $T = [\overline{Tn} \mapsto \overline{T}]T_0$ and $\text{defaultPat}(T_0, C_0, d) = Pat$, then there exists e such that $\text{match}(v, Pat) = e$.
Proof By strong induction on the depth of the derivation of $\text{defaultPat}(T_0, C_0, d) = Pat$. Case analysis of the last rule in the derivation.

- Case DefZero or DefTypeVar or DefFunType. Then Pat has the form $\_$, so by E-MatchWild we have $\text{match}(v, \_) = \{\}$.

- Case DefClassType. Then T_0 has the form $(\overline{T_0}\ C)$ and Pat has the form $(C\ \{\overline{V} = \overline{Pat}\})$ and $\text{repType}(\overline{T_0}\ C) = \{\overline{V} : \overline{T}\}$ and $\text{defaultPat}(\overline{T}, C_0, d-1) = \overline{Pat}$ and $d > 0$. Since $T = [\overline{Tn} \mapsto \overline{T}]T_0$, by Lemma 4.11 we have $\text{repType}(T) = [\overline{Tn} \mapsto \overline{T}]\{\overline{V} : \overline{T}\}$. Further, $T = [\overline{Tn} \mapsto \overline{T}](\overline{T_0}\ C) = ([\overline{Tn} \mapsto \overline{T}]\overline{T_0}\ C)$. Since $T' \leq T$, by Lemma 5.1 T' has the form $(\overline{T_1}\ C')$. Since $\vdash v : T'$, by T-Rep v has the form $(\overline{T_1}\ C')\ \{\overline{V_1} = \overline{v_1}\}$ and $\bullet \vdash (\overline{T_1}\ C')$ OK and $\text{repType}(\overline{T_1}\ C') = \{\overline{V_1} : \overline{T_1}\}$ and $\vdash \overline{v_1} : \overline{T_1'}$ and $\overline{T_1'} \leq \overline{T_1}$.

  Since $(\overline{T_1}\ C') \leq ([\overline{Tn} \mapsto \overline{T}]\overline{T_0}\ C)$, by Lemma 4.5 we have $C' \leq C$. Further, by Lemma 4.12 we have that $\{\overline{V_1} : \overline{T_1}\} = \{\overline{V} : [\overline{Tn} \mapsto \overline{T}]\overline{T}, \overline{V_2} : \overline{T_2}\}$. Therefore there is some prefix $\overline{T_3}$ of $\overline{T_1'}$ such that $\overline{T_3} \leq [\overline{Tn} \mapsto \overline{T}]\overline{T}$. Therefore there is some prefix $\overline{v_3}$ of $\overline{v_1}$ such that $\vdash \overline{v_3} : \overline{T_3}$ and $\overline{T_3} \leq [\overline{Tn} \mapsto \overline{T}]\overline{T}$ and $\text{defaultPat}(\overline{T}, C_0, d-1) = \overline{Pat}$. Therefore by induction, $\text{match}(\overline{v_3}, \overline{Pat}) = \overline{e}$. Therefore by E-MatchClass we have $\text{match}((\overline{T_1}\ C')\ \{\overline{V_1} = \overline{v_1}\}, (C\ \{\overline{V} = \overline{Pat}\})) = \bigcup \overline{e}$.

- Case DefTupType. Then T_0 has the form $T_1 * \cdots * T_k$ and Pat has the form $(Pat_1, \ldots, Pat_k)$ and for all $1 \leq i \leq k$ we have $\text{defaultPat}(T_i, C_0, d-1) = Pat_i$ and $d > 0$. Since $T' \leq [\overline{Tn} \mapsto \overline{T}](T_1 * \cdots * T_k)$, by Lemma 4.6 we have that T' has the form $T_1' * \cdots * T_k'$, where for all $1 \leq i \leq k$ we have $T_i' \leq [\overline{Tn} \mapsto \overline{T}]T_i$. Since $\vdash v : T'$, by T-Tup we have that v has the form $(v_1, \ldots, v_k)$ and for all $1 \leq i \leq k$ we have $\vdash v_i : T_i'$. Therefore by induction, for all $1 \leq i \leq k$ we have that there exists some e_i such that $\text{match}(v_i, Pat_i) = e_i$. Then by E-MatchTup we have $\text{match}(v, Pat) = e_1 \cup \cdots \cup e_k$.

Lemma 5.11 If $\text{CP}(Mt, v) = C_0$ and $C_0 \leq C$ and $\vdash v : T'$ and $T' \leq T$ and $T = [\overline{Tn} \mapsto \overline{T}]\hat{Mt}$ and $\text{defaultPat}(Mt, C, d) = Pat$, then there exists e such that $\text{match}(v, Pat) = e$.

Proof By strong induction on the depth of the derivation of $\text{defaultPat}(Mt, C, d) = Pat$. Case analysis of the last rule in the derivation.

- Case DefZero. Then Pat has the form $\_$, so by E-MatchWild we have $\text{match}(v, \_) = \{\}$.

- Case DefCPClassType. Then Mt has the form $\#(\overline{T_1}\ C')$ and Pat has the form $(C\ \{\overline{V} = \overline{Pat}\})$ and $\text{repType}(\overline{T_1}\ C) = \{\overline{V} : \overline{T}\}$ and $\text{defaultPat}(\overline{T}, C, d-1) = \overline{Pat}$ and $d > 0$. By Lemma 4.11 we have $\text{repType}([\overline{Tn} \mapsto \overline{T}]\overline{T_1}\ C) = [\overline{Tn} \mapsto \overline{T}]\{\overline{V} : \overline{T}\}$. Since $\text{CP}(\#(\overline{T_1}\ C'), v) = C_0$, by CPInstance we have that v is of the form $(\overline{T_0}\ C_0)\ \{\overline{V_1} = \overline{v_1}\}$.

  Since we're given that $\vdash v : T'$, by T-Rep we have that $T' = (\overline{T_0}\ C_0)$ and $\bullet \vdash (\overline{T_0}\ C_0)$ OK and $\text{repType}(\overline{T_0}\ C_0) = \{\overline{V_2} : \overline{T_2}\}$ and $\vdash \overline{v_1} : \overline{T_2'}$ and $\overline{T_2'} \leq \overline{T_2}$. We're given that $T' \leq T$, so that means $(\overline{T_0}\ C_0) \leq ([\overline{Tn} \mapsto \overline{T}]\overline{T_1}\ C')$, and by Lemma 4.4 we have $\overline{T_0} = [\overline{Tn} \mapsto \overline{T}]\overline{T_1}$. Since $C_0 \leq C$ and $\bullet \vdash (\overline{T_0}\ C_0)$ OK, by Lemma 4.7 we have $(\overline{T_0}\ C_0) \leq (\overline{T_0}\ C)$. Therefore by Lemma 4.12 we have $\{\overline{V_2} : \overline{T_2}\} = \{\overline{V} : [\overline{Tn} \mapsto \overline{T}]\overline{T}, \overline{V_3} : \overline{T_3}\}$.

  Therefore there is some prefix $\overline{v_3}$ of $\overline{v_1}$ and some prefix $\overline{T_3}$ of $\overline{T_2'}$ such that $\vdash \overline{v_3} : \overline{T_3}$ and $\overline{T_3} \leq [\overline{Tn} \mapsto \overline{T}]\overline{T}$ and $\text{defaultPat}(\overline{T}, C, d-1) = \overline{Pat}$, so by Lemma 5.10, there exists $\overline{e}$ such that $\text{match}(\overline{v_3}, \overline{Pat}) = \bigcup \overline{e}$. Finally, we're given $C_0 \leq C$, so by E-MatchClass we have $\text{match}((\overline{T_0}\ C_0)\ \{\overline{V_1} = \overline{v_1}\}, (C\ \{\overline{V} = \overline{Pat}\}))$ $= \bigcup \overline{e}$.

- Case DefTupType. Then Mt has the form $T_1 * \cdots * T_{i-1} * Mt_i * T_{i+1} * \cdots * T_k$ and Pat has the form $(Pat_1, \ldots, Pat_k)$ and for all $1 \leq j \leq k$ such that $j \neq i$ we have $\text{defaultPat}(T_j, C, d-1) = Pat_j$ and we have $\text{defaultPat}(Mt_i, C, d-1) = Pat_i$. Let $T_i = \hat{Mt}_i$. Since $T' \leq [\overline{Tn} \mapsto \overline{T}](T_1 * \cdots * T_k)$, by Lemma 4.6 we have that T' has the form $T_1' * \cdots * T_k'$, where for all $1 \leq j \leq k$ we have $T_j' \leq [\overline{Tn} \mapsto \overline{T}]T_j$. Since $\vdash v : T'$, by T-Tup we have that v has the form $(v_1, \ldots, v_k)$ and for all $1 \leq j \leq k$ we have $\vdash v_j : T_j'$. Therefore by Lemma 5.10, for all $1 \leq j \leq k$ such that $j \neq i$ we have that there exists some e_j such that $\text{match}(v_j, Pat_j) = e_j$. We're given that $\text{CP}(Mt, v) = C_0$, so by CPTupVal we have $\text{CP}(Mt_i, v_i) = C_0$. Therefore by induction we have that there exists some e_i such that $\text{match}(v_i, Pat_i) = e_i$. Then by E-MatchTup we have $\text{match}(v, Pat) = e_1 \cup \cdots \cup e_k$.

Lemma 5.12 If $\vdash v : T_2'$ and $T_2' \leq T_2$ and $T_2 = [\overline{Tn} \mapsto \overline{T}]\hat{Mt}$ and $(\texttt{fun}\ \overline{Tn}\ Fn : Mt \to T_0) \in BT(Bn)$ and $\text{CP}(Mt, v) = C_0$ and $C_0 \leq C$ and $\overline{Bn} \vdash Bn.Fn$ has-default-for C, then there exists some $Bn' \in \overline{Bn}$, some $(\texttt{extend fun}_{Mn}\ \overline{Tn_1}\ Bn.Fn\ Pat = E) \in BT(Bn')$, and some environment e such that $\text{match}(v, Pat) = e$.
Proof Since $\overline{Bn} \vdash Bn.Fn$ has-default-for C, by Default we have $\text{defaultPat}(Mt, C) = Pat'$ and by DefPat we have $\text{defaultPat}(Mt, C, d) = Pat'$. Therefore we have $\text{CP}(Mt, v) = C_0$ and $C_0 \leq C$ and $\vdash v : T_2'$ and $T_2' \leq T_2$ and $T_2 = [\overline{Tn} \mapsto \overline{T}]\hat{Mt}$ and $\text{defaultPat}(Mt, C, d) = Pat'$, so by Lemma 5.11 there exists e' such that $\text{match}(v, Pat') = e'$.

Also by Default we have $(\texttt{extend fun}_{Mn}\ \overline{Tn_1}\ Bn.Fn\ Pat = E) \in BT(Bn')$ and $Pat' \leq Pat$ and $Bn' \in \overline{Bn}$. By Lemma 5.7 there exists e such that $\text{match}(v, Pat) = e$, so the result follows.

Lemma 5.13 If $\vdash v : T'$ and $T' \leq T$ and $T = [\overline{Tn} \mapsto \overline{T}]\hat{Mt}$ and $\text{CP}(Mt) = C'$, then there exists some class C such that $\text{CP}(Mt, v) = C$ and $\text{concrete}(C)$ and $C \leq C'$.
Proof By induction on the depth of the derivation of $\vdash v : T'$. Case analysis of the last rule used in the derivation.

- Case T-Rep. Then v has the form $(\overline{T_0}\ C)\ \{\overline{V} = \overline{v}\}$ and $T' = (\overline{T_0}\ C)$ and $\text{concrete}(C)$ and $\text{repType}(\overline{T_0}\ C) = \{\overline{V} : \overline{T}\}$. Since $T' \leq T$, by Lemma 4.3 T has the form $(\overline{T_1}\ C'')$. Since $T = [\overline{Tn} \mapsto \overline{T}]\hat{Mt}$, $\hat{Mt}$ has the form $(\overline{T_2}\ C'')$, and by the grammar for marked types Mt must be $\#(\overline{T_2}\ C'')$. Then by CPInstance we have $\text{CP}(\#(\overline{T_2}\ C''), (\overline{T_0}\ C)\ \{\overline{V} = \overline{v}\}) = C$. We're given $T' \leq T$, so by Lemma 4.5 we have $C \leq C''$. Since $\text{CP}(Mt) = C'$, by CPClass we have $C' = C''$, so $C \leq C'$.

- Case T-Fun. Then v has the form $(\overline{T_1}\ F)$ and T' has the form $T_1 \to T_2$. Therefore by Lemma 5.2 T has the form $T_1' \to T_2'$. Since $T = [\overline{Tn} \mapsto \overline{T}]\hat{Mt}$, $\hat{Mt}$ has the form $T_1'' \to T_2''$, but this contradicts the grammar of marked types. Therefore, T-Fun cannot be the last rule in the derivation.

- Case T-Tup: Then v has the form $(v_1, \ldots, v_k)$ and T' has the form $T_1' * \cdots * T_k'$ and for all $1 \leq j \leq k$ we have $\vdash v_j : T_j'$. Therefore by Lemma 5.3 T has the form $T_1 * \cdots * T_k$, where for all $1 \leq j \leq k$ we have $T_j' \leq T_j$. Since $T = [\overline{Tn} \mapsto \overline{T}]\hat{Mt}$, $\hat{Mt}$ has the form $T_1'' * \cdots * T_k''$, and by the grammar for marked types Mt must have the form $T_1'' * \cdots * T_{i-1}'' * Mt_i * T_{i+1}'' * \cdots * T_k''$, where $1 \leq i \leq k$ and $\hat{Mt}_i = T_i''$. We're given $\text{CP}(Mt) = C'$, so by CPTup we have $\text{CP}(Mt_i) = C'$.

  Therefore we have $\vdash v_i : T_i'$ and $T_i' \leq T_i$ and $T_i = [\overline{Tn} \mapsto \overline{T}]\hat{Mt}_i$ and $\text{CP}(Mt_i) = C'$, so by induction there exists C such that $\text{CP}(Mt_i, v_i) = C$ and $\text{concrete}(C)$ and $C \leq C'$. By CPTupVal we have $\text{CP}(T_1'' * \cdots * T_{i-1}'' * Mt_i * T_{i+1}'' * \cdots * T_k'', (v_1, \ldots, v_k)) = C$, so the result follows.

Lemma 5.14 If $\vdash (\overline{T}\ F) : T_2 \to T$ and $\vdash v : T_2'$ and $T_2' \leq T_2$, then there exists some $Bn' \in \text{dom}(BT)$, some $(\texttt{extend fun}_{Mn}\ \overline{Tn_1}\ F\ Pat = E) \in BT(Bn')$, and some environment e such that $\text{match}(v, Pat) = e$.
Proof Since $\vdash (\overline{T}\ F) : T_2 \to T$, by T-Fun we have $F = Bn.Fn$ and $(\texttt{fun}\ \overline{Tn}\ Fn : Mt \to T_0) \in BT(Bn)$ and $|\overline{Tn}| = |\overline{T}|$ and $T_2 \to T = [\overline{Tn} \mapsto \overline{T}](\hat{Mt} \to T_0)$. Let $BT(Bn) = \texttt{block}\ Bn = \texttt{blk extends}\ \overline{Bn}\ \overline{Ood}\ \texttt{end}$. Then by BlockOK we have $\overline{Bn} \vdash (\texttt{fun}\ \overline{Tn}\ Fn : Mt \to T_0)$ OK in Bn, so by FunOK we have that $\text{CP}(Mt) = Bn''.Cn$. Then by Lemma 5.13 there exists some class C such that $\text{CP}(Mt, v) = C$ and $\text{concrete}(C)$ and $C \leq Bn''.Cn$. Also by FunOK we have either $\overline{Bn} \vdash F$ has-gdefault or $Bn = Bn''$. We consider these cases separately.

- Case $\overline{Bn} \vdash F$ has-gdefault. By GDefault we have $\text{CP}(F) = C'$ and $\overline{Bn} \vdash F$ has-default-for C'. By CPFun, $C' = Bn''.Cn$. Then by Lemma 5.12 there exists some $Bn' \in \overline{Bn}$, some $(\texttt{extend fun}_{Mn}\ \overline{Tn_1}$

$F\ Pat = E) \in BT(Bn')$, and some environment e such that match$(v, Pat) = e$. Since $BT(Bn) =$ `block` $Bn =$ `blk extends` $\overline{Bn}\ \overline{Ood}$ `end`, each member of $\overline{Bn}$ is mentioned in the program, so by sanity condition 2 we have $\overline{Bn} \subseteq$ dom(BT). Therefore $Bn' \in$ dom(BT), and the result is shown.

- Case $Bn = Bn''$. Let $C = Bn_0.Cn_0$. Since concrete(c), by CONCRETE we have (`class` $\overline{Tn_0}\ Cn_0 \ldots$) $\in BT(Bn_0)$. Let $BT(Bn_0) =$ `block` $Bn =$ `blk` Bn_0 `extends` $\overline{Bn_0}\ \overline{Ood_0}$ `end`. Then by BLOCKOK we have $\overline{Bn_0} \vdash$ `class` $\overline{Tn_0}\ Cn_0 \ldots$ OK in Bn_0, so by CLASSOK we have concrete$(C) \Rightarrow \overline{Bn_0} \vdash$ funs-have-ldefault-for C. Since we have shown that concrete(C) holds, we have $\overline{Bn_0} \vdash$ funs-have-ldefault-for C.

  Also by CLASSOK we have $\overline{Bn_0} \vdash C$ transExtended. Since $C \leq Bn''.Cn$ and $Bn'' = Bn$, by Lemma 5.8 we have $Bn \in \overline{Bn_0}$.

  Since $F = Bn.Fn$ and $Bn \in \overline{Bn_0}$, by FUNEXT we have $\overline{Bn_0} \vdash F$ extended. Since (`fun` $\overline{Tn}\ Fn : Mt \rightarrow T_0) \in BT(Bn)$ and CP$(Mt) = Bn.Cn$, by CPFUN we have CP$(F) = Bn.Cn$. Also, we showed above that $C \leq Bn.Cn$. Therefore, since $\overline{Bn_0} \vdash$ funs-have-ldefault-for C, by LDEFAULT we have $\overline{Bn_0} \vdash F$ has-default-for C. By SUBREF $C \leq C$, so by Lemma 5.12 there exists some $Bn' \in \overline{Bn_0}$, some (`extend fun`$_{M_n}$ $\overline{Tn_1}\ Bn.Fn\ Pat = E) \in BT(Bn')$, and some environment e such that match$(v, Pat) = e$. Since $BT(Bn_0) =$ `block` $Bn_0 =$ `blk extends` $\overline{Bn_0}\ \overline{Ood_0}$ `end`, each member of $\overline{Bn_0}$ is mentioned in the program, so by sanity condition (2) we have $\overline{Bn_0} \subseteq$ dom(BT). Therefore $Bn' \in$ dom(BT), and the result is shown.

5.3 Ambiguity

These lemmas ensure that all functions are unambiguous.

5.3.1 Pattern Specificity and Intersection

Lemma 5.15 If $Pat \leq Pat'$ and $Pat' \leq Pat''$ then $Pat \leq Pat''$.
Proof By induction on the depth of the derivation of $Pat' \leq Pat''$. Case analysis of the last rule used in the derivation.

- Case SPECWILD. Then Pat'' has the form _, and by SPECWILD we have $Pat \leq Pat''$.

- Case SPECBIND1. Then Pat' has the form (I `as` Pat'_0) and we have $Pat'_0 \leq Pat''$. We prove this case by induction on the number of consecutive uses of rule SPECBIND1 ending the derivation of $Pat \leq$ (I `as` Pat'_0). Case analysis of the last rule used in the derivation.

  - Case SPECBIND1. Then Pat has the form (I' `as` Pat_0) and $Pat_0 \leq Pat'$. By the inner induction $Pat_0 \leq Pat''$, and by SPECBIND1 $Pat \leq Pat''$.
  - Case SPECBIND2. Then $Pat \leq Pat'_0$. Since also $Pat'_0 \leq Pat''$, by the outer induction we have $Pat \leq Pat''$.

- Case SPECBIND2. Then Pat'' has the form (I `as` Pat''_0) and we have $Pat' \leq Pat''_0$. By induction $Pat \leq Pat''_0$, and by SPECBIND2 $Pat \leq Pat''$.

- Case SPECTUP. Then Pat' has the form $(\overline{Pat'})$ and Pat'' has the form $(\overline{Pat''})$ and $\overline{Pat'} \leq \overline{Pat''}$. We prove this case by induction on the number of consecutive uses of rule SPECBIND1 ending the derivation of $Pat \leq Pat'$. Case analysis of the last rule used in the derivation.

  - Case SPECBIND1. Then Pat has the form (I `as` Pat_0) and we have $Pat_0 \leq Pat'$. By the inner induction $Pat_0 \leq Pat''$, so by SPECBIND1 $Pat \leq Pat''$.

- Case SpecTup. Then Pat has the form $(\overline{Pat})$ $\overline{Pat} \leq \overline{Pat'}$. Therefore by the outer induction, $\overline{Pat} \leq \overline{Pat''}$. Therefore by SpecTup $Pat \leq Pat''$.

- Case SpecClass. Then Pat' has the form C' $\{\overline{V_1} = \overline{Pat'_1}, \overline{V_2} = \overline{Pat'_2}\}$ and Pat'' has the form C'' $\{\overline{V_1} = \overline{Pat''_1}\}$ and $C' \leq C''$ and $\overline{Pat'_1} \leq \overline{Pat''_1}$. We prove this case by induction on the number of consecutive uses of the rule SpecBind1 ending the derivation of $Pat \leq Pat'$. Case analysis of the last rule used in the derivation.

  - Case SpecBind1. Then Pat has the form $(I$ `as` $Pat_0)$ and we have $Pat_0 \leq Pat'$. By the inner induction $Pat_0 \leq Pat''$, so by SpecBind1 $Pat \leq Pat''$.
  - Case SpecClass. Then Pat has the form C $\{\overline{V_1} = \overline{Pat_1}, \overline{V_2} = \overline{Pat_2}, \overline{V_3} = \overline{Pat_3}\}$ and $C \leq C'$ and $\overline{Pat_1} \leq \overline{Pat'_1}$ and $\overline{Pat_2} \leq \overline{Pat'_2}$. Since $C \leq C'$ and $C' \leq C''$, by SubTrans we have $C \leq C''$. By the outer induction we have $\overline{Pat_1} \leq \overline{Pat''_1}$. Therefore by SpecClass $Pat \leq Pat''$.

Lemma 5.16 If $\mathrm{CP}(Mt, Pat') = C'$ and $\mathrm{CP}(Mt, Pat'') = C''$ and $Pat' \cap Pat'' = Pat$, then either $C' \leq C''$ or $C'' \leq C'$.
Proof By induction on the depth of the derivation of $Pat' \cap Pat'' = Pat$. Case analysis of the last rule used in the derivation.

- Case PatIntWild. Then Pat' has the form _. But then it cannot be the case that $\mathrm{CP}(Mt, Pat') = C'$, because none of the three associated rules applies to a wildcard pattern.

- Case PatIntBind. Then Pat' has the form I `as` Pat_0 and $Pat_0 \cap Pat'' = Pat$. Since $\mathrm{CP}(Mt, Pat') = C'$, by CPBindPat we have $\mathrm{CP}(Mt, Pat_0) = C'$. Therefore by induction we have that either $C' \leq C''$ or $C'' \leq C'$.

- Case PatIntTup. Then Pat' has the form $(Pat'_1, \ldots, Pat'_k)$ and Pat'' has the form $(Pat''_1, \ldots, Pat''_k)$ and for all $1 \leq j \leq k$ we have $Pat'_j \cap Pat''_j = Pat_j$. Since $\mathrm{CP}(Mt, Pat') = C'$, by CPTupPat we have $Mt = T_1 * \cdots * T_{i-1} * Mt_i * T_{i+1} * \cdots * T_k$ and $\mathrm{CP}(Mt_i, Pat'_i) = C'$. Since $\mathrm{CP}(Mt, Pat'') = C''$, by CPTupPat we have $\mathrm{CP}(Mt_i, Pat''_i) = C''$. Therefore by induction we have that either $C' \leq C''$ or $C'' \leq C'$.

- Case PatIntClass. Then Pat' has the form $(C_1$ $\{\overline{V} = \overline{Pat'}, \overline{V_2} = \overline{Pat_2}\})$ and Pat'' has the form $(C_2$ $\{\overline{V} = \overline{Pat''}\})$ and $C_1 \leq C_2$. Since $\mathrm{CP}(Mt, Pat') = C'$, by CPClassPat $C' = C_1$. Since $\mathrm{CP}(Mt, Pat'') = C''$, by CPClassPat $C'' = C_2$. Therefore $C' \leq C''$.

- Case PatIntRev. Then $Pat'' \cap Pat' = Pat$, so by induction we have that either $C'' \leq C'$ or $C' \leq C''$.

Lemma 5.17 If $\vdash v : T$ and $\mathrm{match}(v, Pat') = e'$ and $\mathrm{match}(v, Pat'') = e''$ and $\mathrm{matchType}(T', Pat') = \Gamma', T'_0$ and $\mathrm{matchType}(T'', Pat'') = \Gamma'', T''_0$, then there exists some Pat such that $Pat' \cap Pat'' = Pat$.
Proof By induction on the depth of the derivation of $\mathrm{match}(v, Pat') = e'$. Case analysis of the last rule used in the derivation.

- Case E-MatchWild. Then Pat' has the form _, so by PatIntWild we have $Pat' \cap Pat'' = Pat''$.

- Case E-MatchBind. Then Pat' has the form I `as` Pat'_0 and $\mathrm{match}(v, Pat'_0) = e'_0$, for some e'_0. Since $\mathrm{matchType}(T', Pat') = \Gamma', T'_0$, by T-MatchBind we have $\mathrm{matchType}(T', Pat'_0) = \Gamma'_0, T'_0$. Then by induction there exists some Pat such that $Pat'_0 \cap Pat'' = Pat$, so by PatIntBind we have $Pat' \cap Pat'' = Pat$.

- Case E-MATCHTUP. Then $v = (v_1, \ldots, v_k)$ and Pat' has the form $(Pat'_1, \ldots, Pat'_k)$ and for all $1 \leq i \leq k$ we have $\text{match}(v_i, Pat'_i) = e'_i$, for some e'_i. We prove this case by induction on the number of consecutive uses of E-MATCHBIND ending the derivation of $\text{match}(v, Pat'') = e''$. Case analysis of the last rule used in the derivation.

  - Case E-MATCHWILD. Then Pat'' has the form $\_$, so by PATINTWILD we have $Pat'' \cap Pat' = Pat'$, and by PATINTREV $Pat' \cap Pat'' = Pat'$.
  - Case E-MATCHBIND. Then Pat'' has the form I `as` Pat''_0 and $\text{match}(v, Pat''_0) = e''_0$, for some e''_0. Since $\text{matchType}(T'', Pat'') = \Gamma'', T''_0$, by T-MATCHBIND we have $\text{matchType}(T'', Pat''_0) = \Gamma''_0, T''_0$. Then by the inner induction there exists some Pat such that $Pat' \cap Pat''_0 = Pat$. Then by PATINTREV $Pat''_0 \cap Pat' = Pat$, by PATINTBIND $Pat'' \cap Pat' = Pat$, and again by PATINTREV $Pat' \cap Pat'' = Pat$.
  - Case E-MATCHTUP. Then Pat'' has the form $(Pat''_1, \ldots, Pat''_k)$ and for all $1 \leq i \leq k$ we have $\text{match}(v_i, Pat''_i) = e''_i$, for some e''_i. Since $\vdash v : T$, by T-TUP we have $T = T_1 * \cdots * T_k$ and $\vdash v_i : T_i$ for all $1 \leq i \leq k$. Since $\text{matchType}(T', Pat') = \Gamma', T'_0$ and $\text{matchType}(T'', Pat'') = \Gamma'', T''_0$, by T-MATCHTUP we have $T' = T'_1 * \cdots * T'_k$ and $T'' = T''_1 * \cdots * T''_k$ and for all $1 \leq i \leq k$ $\text{matchType}(T'_i, Pat') = \Gamma'_i, T'''_i$ and $\text{matchType}(T''_i, Pat'') = \Gamma''_i, T''''_i$. Then by the outer induction, for all $1 \leq i \leq k$ there exists Pat_i such that $Pat'_i \cap Pat''_i = Pat_i$. Then by PATINTTUP there exists Pat such that $Pat' \cap Pat'' = Pat$.
  - Case E-MATCHCLASS. Then $v = ((\overline{T}\ C)\ \{\overline{V} = \overline{v}\})$, contradicting our assumption that $v = (v_1, \ldots, v_k)$.

- Case E-MATCHCLASS. Then $v = ((\overline{T}\ C)\ \{V_1 = v_1, \ldots, V_k = v_k\})$ and Pat' has the form $(C'\ \{V_1 = Pat'_1, \ldots, V_m = Pat'_m\})$ and $C \leq C'$ and $m \leq k$ and for all $1 \leq i \leq m$ we have $\text{match}(v_i, Pat'_i) = e'_i$ for some e'_i. We prove this case by induction on the number of consecutive uses of E-MATCHBIND ending the derivation of $\text{match}(v, Pat'') = e''$. Case analysis of the last rule used in the derivation.

  - Case E-MATCHWILD. Then Pat'' has the form $\_$, so by PATINTWILD we have $Pat'' \cap Pat' = Pat'$, and by PATINTREV $Pat' \cap Pat'' = Pat'$.
  - Case E-MATCHBIND. Then Pat'' has the form I `as` Pat''_0 and $\text{match}(v, Pat''_0) = e''_0$, for some e''_0. Since $\text{matchType}(T'', Pat'') = \Gamma'', T''_0$, by T-MATCHBIND we have $\text{matchType}(T'', Pat''_0) = \Gamma''_0, T''_0$. Then by the inner induction there exists some Pat such that $Pat' \cap Pat''_0 = Pat$. Then by PATINTREV $Pat''_0 \cap Pat' = Pat$, by PATINTBIND $Pat'' \cap Pat' = Pat$, and again by PATINTREV $Pat' \cap Pat'' = Pat$.
  - Case E-MATCHTUP. Then $v = (\overline{v})$, contradicting our assumption that $v = ((\overline{T}\ C)\ \{V_1 = v_1, \ldots, V_k = v_k\})$.
  - Case E-MATCHCLASS. Then Pat'' has the form $(C''\ \{V_1 = Pat''_1, \ldots, V_p = Pat''_p\})$ and $C \leq C''$ and $p \leq k$ and for all $1 \leq i \leq p$ we have $\text{match}(v_i, Pat''_i) = e''_i$ for some e''_i. Since $\vdash v : T$, by T-REP we have $\bullet \vdash (\overline{T}\ C)$ OK and for all $1 \leq i \leq k$ we have $\vdash v_i : T_i$ for some T_i. Since $C \leq C'$ and $C \leq C''$, by Lemma 4.7 we have $\bullet \vdash (\overline{T}\ C')$ OK and $\bullet \vdash (\overline{T}\ C'')$ OK. Since $\text{matchType}(T', Pat') = \Gamma', T'_0$ and $\text{matchType}(T'', Pat'') = \Gamma'', T''_0$, by T-MATCHCLASS we have $\text{repType}(\overline{T_0}\ C')$ has the form $\{V_1 : T'_1, \ldots, V_m : T'_m\}$ and $\text{repType}(\overline{T_1}\ C'')$ has the form $\{V_1 : T''_1, \ldots, V_p : T''_p\}$, for some $\overline{T_0}$ and $\overline{T_1}$. Therefore by inspection of REPTYPE, also $\text{repType}(\overline{T}\ C')$ has the form $\{V_1 : T'''_1, \ldots, V_m : T'''_m\}$ and $\text{repType}(\overline{T}\ C'')$ has the form $\{V_1 : T''''_1, \ldots, V_p : T''''_p\}$. Also by T-MATCHCLASS, for all $1 \leq i \leq m$ we have $\text{matchType}(T'_i, Pat') = \Gamma'_i, T'''_i$ and for all $1 \leq i \leq p$ we have $\text{matchType}(T''_i, Pat'') = \Gamma''_i, T''''_i$. Since $C \leq C'$ and $C \leq C''$, by Lemma 5.4 either $C' \leq C''$ or $C'' \leq C'$.

* Case $C' \leq C''$. Since $\bullet \vdash (\overline{T}\ C')$ OK, by Lemma 4.7 we have $(\overline{T}\ C') \leq (\overline{T}\ C'')$. Then by Lemma 4.12 we have that $p \leq m$. Then by the outer induction we have that for all $1 \leq i \leq p$ there exists Pat_i such that $Pat'_i \cap Pat''_i = Pat_i$. Then by PatIntClass there exists Pat such that $Pat' \cap Pat'' = Pat$.
* Case $C'' \leq C'$. Since $\bullet \vdash (\overline{T}\ C'')$ OK, by Lemma 4.7 we have $(\overline{T}\ C'') \leq (\overline{T}\ C')$. Then by Lemma 4.12 we have that $m \leq p$. Then by the outer induction we have that for all $1 \leq i \leq m$ there exists Pat_i such that $Pat'_i \cap Pat''_i = Pat_i$. Then by PatIntRev we have that for all $1 \leq i \leq m$ there exists Pat_i such that $Pat''_i \cap Pat'_i = Pat_i$. Then by PatIntClass there exists Pat such that $Pat'' \cap Pat' = Pat$, and the result follows by PatIntRev.

Lemma 5.18 If $\text{match}(v, Pat') = e'$ and $\text{match}(v, Pat'') = e''$ and $Pat' \cap Pat'' = Pat$, then there exists some e such that $\text{match}(v, Pat) = e$.
Proof By induction on the depth of the derivation of $Pat' \cap Pat'' = Pat$. Case analysis of the last rule used in the derivation.

- Case PatIntWild. Then Pat is identical to Pat'', so $\text{match}(v, Pat) = e''$.

- Case PatIntBind. Then Pat' has the form I `as` Pat'_0 and $Pat'_0 \cap Pat'' = Pat$. Since $\text{match}(v, Pat') = e'$, by E-MatchBind there exists some e'_0 such that $\text{match}(v, Pat'_0) = e'_0$. Therefore by induction there exists some e such that $\text{match}(v, Pat) = e$.

- Case PatIntTup. Then Pat' has the form $(\overline{Pat'})$ and Pat'' has the form $(\overline{Pat''})$ and Pat has the form $(\overline{Pat})$ and $\overline{Pat'} \cap \overline{Pat''} = \overline{Pat}$. Since $\text{match}(v, Pat') = e'$, by E-MatchTup $v = (\overline{v})$ and $\text{match}(\overline{v}, \overline{Pat'}) = \overline{e'}$. Since $\text{match}(v, Pat'') = e''$, by E-MatchTup $\text{match}(\overline{v}, \overline{Pat''}) = \overline{e''}$. Therefore by induction $\text{match}(\overline{v}, \overline{Pat}) = \overline{e}$. Then by E-MatchTup there exists e such that $\text{match}(v, Pat) = e$.

- Case PatIntClass. Then Pat' has the form $(C'\ \{V_1 = Pat'_1, \ldots, V_m = Pat'_m\})$ and Pat'' has the form $(C''\ \{V_1 = Pat''_1, \ldots, V_p = Pat''_p\})$ and $m \geq p$ and Pat has the form $(C'\ \{V_1 = Pat_1, \ldots, V_p = Pat_p, V_{p+1} = Pat'_{p+1}, \ldots, V_m = Pat'_m\})$ and $C' \leq C''$ and $Pat'_i \cap Pat''_i = Pat_i$ for all $1 \leq i \leq m$. Since $\text{match}(v, Pat') = e'$, by E-MatchClass $v = ((\overline{T}\ C)\ \{V_1 = v_1, \ldots, V_k = v_k\})$ and $C \leq C'$ and $k \geq m$ and $\text{match}(v_i, Pat'_i) = e'_i$ for all $1 \leq i \leq m$. Since $\text{match}(v, Pat'') = e''$, by E-MatchClass we have $\text{match}(v_i, Pat''_i) = e''_i$ for all $1 \leq i \leq p$. Then by induction, there exists e_i such that $\text{match}(v_i, Pat_i) = e_i$, for all $1 \leq i \leq p$. Then by E-MatchClass there exists e such that $\text{match}(v, Pat) = e$.

- Case PatIntRev. Then $Pat'' \cap Pat' = Pat$. Then by induction there exists e such that $\text{match}(v, Pat) = e$.

5.3.2 Ambiguity

Lemma 5.19 If $\text{CP}(Mt, Pat) = Bn.Cn$ and $\overline{Tn} \vdash \text{matchType}(T, Pat) = (\Gamma, T')$, then there exists some (`<abstract> class` $\overline{Tn_0}$ $Cn\ldots) \in BT(Bn)$.
Proof By induction on the depth of the derivation of $\text{CP}(Mt, Pat) = Bn.Cn$. Case analysis of the last rule used in the derivation.

- Case CPBindPat. Then Pat has the form I `as` Pat' and $\text{CP}(Mt, Pat') = Bn.Cn$. Since $\overline{Tn} \vdash \text{matchType}(T, Pat) = (\Gamma, T')$, by T-MatchBind we have that there exists some Γ' such that $\overline{Tn} \vdash \text{matchType}(T, Pat') = (\Gamma', T')$. Therefore by induction there exists some (`<abstract> class` $\overline{Tn_0}$ $Cn\ldots) \in BT(Bn)$.

- Case CPTupPat. Then Pat has the form $(Pat_1, \ldots, Pat_k)$ and $Mt = T_1 * \cdots * T_{i-1} * Mt_i * T_{i+1} * \cdots * T_k$ and $\mathrm{CP}(Mt_i, Pat_i) = Bn.Cn$. Since $\overline{Tn} \vdash \mathrm{matchType}(T, Pat) = (\Gamma, T')$, by T-MatchTup there exist some T_i, Γ_i, and T_i' such that $\overline{Tn} \vdash \mathrm{matchType}(T_i, Pat_i) = (\Gamma_i, T_i')$. Therefore by induction there exists some (<abstract> `class` $\overline{Tn_0}$ $Cn \ldots$) $\in BT(Bn)$.

- Case CPClassPat. Then Pat has the form $Bn.Cn\ \{\overline{V} = \overline{Pat}\}$. Since $\overline{Tn} \vdash \mathrm{matchType}(T, Pat) = (\Gamma, T')$, by T-MatchClass we have $T = (\overline{T}\ C')$ and $\mathrm{repType}(\overline{T}\ C) = \{\overline{V} : \overline{T_1}\}$. Then by Rep there exists some (<abstract> `class` $\overline{Tn_0}$ $Cn \ldots$) $\in BT(Bn)$.

The following lemma says that the modular ambiguity checks for a function case are enough to ensure global unambiguity of the function case.

Lemma 5.20 If (`extend fun`$_{Mn}$ $\overline{Tn}\ F\ Pat = E$) $\in BT(Bn)$, then $\mathrm{dom}(BT) \vdash$ `extend fun`$_{Mn}$ $\overline{Tn}\ F\ Pat = E$ unambiguous in Bn.

Proof Suppose not. Then we have (`extend fun`$_{Mn}$ $\overline{Tn}\ F\ Pat = E$) $\in \overline{Ood}$, but it is not the case that $\mathrm{dom}(BT) \vdash$ `extend fun`$_{Mn}$ $\overline{Tn}\ F\ Pat = E$ unambiguous in Bn. Then by BlAmb we have that there exists some $Bn' \in \mathrm{dom}(BT)$, some (`extend fun`$_{Mn'}$ $\overline{Tn_1}\ F\ Pat' = E'$) $\in BT(Bn')$, and some Pat_0 such that $Pat \cap Pat' = Pat_0 \wedge Bn.Mn \neq Bn'.Mn' \wedge \neg\exists Bn'' \in \mathrm{dom}(BT).\exists($`extend fun`$_{Mn''}$ $\overline{Tn_2}\ F\ Pat'' = E'') \in BT(Bn'').(Pat_0 \leq Pat'' \wedge Pat'' \leq Pat \wedge Pat'' \leq Pat' \wedge (Pat \not\leq Pat'' \vee Pat' \not\leq Pat''))$.

Let $BT(Bn)$ be (`block` Bn = `blk extends` $\overline{Bn}\ \overline{Ood}$ `end`). Since (`extend fun`$_{Mn}$ $\overline{Tn}\ F\ Pat = E$) $\in BT(Bn)$, by BlockOK we have $\overline{Bn} \vdash$ (`extend fun`$_{Mn}$ $\overline{Tn}\ F\ Pat = E$) OK in Bn, so by CaseOK we have $Bn; \overline{Bn} \vdash$ `extend fun`$_{Mn}$ $\overline{Tn}\ F\ Pat = E$ unambiguous. Let $BT(Bn') =$ (`block` Bn' = `blk extends` $\overline{Bn'}\ \overline{Ood'}$ `end`). Since (`block` Bn' = `blk extends` $\overline{Bn'}\ \overline{Ood'}$ `end`) $= BT(Bn')$ and (`extend fun`$_{Mn'}$ $\overline{Tn_1}\ F\ Pat' = E'$) $\in BT(Bn')$, by BlockOK we have $\overline{Bn'} \vdash$ (`extend fun`$_{Mn'}$ $\overline{Tn_1}\ F\ Pat' = E'$) OK in Bn, so by CaseOK we have $Bn'; \overline{Bn'} \vdash$ `extend fun`$_{Mn'}$ $\overline{Tn_1}\ F\ Pat' = E'$ unambiguous.

We divide the proof into several cases.

- Case $Bn' \in \overline{Bn}$. Since $Bn; \overline{Bn} \vdash$ `extend fun`$_{Mn}$ $\overline{Tn}\ F\ Pat = E$ unambiguous, by Amb we have $\overline{Bn} \vdash$ `extend fun`$_{Mn}$ $\overline{Tn}\ F\ Pat = E$ unambiguous in Bn. Since $Bn' \in \overline{Bn}$ and we saw above that (`extend fun`$_{Mn'}$ $\overline{Tn_1}\ F\ Pat' = E'$) $\in BT(Bn')$ and $Pat \cap Pat' = Pat_0$ and $Bn.Mn \neq Bn'.Mn'$, by BlAmb we have $\exists Bn'' \in \overline{Bn}.\exists($`extend fun`$_{Mn''}$ $\overline{Tn_2}\ F\ Pat'' = E'') \in BT(Bn'').(Pat_0 \leq Pat'' \wedge Pat'' \leq Pat \wedge Pat'' \leq Pat' \wedge (Pat \not\leq Pat'' \vee Pat' \not\leq Pat''))$. Since (`block` = `blk` Bn `extends` $\overline{Bn}\ \overline{Ood}$ `end`) $= BT(Bn)$, each block name in $\overline{Bn}$ appears in the program, so by sanity condition 2 we have $\overline{Bn} \subseteq \mathrm{dom}(BT)$. Therefore we have $\exists Bn'' \in \mathrm{dom}(BT).\exists($`extend fun`$_{Mn''}$ $\overline{Tn_2}\ F\ Pat'' = E'') \in BT(Bn'').(Pat_0 \leq Pat'' \wedge Pat'' \leq Pat \wedge Pat'' \leq Pat' \wedge (Pat \not\leq Pat'' \vee Pat' \not\leq Pat''))$, and we have a contradiction.

- Case $Bn \in \overline{Bn'}$. Since $Bn'; \overline{Bn'} \vdash$ `extend fun`$_{Mn'}$ $\overline{Tn_1}\ F\ Pat' = E'$ unambiguous, by Amb we have $\overline{Bn'} \vdash$ `extend fun`$_{Mn'}$ $\overline{Tn_1}\ F\ Pat' = E'$ unambiguous in Bn'. By assumption $Bn \in \overline{Bn'}$, and we're given that (`extend fun`$_{Mn}$ $\overline{Tn}\ F\ Pat = E$) $\in BT(Bn)$. We're also given $Pat \cap Pat' = Pat_0$, so by PatIntRev also $Pat' \cap Pat = Pat_0$. Finally, we're given $Bn.Mn \neq Bn'.Mn'$. Therefore by BlAmb we have $\exists Bn'' \in \overline{Bn'}.\exists($`extend fun`$_{Mn''}$ $\overline{Tn_2}\ F\ Pat'' = E'') \in BT(Bn'').(Pat_0 \leq Pat'' \wedge Pat'' \leq Pat' \wedge Pat'' \leq Pat \wedge (Pat \not\leq Pat'' \vee Pat' \not\leq Pat''))$. Since (`block` = `blk` Bn' `extends` $\overline{Bn'}\ \overline{Ood'}$ `end`) $= BT(Bn')$, each block name in $\overline{Bn'}$ appears in the program, so by sanity condition 2 we have $\overline{Bn'} \subseteq \mathrm{dom}(BT)$. Therefore we have $\exists Bn'' \in \mathrm{dom}(BT).\exists($`extend fun`$_{Mn''}$ $\overline{Tn_2}\ F\ Pat'' = E'') \in BT(Bn'').(Pat_0 \leq Pat'' \wedge Pat'' \leq Pat \wedge Pat'' \leq Pat' \wedge (Pat \not\leq Pat'' \vee Pat' \not\leq Pat''))$, and we have a contradiction.

- Case $Bn' \notin \overline{Bn}$ and $Bn \notin \overline{Bn'}$. Since $Bn; \overline{Bn} \vdash$ `extend fun`$_{Mn}$ $\overline{Tn}\ F\ Pat = E$ unambiguous, by Amb we have $F = Bn_1.Fn$ and (`fun` $\overline{Tn_3}\ Fn : Mt \to T$) $\in BT(Bn_1)$ and $\mathrm{CP}(Mt, Pat) = Bn_2.Cn$ and

$Bn = Bn_1 \vee Bn = Bn_2$. Since $Bn'; \overline{Bn'} \vdash$ `extend fun`$_{Mn'}$ $\overline{Tn_1}$ F $Pat' = E'$ unambiguous, by AMB we have $\mathrm{CP}(Mt, Pat') = Bn_3.Cn'$ and $Bn' = Bn_1 \vee Bn' = Bn_3$. We have three sub-cases.

- Case $Bn' = Bn_1$. Since $\overline{Bn} \vdash$ (`extend fun`$_{Mn}$ $\overline{Tn}$ F $Pat = E$) OK in Bn, by CASEOK we have $\overline{Bn} \vdash F$ extended, so by FUNEXT we have $Bn_1 \in \overline{Bn}$. Therefore we've shown $Bn' \in \overline{Bn}$, so we have a contradiction.
- Case $Bn = Bn_1$. Since $\overline{Bn'} \vdash$ (`extend fun`$_{Mn'}$ $\overline{Tn_1}$ F $Pat' = E'$) OK in Bn', by CASEOK we have $\overline{Bn'} \vdash F$ extended, so by FUNEXT we have $Bn_1 \in \overline{Bn'}$. Therefore we've shown $Bn \in \overline{Bn'}$, so we have a contradiction.
- Case $Bn' \neq Bn_1$ and $Bn \neq Bn_1$. Since $Bn = Bn_1 \vee Bn = Bn_2$, we have $Bn = Bn_2$. Since $Bn' = Bn_1 \vee Bn' = Bn_3$, we have $Bn' = Bn_3$. Since $\mathrm{CP}(Mt, Pat) = Bn_2.Cn$ and $\mathrm{CP}(Mt, Pat') = Bn_3.Cn'$ and $Pat \cap Pat' = Pat_0$, by Lemma 5.16 we have that either $Bn_2.Cn \leq Bn_3.Cn'$ or $Bn_3.Cn' \leq Bn_2.Cn$. Equivalently, either $Bn.Cn \leq Bn'.Cn'$ or $Bn'.Cn' \leq Bn.Cn$. There are two subcases.
  - Case $Bn.Cn \leq Bn'.Cn'$. Since $\overline{Bn} \vdash$ (`extend fun`$_{Mn}$ $\overline{Tn}$ F $Pat = E$) OK in Bn, by CASEOK we have $\overline{Tn_0} \vdash \mathrm{match}(T_0, Pat) = (\Gamma_0, T_0')$, for some $\overline{Tn_0}, T_0, Pat, \Gamma_0$, and T_0'. Since $\mathrm{CP}(Mt, Pat) = Bn.Cn$, by Lemma 5.19 there exists some (`<abstract> class` $\overline{Tn_4}$ $Cn\ldots$) $\in BT(Bn)$. Therefore by BLOCKOK we have $\overline{Bn} \vdash$ (`<abstract> class` $\overline{Tn_4}$ $Cn\ldots$) OK in Bn, so by CLASSOK we have $\overline{Bn} \vdash Bn.Cn$ transExtended. Since $Bn.Cn \leq Bn'.Cn'$, by Lemma 5.8 we have $Bn' \in \overline{Bn}$, which is a contradiction.
  - Case $Bn'.Cn' \leq Bn.Cn$. Since $\overline{Bn'} \vdash$ (`extend fun`$_{Mn'}$ $\overline{Tn_1}$ F $Pat' = E'$) OK in Bn', by CASEOK we have $\overline{Tn_0} \vdash \mathrm{match}(T_0, Pat') = (\Gamma_0, T_0')$, for some $\overline{Tn_0}, T_0, Pat, \Gamma_0$, and T_0'. Since $\mathrm{CP}(Mt, Pat') = Bn'.Cn'$, by Lemma 5.19 there exists some (`<abstract> class` $\overline{Tn_4}$ $Cn'\ \ldots$) $\in BT(Bn')$. Therefore by BLOCKOK we have $\overline{Bn'} \vdash$ (`<abstract> class` $\overline{Tn_4}$ $Cn'\ \ldots$) OK in Bn', so by CLASSOK we have $\overline{Bn'} \vdash Bn'.Cn'$ transExtended. Since $Bn'.Cn' \leq Bn.Cn$, by Lemma 5.8 we have $Bn \in \overline{Bn'}$, which is a contradiction.

Lemma 5.21 If $\vdash v : T$ and $Bn \in \mathrm{dom}(BT)$ and (`extend fun`$_{Mn}$ $\overline{Tn}$ F $Pat = E$) $\in BT(Bn)$ and $\mathrm{match}(v, Pat) = e$, then there exists some $Bn' \in \mathrm{dom}(BT)$, some (`extend fun`$_{Mn'}$ $\overline{Tn_1}$ F $Pat' = E'$) $\in BT(Bn')$, and some e' such that $\mathrm{match}(v, Pat') = e$ and $\forall Bn'' \in \mathrm{dom}(BT).\forall$(`extend fun`$_{Mn''}$ $\overline{Tn_2}$ F $Pat'' = E''$) $\in BT(Bn'')$. $\forall e''.((\mathrm{match}(v, Pat'') = e' \wedge Bn'.Mn' \neq Bn''.Mn'') \Rightarrow Pat' < Pat'')$.

Proof By (strong) induction on the number of function cases of the form (`extend fun`$_{Mn_0}$ $\overline{Tn_0}$ F $Pat_0 = E_0$) such that (`extend fun`$_{Mn_0}$ $\overline{Tn_0}$ F $Pat_0 = E_0$) $\in BT(Bn_0)$ for some block $Bn_0 \in \mathrm{dom}(BT)$, and $\mathrm{match}(v, Pat_0) = e_0$ for some e_0, and $Pat \not< Pat_0$.

- Case there are zero function cases of the form (`extend fun`$_{Mn_0}$ $\overline{Tn_0}$ F $Pat_0 = E_0$) such that (`extend fun`$_{Mn_0}$ $\overline{Tn_0}$ F $Pat_0 = E_0$) $\in BT(Bn_0)$ for some block $Bn_0 \in \mathrm{dom}(BT)$, and $\mathrm{match}(v, Pat_0) = e_0$ for some e_0, and $Pat \not< Pat_0$.

  We're given that $Bn \in \mathrm{dom}(BT)$ and (`extend fun`$_{Mn}$ $\overline{Tn}$ F $Pat = E$) $\in BT(Bn)$ and $\mathrm{match}(v, Pat) = e$. Further, since it cannot both be the case that $Pat \leq Pat$ and $Pat \not\leq Pat$, we have $Pat \not< Pat$. Therefore, we have found a function case that contradicts the initial assumption of this case.

- Case there is exactly one function case of the form (`extend fun`$_{Mn_0}$ $\overline{Tn_0}$ F $Pat_0 = E_0$) such that (`extend fun`$_{Mn_0}$ $\overline{Tn_0}$ F $Pat_0 = E_0$) $\in BT(Bn_0)$ for some block $Bn_0 \in \mathrm{dom}(BT)$, and $\mathrm{match}(v, Pat_0) = e_0$ for some e_0, and $Pat \not< Pat_0$.

  As we saw in the previous case, (`extend fun`$_{Mn}$ $\overline{Tn}$ F $Pat = E$) $\in BT(Bn)$ and $\mathrm{match}(v, Pat) = e$ and $Pat \not< Pat$, so $Bn.Mn$ is the single case satisfying all the conditions. Therefore it follows that $\forall Bn'' \in \mathrm{dom}(BT).\forall(\texttt{extend fun}_{Mn''}\ \overline{Tn_2}\ F\ Pat'' = E'') \in BT(Bn'').\forall e''.((\mathrm{match}(v, Pat'') = e' \wedge Bn.Mn \neq Bn''.Mn'') \Rightarrow Pat < Pat'')$. Then the result follows.

- There are $k > 1$ function cases of the form $(\texttt{extend fun}_{Mn_0}\ \overline{Tn_0}\ F\ Pat_0 = E_0)$ such that $(\texttt{extend fun}_{Mn_0}\ \overline{Tn_0}\ F\ Pat_0 = E_0) \in BT(Bn_0)$ for some block $Bn_0 \in \mathrm{dom}(BT)$, and $\mathrm{match}(v, Pat_0) = e_0$ for some e_0, and $Pat \not< Pat_0$. Let $(\texttt{extend fun}_{Mn_1}\ \overline{Tn_3}\ F\ Pat_1 = E_1)$ be one such function case, so $(\texttt{extend fun}_{Mn_1}\ \overline{Tn_3}\ F\ Pat_1 = E_1) \in BT(Bn_1)$ for some block $Bn_1 \in \mathrm{dom}(BT)$, and $\mathrm{match}(v, Pat_1) = e_1$ for some e_1, and $Pat \not< Pat_1$. Since $k > 1$, at least one of the function cases satisfying the conditions is not $Bn.Mn$, so assume WLOG that $Bn.Mn \neq Bn_1.Mn_1$.

  Since $(\texttt{extend fun}_{Mn}\ \overline{Tn}\ F\ Pat = E) \in BT(Bn)$ and $(\texttt{extend fun}_{Mn_1}\ \overline{Tn_3}\ F\ Pat_1 = E_1) \in BT(Bn_1)$ and $Bn \in \mathrm{dom}(BT)$ and $Bn_1 \in \mathrm{dom}(BT)$, by CaseOK we have $\mathrm{matchType}(T_0, Pat) = \Gamma_0, T_0'$ and $\mathrm{matchType}(T_1, Pat_1) = \Gamma_1, T_1'$. We're given that $\vdash v : T$. Finally, we saw above that $\mathrm{match}(v, Pat) = e$ and $\mathrm{match}(v, Pat_1) = e_1$. Therefore by Lemma 5.17 there exists some Pat_{int} such that $Pat \cap Pat_1 = Pat_{int}$. We're given that $(\texttt{extend fun}_{Mn}\ \overline{Tn}\ F\ Pat = E) \in BT(Bn)$, so by Lemma 5.20 we have $\mathrm{dom}(BT) \vdash \texttt{extend fun}_{Mn}\ \overline{Tn}\ F\ Pat = E$ unambiguous in Bn. Therefore by BlAmb there exists some $Bn_2 \in \mathrm{dom}(BT)$ and some $(\texttt{extend fun}_{Mn_2}\ \overline{Tn_4}\ F\ Pat_2 = E_2) \in BT(Bn_2)$ such that $Pat_{int} \leq Pat_2$ and $Pat_2 \leq Pat$ and $Pat_2 \leq Pat_1$ and ($Pat \not\leq Pat_2$ or $Pat_1 \not\leq Pat_2$). Since $\mathrm{match}(v, Pat) = e$ and $\mathrm{match}(v, Pat_1) = e_1$ and $Pat \cap Pat_1 = Pat_{int}$, by Lemma 5.18 there exists some e_{int} such that $\mathrm{match}(v, Pat_{int}) = e_{int}$. Then since $Pat_{int} \leq Pat_2$, by Lemma 5.7 there exists e_2 such that $\mathrm{match}(v, Pat_2) = e_2$.

  So we have shown there exists some $Bn_2 \in \mathrm{dom}(BT)$ and some $(\texttt{extend fun}_{Mn_2}\ \overline{Tn_4}\ F\ Pat_2 = E_2) \in BT(Bn_2)$ and some e_2 such that $\mathrm{match}(v, Pat_2) = e_2$. Suppose there are l function cases of the form $(\texttt{extend fun}_{Mn_0}\ \overline{Tn_0}\ F\ Pat_0 = E_0)$ such that $(\texttt{extend fun}_{Mn_0}\ \overline{Tn_0}\ F\ Pat_0 = E_0) \in BT(Bn_0)$ for some block $Bn_0 \in \mathrm{dom}(BT)$, and $\mathrm{match}(v, Pat_0) = e_0$ for some e_0, and $Pat_2 \not< Pat_0$. If $l < k$, then this case is proven by induction.

  Consider some block $Bn_0 \in \mathrm{dom}(BT)$, some $(\texttt{extend fun}_{Mn_0}\ \overline{Tn_0}\ F\ Pat_0 = E_0) \in BT(Bn_0)$, and some e_0 such that $\mathrm{match}(v, Pat_0) = e_0$ and $Pat_2 \not< Pat_0$. I claim that also $Pat \not< Pat_0$. Since $Pat_2 \not< Pat_0$, we have that ($Pat_2 \not\leq Pat_0$ or $Pat_0 \leq Pat_2$), so we consider these cases in turn.

  - Case $Pat_2 \not\leq Pat_0$. Then I claim that $Pat \not\leq Pat_0$, so also $Pat \not< Pat_0$. Suppose not, so $Pat \leq Pat_0$. Since $Pat_2 \leq Pat$, by Lemma 5.15 we have $Pat_2 \leq Pat_0$, contradicting the assumption of this case.
  - Case $Pat_0 \leq Pat_2$. We showed above that $Pat_2 \leq Pat$, so by Lemma 5.15 $Pat_0 \leq Pat$, so $Pat \not< Pat_0$.

  Therefore we have shown that every function case of the appropriate form with respect to $Bn_2.Mn_2$ is also of the appropriate form with respect to $Bn.Mn$, so $l \leq k$.

  To finish the proof, we show that there exists a function case of the appropriate form w.r.t. $Bn.Mn$ that is not of the appropriate form w.r.t. $Bn_2.Mn_2$. In particular, we showed in the first case above that $Bn.Mn$ is of the appropriate form w.r.t. itself, since $Pat \not< Pat$. To show that $Bn.Mn$ is not of the appropriate form w.r.t $Bn_2.Mn_2$, we must show that $Pat_2 < Pat$. We showed above that $Pat_2 \leq Pat$, so we simply need to prove that $Pat \not\leq Pat_2$. We showed above that either $Pat \not\leq Pat_2$ or $Pat_1 \not\leq Pat_2$, so we consider each case.

  - Case $Pat \not\leq Pat_2$. Then $Pat \not\leq Pat_2$.
  - Case $Pat_1 \not\leq Pat_2$ and $Pat \leq Pat_2$. We're given above that $Pat \not< Pat_1$, so either $Pat \not\leq Pat_1$ or $Pat_1 \leq Pat$. We saw above that $Pat_2 \leq Pat_1$, so since we assume $Pat \leq Pat_2$, by Lemma 5.15 we have $Pat \leq Pat_1$. Therefore $Pat_1 \leq Pat$. Again since we assume $Pat \leq Pat_2$, by Lemma 5.15 we have $Pat_1 \leq Pat_2$, contradicting the assumption of this case.

Lemma 5.22 If $\vdash (\overline{T}\ F) : T_2 \rightarrow T$ and $\vdash v : T_2'$ and $T_2' \leq T_2$ then there exist e_0 and E_0 such that most-specific-case-for $((\overline{T}\ F), v) = (e_0, E_0)$.
Proof By Lemma 5.14, there exists some $Bn \in \mathrm{dom}(BT)$, some $(\texttt{extend fun}_{Mn}\ \overline{Tn}\ F\ Pat = E) \in BT(Bn)$, and some environment e such that $\mathrm{match}(v, Pat) = e$. Then by Lemma 5.21 there exists some $Bn' \in \mathrm{dom}(BT)$, some $(\texttt{extend fun}_{Mn'}\ \overline{Tn_1}\ F\ Pat' = E') \in BT(Bn')$, and some e' such that $\mathrm{match}(v, Pat') = e$ and $\forall Bn'' \in \mathrm{dom}(BT). \forall (\texttt{extend fun}_{Mn''}\ \overline{Tn_2}\ F\ Pat'' = E'') \in BT(Bn''). \forall e''. ((\mathrm{match}(v, Pat'') = e' \wedge Bn'.Mn' \neq Bn''.Mn'') \Rightarrow Pat' \leq Pat'' \wedge Pat'' \not\leq Pat')$. Since $\vdash (\overline{T}\ F) : T_2 \rightarrow T$, by T-Fun we have $F = Bn_0.Fn_0$ and $(\texttt{fun}\ \overline{Tn_0}\ Fn_0 : Mt_0 \rightarrow T_0)$ and $|\overline{Tn_0}| = |\overline{T}|$. Since $(\texttt{extend fun}_{Mn'}\ \overline{Tn_1}\ F\ Pat' = E') \in BT(Bn')$, by CaseOK we have $|\overline{Tn_1}| = |\overline{Tn_0}|$. Therefore we have $|\overline{Tn_1}| = |\overline{T}|$, so by Lookup there exists some e_0 and E_0 such that most-specific-case-for $((\overline{T}\ F), v) = (e_0, E_0)$.

5.4 Progress

Theorem 5.1 (Progress): If $\vdash E : T$ and E is not a value, then there exists an E' such that $E \longrightarrow E'$.
Proof By (strong) induction on the depth of the derivation of $\vdash E : T$. Case analysis of the last rule used in the derivation.

- Case T-Id. Then $E = I$ and $(I, T) \in \{\}$, so we have a contradiction. Therefore this rule could not be the last rule used in the derivation.
- Case T-New. Then $E = Ct(\overline{E})$ and $Ct = (\overline{T}\ Bn.Cn)$ and $\bullet \vdash Ct(\overline{E})$ OK and concrete$(Bn.Cn)$. Then by T-Super also $\bullet \vdash (\overline{T}\ Bn.Cn)$ OK and and $(\texttt{<abstract> class}\ \overline{Tn_0}\ Cn(\overline{I_0} : \overline{T_0})\ \ldots) \in BT(Bn)$ and $|\overline{I_0}| = |\overline{E}|$. Therefore by Lemma 5.9 rep$(Ct(\overline{E})$ is well-defined and has the form $\{\overline{V_1 = E_1}\}$. Then by E-New we have $E \longrightarrow Ct\ \{\overline{V_1 = E_1}\}$.
- Case T-Rep. Then $E = Ct\ \{V_1 = E_1, \ldots, V_k = E_k\}$ and for all $1 \leq i \leq k$ we have $\vdash E_i : T_i$ for some T_i. We have two subcases:
  - For all $1 \leq i \leq k$, E_i is a value. Then E is a value, contradicting our assumption.
  - There exists $1 \leq j \leq k$ such that E_j is not a value. By induction, there exists an E_j' such that $E_j \longrightarrow E_j'$. Therefore by E-Rep we have $Ct\ \{V_1 = E_1, \ldots, V_k = E_k\} \longrightarrow Ct\ \{V_1 = E_1, \ldots, V_{j-1} = E_{j-1}, V_j = E_j', V_{j+1} = E_{j+1}, \ldots, V_k = E_k\}$.
- Case T-Fun. Then $E = \overline{T}\ Bn.Fn$. Then E is a value, contradicting our assumption.
- Case T-Tup. Then $E = (E_1, \ldots, E_k)$ and $T = T_1 * \cdots * T_k$ and for all $1 \leq i \leq k$ we have $\vdash E_i : T_i$. We have two subcases:
  - For all $1 \leq i \leq k$, E_i is a value. Then E is a value, contradicting our assumption.
  - There exists $1 \leq j \leq k$ such that E_j is not a value. By induction, there exists an E_j' such that $E_j \longrightarrow E_j'$. Therefore by E-Tup we have $(E_1, \ldots, E_k) \longrightarrow (E_1, \ldots, E_{j-1}, E_j', E_{j+1}, \ldots, E_k)$.
- Case T-App. Then $E = E_1\ E_2$ and $\vdash E_1 : T_2 \rightarrow T$ and $\vdash E_2 : T_2'$ and $T_2' \leq T_2$. We have three subcases:
  - E_1 is not a value. Then by induction, there exists an E_1' such that $E_1 \longrightarrow E_1'$. Therefore by E-App1 we have $E_1\ E_2 \longrightarrow E_1'\ E_2$.
  - E_2 is not a value. Then by induction, there exists an E_2' such that $E_2 \longrightarrow E_2'$. Therefore by E-App2 we have $E_1\ E_2 \longrightarrow E_1\ E_2'$.

– Both E_1 and E_2 are values. Since $\vdash E_1 : T_2 \rightarrow T$ and E_1 is a value, the last rule in the derivation of $\vdash E_1 : T_2 \rightarrow T$ must be T-FUN, so E_1 has the form *Fv*. Therefore by Lemma 5.22 we have that there exist e_0 and E_0 such that most-specific-case-for $(Fv, E_2) = (e_0, E_0)$. Let $e_0 = \{(\overline{I}, \overline{v})\}$. Then by E-APPRED we have $Fv\ E_2 \longrightarrow [\overline{I} \mapsto \overline{v}]E_0$.