Semantic Types for Classes and Mixins

Ugo de'Liguoro
Department of Computer Science
University of Turin, Italy
ugo.deliguoro@unito.it

Tzu-chun Chen
Department of Computer Science
University of Turin, Italy
chen@unito.it

We consider a formalization of mixin composition and class linearization yielding a class in mixin-based inheritance. We provide an interpretation of the mixin calculus into a λ-calculus extended with records and a record merge operator. After extending the BCD intersection type assignment to such calculus showing that types are preserved by subject expansion and reduction, we naturally interpret mixin terms as the sets of the types that can be deduced of their translations. It turns out that the class obtained from a composition of mixins and the composition itself have the same logical meaning.

1 Introduction

Mixin inheritance has been proposed since the late 80's to enhance modularity and code reusability in the context of object-oriented class based programming languages. A mixin is a heir class which is parametric in a superclass, to which it may add or override both fields and methods. Mixin inheritance also provides a solution to the problem of ambiguities with multiple inheritance, as the programmer is provided with more control over class linearization.

To provide mathematical foundations to the complex mechanism on which mixins rely and to the sophisticated typing systems that mixin based programs require, programming languages have been seen as sugared λ-calculi with records, adding a certain amount of new operators among which the *record merge* operator plays a central role. An extensive study is [7].

Intersection types are considered as semantic types because they can be interpreted as the compact points of certain Scott domains, and are such that the meaning of λ-terms is characterized by the types that can be assigned in suitable type assignment systems, of which that one in [4] is the paradigmatic example. We advocate the claim that intersection types can serve to understand the semantics of mixins and to clarify their typing discipline.

In [9, 10, 3] intersection type systems have been shown to be a tool for modeling λ-calculi with records in such a way that the meaning of a term can be understood as the set of its types, much like it happens with filter-models of the ordinary λ-calculus [4]. Moreover the type system can be extended to the object-calculus of [1] via its interpretation into the λ-calculus with records.

This approach scales to Featherweight Java [13], that is a simplified class-based language, as shown in [16], where intersection types can be deduced directly for Featherweight Java expressions. This is however insufficient for our purposes, since classes have no explicit type in that system. On the contrary, if we want to characterize the meaning of the class resulting from mixin composition, classes and mixins must be first class objects in our formalism.

Toward devising an intersection type assignment system for classes and mixins, we introduce in this paper a (very) simplified calculus to formalize mixin application and composition in a Featherweight Java like syntax. We then express class linearization as a reduction relation that we understand as a pre-compilation process, that precedes actual execution of its instances, something which we do not consider

Submitted to:
ITRS 2014

here. To rule out meaningless overridings we also introduce a typing system, extending that one of Featherweight Java, and establish its correctness w.r.t. the compilation relation.

Turning to the semantic type, we do not propose a system to assign types directly to terms of the mixin calculus. Rather we refine the λ-calculus with records and the intersection type assignment system from [9], in such a way that the mixin calculus can be easily interpretated into the λ-calculus, obtaining in this way a logical semantics for the mixins thought indirectly. In fact we interpret a term as the set of intersection types that can be assigned to its translation; building over the invariance of typing both under reduction and expansion in intersection type systems, we ultimately prove that any mixin term and the class to which it reduces have the same meaning.

2 A calculus of classes and mixins

We consider a calculus modeling class declaration via mixin inheritance, which is inspired to Featherweight Java [13], hereafter FJ, Jam [2] and Scala [14], but much coarser than those, to feature only the field/method extension and overriding, on which we concentrate. Terms in the calculus represent flat classes (namely without the `extends` construct of Java like languages) that are considered as the values to which a composition of mixins applied to some initial (possibly empty) class eventually reduces. The reduction relation, on the other hand, doesn't represent actual computation, rather it is abstract precompilation, computing the class linearization; here we follow [6], again by a non full-fledged calculus.

Definition 2.1 (Class and Mixin Terms) Let $\mathtt{C},\mathtt{D}$ vary over a denumerable set of class names:

$$\begin{array}{rcll} \mathtt{T} & ::= & \mathtt{class\ C\ R} & \text{(class)} \\ & | & \mathtt{mixin\ C = D\ with\ R} & \text{(mixin)} \\ & | & \mathtt{T} \diamond \mathtt{T} & \text{(inheritance)} \end{array} \qquad \begin{array}{rcll} \mathtt{R} & ::= & \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \bar{\mathtt{M}}\} & \text{(record)} \\ & | & \mathtt{R} \bullet \mathtt{R} & \text{(merge)} \\ \mathtt{M} & ::= & \mathtt{C\ m}(\bar{\mathtt{D}}\ \bar{x})\{e\} & \text{(method)} \end{array}$$

Since we do not consider the execution model, we leave method and expression syntax unspecified.

We say that $\mathtt{C\ f}$ and $\mathtt{D\ g}$ match if $\mathtt{f} \equiv \mathtt{g}$. Similarly we say that $\mathtt{C\ m}(\bar{\mathtt{D}}\ \bar{x})\{e\}$ and $\mathtt{E\ n}(\bar{\mathtt{F}}\ \bar{x})\{e'\}$ match if $\mathtt{m} \equiv \mathtt{n}$. By $\bar{\mathtt{C}}\ \bar{\mathtt{f}} \backslash \bar{\mathtt{D}}\ \bar{\mathtt{g}}$ we denote the sequence of the field declarations in $\bar{\mathtt{C}}\ \bar{\mathtt{f}}$ that do not match with those in $\bar{\mathtt{D}}\ \bar{\mathtt{g}}$. Similarly for $\bar{\mathtt{M}} \backslash \bar{\mathtt{N}}$.

Definition 2.2 (Compiling Reduction)

$$(\mathtt{mixin\ C = D\ with\ R}) \diamond (\mathtt{class\ E\ R'}) \rightsquigarrow \mathtt{class\ C\ (R' \bullet R)}$$

$$\frac{\bar{\mathtt{E}}\ \bar{\mathtt{h}} \equiv (\bar{\mathtt{C}}\ \bar{\mathtt{f}} \backslash \bar{\mathtt{D}}\ \bar{\mathtt{g}})\ \bar{\mathtt{D}}\ \bar{\mathtt{g}} \qquad \bar{\mathtt{K}} \equiv (\bar{\mathtt{M}} \backslash \bar{\mathtt{N}})\ \bar{\mathtt{N}}}{\{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \bar{\mathtt{M}}\} \bullet \{\bar{\mathtt{D}}\ \bar{\mathtt{g}};\ \bar{\mathtt{N}}\} \rightsquigarrow \{\bar{\mathtt{E}}\ \bar{\mathtt{h}};\ \bar{\mathtt{K}}\}} \qquad \frac{\mathtt{R} \rightsquigarrow \mathtt{R'}}{\mathtt{class\ C\ R} \rightsquigarrow \mathtt{class\ C\ R'}}$$

Since fields and methods can be overridden regardless to their types, compilation of a mixin term might lead to a non well typed class, at least according to typing systems like for FJ (without casting). To enforce such a property we introduce an extension of FJ type system by class types resembling object types from [1], and adopting structural subtyping.

Definition 2.3 (Types and Subtyping) Types are defined by the grammar:

$$\begin{array}{rcll} A,B & ::= & \mathtt{C} & \text{(class name)} \\ & | & [\bar{\mathtt{f}} : \bar{A};\ \bar{\mathtt{m}} : \bar{F}] & \text{(class type)} \end{array} \qquad \begin{array}{rcll} F,G & ::= & \bar{A} \to A & \text{(functional type)} \end{array}$$

We do not have the `Object` type, which is considered as abbreviation of $[\,;\,]$.
Let $\Theta = \{\mathtt{C}_1 \mapsto A_1, \ldots, \mathtt{C}_n \mapsto A_n\}$ be a finite map from class names to class types; then we deduce judgments $\Theta \vdash A <: B$ by the rules:

$$\frac{}{\Theta \vdash A <: A}\,(s\text{-}id) \qquad \frac{\Theta \vdash A <: B \qquad \Theta \vdash B <: C}{\Theta \vdash A <: C}\,(s\text{-}trans) \qquad \frac{\Theta(\mathtt{C}) = A}{\Theta \vdash \mathtt{C} <: A, \quad A <: \mathtt{C}}\,(s\text{-}map)$$

$$\frac{}{\Theta \vdash [\bar{\mathtt{f}} : \bar{A}, \bar{\mathtt{g}} : \bar{B};\ \bar{\mathtt{m}} : \bar{F}, \bar{\mathtt{n}} : \bar{G}] <: [\bar{\mathtt{f}} : \bar{A};\ \bar{\mathtt{m}} : \bar{F}]}\,(s\text{-}record_1)$$

$$\frac{\Theta \vdash \bar{A} <: \bar{B} \qquad \Theta \vdash \bar{F} <: \bar{G}}{\Theta \vdash [\bar{\mathtt{f}} : \bar{A};\ \bar{\mathtt{m}} : \bar{F}] <: [\bar{\mathtt{f}} : \bar{B};\ \bar{\mathtt{m}} : \bar{G}]}\,(s\text{-}record_2) \qquad \frac{\Theta \vdash \bar{A}' <: \bar{A} \qquad \Theta \vdash B <: B'}{\Theta \vdash \bar{A} \to B <: \bar{A}' \to B'}\,(s\text{-}func)$$

We observe that, as it happens with FJ, nothing prevents the class name `C` from occurring in the type $\Theta(\mathtt{C})$, so that class types are implicitly recursive; in fact in the subtyping system the name `C` is identified with $\Theta(\mathtt{C})$ up to the $<:$ relation.

Let us introduce the abbreviation

$$[\bar{\mathtt{f}} : \bar{A};\ \bar{\mathtt{m}} : \bar{F}] \bullet [\bar{\mathtt{g}} : \bar{B};\ \bar{\mathtt{n}} : \bar{G}] \equiv [\bar{\mathtt{h}} : \bar{C};\ \bar{\mathtt{p}} : \bar{H}]$$

where $\bar{\mathtt{h}} : \bar{C} = (\bar{\mathtt{f}} : \bar{A} \setminus \bar{\mathtt{g}} : \bar{B})\,\bar{\mathtt{g}} : \bar{B}$ and $\bar{\mathtt{p}} : \bar{H} = (\bar{\mathtt{m}} : \bar{F} \setminus \bar{\mathtt{n}} : \bar{G})\,\bar{\mathtt{n}} : \bar{G}$.

Definition 2.4 (Mixin Typing System) Let $\Pi = \{\mathtt{f}_1 : \mathtt{C}_1, \ldots, \mathtt{f}_h : \mathtt{C}_h, \mathtt{m}_1 : F_1, \ldots, \mathtt{m}_k : F_k\}$, E be either a term `T` or a record `R`, and U, V be either class types A, B or functional types F, G.

$$\frac{\Pi(\mathtt{f}) = \mathtt{C}}{\Theta;\Pi \vdash (\mathtt{C}\ \mathtt{f}) : \mathtt{C}}\,(t\text{-}field) \qquad \frac{\Pi(\mathtt{m}) = \bar{\mathtt{D}} \to \mathtt{C}}{\Theta;\Pi \vdash \mathtt{C}\ \mathtt{m}(\bar{\mathtt{D}}\ \bar{x})\{e\} : \bar{\mathtt{D}} \to \mathtt{C}}\,(t\text{-}method)$$

$$\frac{\Theta;\Pi \vdash (\bar{\mathtt{C}}\ \bar{\mathtt{f}}) : \bar{A} \qquad \Theta;\Pi \vdash \bar{\mathtt{M}} : \bar{F}}{\Theta;\Pi \vdash \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \bar{\mathtt{M}}\} : [\bar{\mathtt{f}} : \bar{A};\ \bar{\mathtt{m}} : \bar{F}]}\,(t\text{-}body) \qquad \frac{\Theta;\Pi \vdash \mathtt{R} : A \qquad \Theta;\Pi \vdash \mathtt{R}' : B}{\Theta;\Pi \vdash \mathtt{R} \bullet \mathtt{R}' : A \bullet B}\,(t\text{-}merge)$$

$$\frac{\Theta;\Pi \vdash \mathtt{R} : A \qquad \Theta(\mathtt{C}) = A}{\Theta;\Pi \vdash \mathtt{class}\ \mathtt{C}\ \mathtt{R} : \mathtt{C}}\,(t\text{-}class) \qquad \frac{\Theta;\Pi \vdash E : U \qquad \Theta \vdash U <: V}{\Theta;\Pi \vdash E : V}\,(t\text{-}sub)$$

$$\frac{\Theta;\Pi \vdash \mathtt{R} : B \qquad \Theta \vdash \mathtt{D} \bullet B <: \mathtt{C}}{\Theta;\Pi \vdash \mathtt{mixin}\ \mathtt{C} = \mathtt{D}\ \mathtt{with}\ \mathtt{R} : \mathtt{D} \to \mathtt{C}}\,(t\text{-}mixin) \qquad \frac{\Theta;\Pi \vdash \mathtt{T} : A \to B \qquad \Theta;\Pi \vdash \mathtt{T}' : A}{\Theta;\Pi \vdash \mathtt{T} \diamond \mathtt{T}' : B}\,(t\text{-}app)$$

To asses soundness of compilation of typed mixin terms, we adapt to the present case the familiar subject reduction property.

Theorem 2.5 (Subject Reduction for Compiling) *If* $\Theta;\Pi \vdash E : U$ *and* $E \rightsquigarrow^* E'$ *then* $\Theta;\Pi \vdash E' : U$.

3 A lambda calculus of records

We consider here a variant of the calculus of records used in [9] and of its intersection type assignment system. With respect to the original calculus we do not treat the update/extension symbol := as an operator, but as a term constructor, and formalize the field selection in a record by the reduction rules

that correspond to well known field/method lookup functions. Further we add the record merge operator $\oplus$ from [8]. Although this is not necessary for the purposes of this work, we feel it worthy to show that both the λ-calculus and the type system can accomodate such an operator, that is quite common in foundational calculi dealing with inheritance.

The syntax of Λ_R is defined by the following grammar:

$$\begin{array}{rcll} M,N & ::= & x \mid \lambda x.M \mid (M)N \mid R \mid R.a & \text{(term)} \\ S & ::= & \langle\rangle \mid S.a := M & \\ R & ::= & x \mid \langle\rangle \mid R.a := M \mid R \oplus S & \text{(record)} \end{array}$$

where $x \in \mathbf{Var}$ and $a \in \mathbf{Label}$ range over denumerably many variables and labels respectively. Here S is an auxiliary syntactical category in the definition of records R. Clearly any S is also a record, but not of the shape $(\cdots(x.a_1 := M_1)\cdots).a_k := M_k$ for any $M_1,\ldots,M_k$, so in particular S cannot be a variable. This restriction is used in the definition of records to restrict the right hand side subterm S in $R \oplus S$. The reason for introducing such a restriction is discussed in Remark 3.4 below.

Definition 3.1 (Λ_R Reduction)

$$\begin{array}{lll} (\beta) & (\lambda x.M)N \longrightarrow M\{N/x\} & \\ (r_1) & (R.a := N).a \longrightarrow N & \\ (r_2) & (R.a := N).b \longrightarrow R.b & \text{if } a \not\equiv b \\ (\oplus_1) & R \oplus \langle\rangle \longrightarrow R & \\ (\oplus_2) & R \oplus (S.a := M) \longrightarrow (R \oplus S).a := M & \end{array}$$

By definition a record R is a term of the shape:

$$R \equiv r.a_1 := M_1 \cdots a_k := M_k \;\equiv\; (\cdots(r.a_1 := M_1)\cdots).a_k := M_k$$

where r is either a variable or the empty record $\langle\rangle$. The set of labels of R is $lbl(R) = \{a_1,\ldots,a_k\}$. More precisely:

$$lbl(x) = lbl(\langle\rangle) = \varnothing,\;\; lbl(R.a := M) = \{a\} \cup lbl(R),\;\; lbl(R \oplus R') = lbl(R) \cup lbl(R').$$

If the labels of R are pairwise distinct, we abbreviate R by $\langle a_i = M_i \;\; i \in I\rangle$ where $\{a_i \mid i \in I\} = lbl(R)$. Then we have:

$$\begin{array}{rrcll} (rsel) & \langle a_i = M_i \;\; i \in I\rangle.a_j & \xrightarrow{*} & M_j & \text{if } j \in I \\ (rupd) & \langle a_i = M_i \;\; i \in I\rangle.a_j := N & \xrightarrow{*} & \langle a_i = M_i \;\; i \in I\backslash\{j\}, a_j = N\rangle & \\ (\oplus_3) & \langle a_i = M_i \;\; i \in I\rangle \oplus \langle a_j = N_j \;\; j \in J\rangle & \xrightarrow{*} & \langle a_i = M_i \; i \in I\backslash J, \;\; a_j = N_j \; j \in J\rangle & \end{array}$$

The last reduction makes it apparent that $\oplus$ models record merge in [7], which is written $\leftarrow_r$, and the analogous operator in [8], although in our notation $R \oplus S$ the record arguments are listed in the opposite order than in [8], since the fields in S prevail over those fields in R having the same labels.

3.1 Intersection types for Λ_R

The type system below is an extension of that one in [4] to the calculus with records. Type syntax is the following (see [9]):

$$\sigma,\tau ::= \alpha \mid \omega \mid \sigma \to \tau \mid \sigma \wedge \tau \mid \langle a : \sigma \rangle.$$

Here α ranges over a countable set of type variables; ω is the universal type; $\sigma \to \tau$ and $\sigma \wedge \tau$ are the arrow and the intersection types respectively. Finally $\langle a : \sigma \rangle$ is the type of records having a label a with value of type σ.

Definition 3.2 (Type Inclusion) Over types we consider the preorder $\leqslant$ which is that one in [4] plus:

1. $\sigma \leqslant \tau \Rightarrow \langle a : \sigma \rangle \leqslant \langle a : \tau \rangle$;
2. $\langle a : \sigma \rangle \wedge \langle a : \tau \rangle \leqslant \langle a : \sigma \wedge \tau \rangle$.

Note that we have

$$\langle a : \sigma \rangle \wedge \langle a : \tau \rangle = \langle a : \sigma \wedge \tau \rangle$$

where $=$ is the symmetric closure of $\leqslant$. Also we have $\langle a : \omega \rangle \neq \omega$ for any a. In concrete contexts one could consider adding axioms involving type constants, like $\mathsf{int} \leqslant \mathsf{real}$ or $\mathsf{even} \leqslant \mathsf{int}$, or class names $\mathtt{C},\mathtt{D},\ldots$

Definition 3.3 (Type Assignment) The assignment system adds to the rules in [4] (but with the $\leqslant$ relation from Def. 3.2) the following ones:

$$\frac{\Gamma \vdash R : \langle a : \sigma \rangle}{\Gamma \vdash R.a : \sigma}\,(sel)$$

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash R.a := M : \langle a : \sigma \rangle}\,(upd_1) \qquad \frac{\Gamma \vdash R : \langle a : \sigma \rangle \quad b \not\equiv a}{\Gamma \vdash R.b := M : \langle a : \sigma \rangle}\,(upd_2)$$

$$\frac{\Gamma \vdash S : \langle a : \sigma \rangle}{\Gamma \vdash R \oplus S : \langle a : \sigma \rangle}\,(\oplus_r) \qquad \frac{\Gamma \vdash R : \langle a : \sigma \rangle \qquad a \notin lbl(S)}{\Gamma \vdash R \oplus S : \langle a : \sigma \rangle}\,(\oplus_l)$$

Rules (upd_1) and $(\oplus_r)$ on one hand, and (upd_2) and $(\oplus_l)$ on the other clearly correspond each other. In particular the side condition $a \notin lbl(R')$ in $(\oplus_l)$ is needed for the same reason of the side condition $b \not\equiv a$ in (upd_2), that would be revealed by the loose of subject reduction property.

Remark 3.4 If we admit record terms of the shape $R \oplus x$ then their typing would be highly problematic. Suppose for simplicity that $x \notin (R)$. Since $lbl(x) = \varnothing$ the side condition $a \notin \varnothing$ of rule $(\oplus_l)$ is always satisfied, and we could derive for $R \oplus x$ all the types of R; hence for any σ:

$$\frac{\dfrac{\Gamma, x : \sigma \vdash R : \langle a : \tau \rangle \qquad a \notin lbl(x) = \varnothing}{\Gamma, x : \sigma \vdash R \oplus x : \langle a : \tau \rangle}}{\Gamma \vdash \lambda x(R \oplus x) : \sigma \to \langle a : \tau \rangle}$$

Let R' be a record such that $a \in lbl(R')$, $\Gamma \vdash R' : \sigma$ but $\Gamma \nvdash R' : \langle a : \tau \rangle$: then $\Gamma \vdash (\lambda x(R \oplus x))R' : \langle a : \tau \rangle$, while $\Gamma \nvdash R \oplus R' : \langle a : \tau \rangle$, contradicting subject reduction.

One could think to change the definition of $lbl(x)$ to $lbl'(x) = \mathbf{Label}$, since the variable x can be replaced by β-reduction with any record R', whose set of labels we cannot predict. This time the side

condition $a \notin \mathbf{Label}$ never holds true, so that $(\lambda x.R \oplus x)R'$ would have less types than $R \oplus R'$, breaking subject expansion.

Similar remarks apply to the case $R \oplus (x.a := M)$, and in general to any merge term $R \oplus R'$ where R' is not of the restricted shape S in the definition of Λ_R.

Let us abbreviate:

$$\langle a_i : \sigma_i \; i \in I \rangle \equiv \bigwedge_{i \in I} \langle a_i : \sigma_i \rangle$$

where we assume the a_i to be pairwise distinct. Then the following rule is admissible:

$$\frac{\sigma_i \leqslant \tau_i \qquad \forall j \in J \subseteq I}{\langle a_i : \sigma_i \; i \in I \rangle \leqslant \langle a_j : \tau_j \; j \in J \rangle}$$

which is record subtyping in width and depth. Further we have the following admissible typing rules:

$$\frac{\Gamma \vdash N_i : \sigma_i \qquad \forall i \in I \subseteq J}{\Gamma \vdash \langle a_i = N_i \;\; i \in J \rangle : \langle a_i : \sigma_i \; i \in I \rangle} \; (\textit{ta-record})$$

$$\frac{\Gamma \vdash \langle a_i = N_i \;\; i \in I \rangle : \langle a_j : \sigma_j \; j \in J \rangle \qquad I \supseteq J \qquad \Gamma \vdash N : \tau}{\Gamma \vdash \langle a_i = N_i \;\; i \in I \rangle . a_k := N : \langle a_j : \sigma_j \; j \in J \backslash \{k\}, a_k : \tau \rangle} \; (\textit{ta-upd})$$

$$\frac{\Gamma \vdash R : \langle a_i : \sigma_i \; i \in I \rangle \qquad \Gamma \vdash S : \langle a_j : \tau_j \; j \in J \rangle}{\Gamma \vdash R \oplus S : \langle a_i : \sigma_i \;\; i \in I \backslash J, \; a_j : \tau_j \;\; j \in J \rangle} \; (\textit{ta-merge})$$

Let $M = N$ be the convertibility relation generated by $\longrightarrow$; then we can prove:

Theorem 3.5 (Type Invariance for Λ_R)

$$\Gamma \vdash M : \sigma \;\&\; M = N \;\; \Rightarrow \;\; \Gamma \vdash N : \sigma.$$

4 Mixin logical semantics

Let $R \equiv \langle a_i = M_i \;\; i \in I \rangle$. The *difference* operator Δ_R in [8] can be written in the form:

$$\Delta_R \equiv \lambda z.z \oplus \langle a_i = M_i \;\; i \in I \rangle.$$

In case the variable $z \notin fv(M_i)$ for all $i \in I$ we say that Δ_R is *simple*; simple difference operators cannot handle super calls, which are not considered here, however.

Although the $\oplus$ operator is not definable from the other operators in Λ_R, this is not the case for Δ_R, either simple or not. In fact we have $\Delta_R \xrightarrow{*} \lambda z.(z.a_1 := M 1 \cdots a_k := M_k)$ if $R \equiv \langle\rangle.a_1 := M 1 \cdots a_k := M_k$.

Definition 4.1 (Translation of Mixin Terms Into Λ_R) Let $\langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}}$ be the partially defined map interpreting mixin records into Λ_R:

$$\begin{aligned} \langle\!\langle \{\bar{\mathtt{C}} \; \bar{\mathtt{f}}; \; \bar{\mathtt{M}}\} \rangle\!\rangle_{\bar{x},\bar{y}} &= \langle \bar{\mathtt{f}} = \bar{x}, \bar{\mathtt{m}} = \bar{y} \rangle \\ \langle\!\langle \mathtt{R} \bullet \mathtt{R}' \rangle\!\rangle_{\bar{x},\bar{y}} &= \langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}} \oplus \langle\!\langle \mathtt{R}' \rangle\!\rangle_{\bar{y}} \quad (\bar{x} \cap \bar{y} = \varnothing) \end{aligned}$$

Then we extend $\langle\!\langle \cdot \rangle\!\rangle_{\bar{x}}$ to mixin terms $\langle\!\langle \mathtt{T} \rangle\!\rangle_{\bar{x}}$ as follows:

$$\begin{array}{rcll} \langle\!\langle \mathtt{class\ C\ R} \rangle\!\rangle_{\bar{x}} & = & \langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}} & \\ \langle\!\langle \mathtt{mixin\ C = D\ with\ R} \rangle\!\rangle_{\bar{x}} & = & \lambda z.z \oplus \langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}} & (z \notin \bar{x}) \\ \langle\!\langle \mathtt{T} \diamond \mathtt{T}' \rangle\!\rangle_{\bar{x},\bar{y}} & = & (\langle\!\langle \mathtt{T} \rangle\!\rangle_{\bar{x}})\langle\!\langle \mathtt{T}' \rangle\!\rangle_{\bar{y}} & (\bar{x} \cap \bar{y} = \varnothing) \end{array}$$

where we can write equivalently $\langle\!\langle \mathtt{mixin\ C = D\ with\ R} \rangle\!\rangle_{\bar{x}} = \Delta_{\langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}}}$.

For any R there exists a suitably large vector $\bar{x}$ such that $\langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}}$ is well defined; the same hold of $\langle\!\langle \mathtt{T} \rangle\!\rangle_{\bar{x}}$.

We are now in place to define the "meaning" of a mixin term, in the more abstract setting of logical semantics, where a term denotes the set of its properties, here represented by the intersection types that can be assigned to its translation in Λ_R:

Definition 4.2 (Logical Semantics) Let $\bar{x}$ be such that $\langle\!\langle \mathtt{T} \rangle\!\rangle_{\bar{x}} \in \Lambda_R$ is well defined, then

$$[\![\mathtt{T}]\!] = \{\sigma \mid \exists \Gamma.\ \bar{x} \subseteq dom(\Gamma)\ \&\ \Gamma \vdash \langle\!\langle \mathtt{T} \rangle\!\rangle_{\bar{x}} : \sigma\}.$$

By a tedious but simple check, we can argue that the last definition does not depend on the choice of $\bar{x}$. The next lemma is routine:

Lemma 4.3 If $\bar{x}$ is suitably large, then

1. $\mathtt{R} \rightsquigarrow^* \mathtt{R}' \Rightarrow \langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}} \longrightarrow^* \langle\!\langle \mathtt{R}' \rangle\!\rangle_{\bar{x}}$;
2. $\mathtt{T} \rightsquigarrow^* \mathtt{T}' \Rightarrow \langle\!\langle \mathtt{T} \rangle\!\rangle_{\bar{x}} \longrightarrow^* \langle\!\langle \mathtt{T}' \rangle\!\rangle_{\bar{x}}$.

Now the following theorem immediately follows by 3.5 and 4.3:

Theorem 4.4 (Soundness of Compilation w.r.t. Logical Semantics)

$$\mathtt{T} \rightsquigarrow^* \mathtt{T}' \Rightarrow [\![\mathtt{T}]\!] = [\![\mathtt{T}']\!].$$

5 Further developments and applications

In this paper we have limited our attention to mixins of a very simple kind, namely such that methods in the superclass are just discarded when the class is inherited by a mixin that redefines the bodies attached to their names. The mixin notation could be extended by adding to the method syntax expressions like $e[\mathtt{super}]$ for bodies, to represent the super calls in e, and by redefining the semantics of the merge operator by:

$$\frac{\bar{\mathtt{E}}\ \bar{\mathtt{h}} \equiv (\bar{\mathtt{C}}\ \bar{\mathtt{f}} \backslash \bar{\mathtt{D}}\ \bar{\mathtt{g}})\ \bar{\mathtt{D}}\ \bar{\mathtt{g}} \qquad \bar{\mathtt{K}} \equiv (\bar{\mathtt{M}} \backslash \bar{\mathtt{N}})\ \bar{\mathtt{N}}[\{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \bar{\mathtt{M}}\}]}{\{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \bar{\mathtt{M}}\} \bullet \{\bar{\mathtt{D}}\ \bar{\mathtt{g}};\ \bar{\mathtt{N}}\} \rightsquigarrow \{\bar{\mathtt{E}}\ \bar{\mathtt{h}};\ \bar{\mathtt{K}}\}}$$

where $\bar{\mathtt{N}}[\{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \bar{\mathtt{M}}\}]$ represents the substitution of the (record defining the) superclass to the variable super in the bodies $e[\mathtt{super}]$ of the methods in $\bar{\mathtt{N}}$.

Although this doesn't seem to affect the type system of mixin terms, which abstracts away from the actual shape of the bodies, it has some impact on the translation into Λ_R terms, that will use unrestricted Δ combinators.

A further work is to integrate the mixin typed calculus with the intersection type system for the interpretation of mixin terms into Λ_R. Following ideas of domain logic, this has been done in [3] in

the case of ς-calculus. First we should associate to each class type A the languages $L(A)$ of intersection types of "kind" A, or more appropriately some language $L_\Theta(A)$ some class name environment Θ, keeping into account that class types are recursive and that the (one step) unfolding a A w.r.t. Θ is the type A^Θ resulting by replacing each class name $\mathtt{C}$ by $\Theta(\mathtt{C})$ in A. Next one should develop a system to derive typings of the shape $\mathtt{T} : A : \sigma$, where $\mathtt{T}$ is a mixin term, A a class or mixin type and σ is in the language of A. The idea is that the types of the mixin calculus play the role of "kinds" of intersection types, that are semantic specifications rather than constraints for sound term construction. The ultimate goal is to construct a type system such that

$$\Theta;\Pi_\Gamma \vdash \mathtt{T} : A : \sigma \Leftrightarrow (\Theta;\Pi \vdash \mathtt{T} : A \ \ \& \ \ \Gamma \vdash \langle\!\langle \mathtt{T} \rangle\!\rangle_{\bar{x}} : \sigma)$$

where $\mathtt{f} : \mathtt{C} : \sigma \in \Pi_\Gamma$ implies $\sigma \in L_\Theta(\mathtt{C})$ and $\Gamma(y) = \sigma$ if $y \in \bar{x}$ corresponds to $\mathtt{f}$ (and similarly for method names). Such a system would have the advantage of assigning intersection types to well typed mixin terms only, which is of interest in the perspective of applying to the mixin calculus the "synthesis by inhabitation method" proposed in [15, 11].

The most interesting application we are figuring of the present system is indeed to program synthesis out of module repositories. This is the subject of ongoing work presented in [5] and based on [12]. The main difference w.r.t. the system with "kinding" above is the extension of the type system of the object language L_0 (e.g. a fragment of Java) by means of constants and operators of the intersection type discipline. Then mixins are introduced as terms of the metalanguage L_1 that are also typed using an extended language with intersection and ω plus a modal operator to include L_0 types.

In spite of these differences, [5] and the present work are complementary, and they could lead to the development of software tools supporting programmers while developing programs out of large libraries of modules.

References

[1] Martín Abadi & Luca Cardelli (1996): *A Theory of Objects*. Springer.

[2] Davide Ancona, Giovanni Lagorio & Elena Zucca (2003): *Jam - designing a Java extension with mixins*. *ACM Trans. Program. Lang. Syst.* 25(5), pp. 641–712. Available at `http://doi.acm.org/10.1145/937563.937567`.

[3] Steffen van Bakel & Ugo de'Liguoro (2008): *Logical Equivalence for Subtyping Object and Recursive Types*. *Theory Comput. Syst.* 42(3), pp. 306–348. Available at `http://dx.doi.org/10.1007/s00224-007-9079-5`.

[4] H. Barendregt, M. Coppo & M. Dezani-Ciancaglini (1983): *A Filter Lambda Model and the Completeness of Type Assignment*. *Journal of Symbolic Logic* 48(4), pp. 931–940.

[5] Jan Bessai, Boris Düser, Andrej Dudenhefer & Moritz Martens (2014): *Delegation-based Mixin Composition Synthesis*. In: *Proc. of ITRS14*, EPTCS.

[6] Viviana Bono, Amit Patel & Vitaly Shmatikov (1999): *A Core Calculus of Classes and Mixins*. In: *ECOOP*, *Lecture Notes in Computer Science* 1628, pp. 43–66. Available at `http://dx.doi.org/10.1007/3-540-48743-3_3`.

[7] Gilad Bracha (1992): *The Programming Language JIGSAW: Mixins, Modularity and Multiple Inheritance*. Ph.D. thesis, Univeristy of Utha.

[8] Gilad Bracha & William R. Cook (1990): *Mixin-based Inheritance*. In: *OOPSLA/ECOOP*, pp. 303–311. Available at `http://doi.acm.org/10.1145/97945.97982`.

[9] Ugo de'Liguoro (2001): *Characterizing Convergent Terms in Object Calculi via Intersection Types*. In: *TLCA*, pp. 315–328. Available at http://dx.doi.org/10.1007/3-540-45413-6_25.

[10] Ugo de'Liguoro (2002): *Subtyping in Logical Form*. *Electr. Notes Theor. Comput. Sci.* 70(1), pp. 72–87. Available at http://dx.doi.org/10.1016/S1571-0661(04)80491-2.

[11] Boris Düdder, Oliver Garbe, Moritz Martens, Jakob Rehof & Paweł Urzyczyn (2012): *Using Inhabitation in Bounded Combinatory Logic with Intersection Types for GUI Synthesis*. In: *Proceedings of ITRS'12*.

[12] Boris Düdder, Moritz Martens & Jakob Rehof (2014): *Staged Composition Synthesis*. In: *ESOP*, *Lecture Notes in Computer Science* 8410, pp. 67–86. Available at http://dx.doi.org/10.1007/978-3-642-54833-8_5.

[13] Atsushi Igarashi, Benjamin C. Pierce & Philip Wadler (2001): *Featherweight Java: a minimal core calculus for Java and GJ*. *ACM Trans. Program. Lang. Syst.* 23(3), pp. 396–450.

[14] Martin Odersky & Matthias Zenger (2005): *Scalable component abstractions*. In: *OOPSLA*, ACM, pp. 41–57. Available at http://doi.acm.org/10.1145/1094811.1094815.

[15] Jakob Rehof (2013): *Towards Combinatory Logic Synthesis*. In: *BEAT'13, 1st International Workshop on Behavioural Types*, ACM.

[16] Reuben N. S. Rowe & Steffen van Bakel (2014): *Semantic Types and Approximation for Featherweight Java*. *Theor. Comput. Sci.* 517, pp. 34–74. Available at http://dx.doi.org/10.1016/j.tcs.2013.08.017.

A Appendix: Proofs

Lemma A.1 (Inversion Lemma for Classes and Mixins)

1. If $\Theta;\Pi \vdash (\mathtt{C}\ \mathtt{f}) : U$, then $U = \mathtt{C}$ and $\Pi(\mathtt{f}) = \mathtt{C}$.
2. If $\Theta;\Pi \vdash \mathtt{C}\ \mathtt{m}(\bar{\mathtt{D}}\ \bar{x})\{\, e \,\} : U$, then $U = \bar{\mathtt{D}} \to \mathtt{C}$ and $\Pi(\mathtt{m}) = \bar{\mathtt{D}} \to \mathtt{C}$.
3. If $\Theta;\Pi \vdash \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \bar{\mathtt{M}}\} : U$, then $U = [\bar{\mathtt{f}} : \bar{A};\ \bar{\mathtt{m}} : \bar{F}]$ for some $\bar{A}$ and $\bar{F}$ and $\Theta;\Pi \vdash \bar{\mathtt{C}}\ \bar{\mathtt{f}} : \bar{A}$ and $\Theta;\Pi \vdash \bar{\mathtt{M}} : \bar{F}$.
4. If $\Theta;\Pi \vdash \mathtt{R} \bullet \mathtt{R}' : U$, then $U = A \bullet B$ and $\Theta;\Pi \vdash \mathtt{R} : A$ and $\Theta;\Pi \vdash \mathtt{R}' : B$.
5. If $\Theta;\Pi \vdash \mathtt{class}\ \mathtt{C}\ \mathtt{R} : U$, then $U = \mathtt{C}$ and there exists A such that $\Theta;\Pi \vdash \mathtt{R} : A$ and $\Theta(\mathtt{C}) = A$.
6. If $\Theta;\Pi \vdash E : U$, then $\Theta;\Pi \vdash E : V$ and $\Theta \vdash V <: U$.
7. If $\Theta;\Pi \vdash \mathtt{mixin}\ \mathtt{C} = \mathtt{D}\ \mathtt{with}\ \mathtt{R} : U$, then $U = \mathtt{D} \to \mathtt{C}$ and there exists B such that $\Theta;\Pi \vdash \mathtt{R} : B$ and $\Theta \vdash \mathtt{D} \bullet B <: \mathtt{C}$.
8. If $\Theta;\Pi \vdash \mathtt{T} \diamond \mathtt{T}' : U$, then U is a class type, say $U = B$, and $\Theta;\Pi \vdash \mathtt{T} : A \to B$ and $\Theta;\Pi \vdash \mathtt{T}' : A$.

Proof. By Definition 2.4 for $\Theta;\Pi \vdash E : U$. □

Lemma A.2 If $\Theta \vdash A_1 <: A_2$, then $\Theta \vdash A_1 \bullet B <: A_2 \bullet B$.

Proof. The operation $\bullet$ is only applied when A_1, A_2, and B are class types. Withtout loss of generality, assume $A_1 = [\bar{\mathtt{f}} : \bar{A_1'};\ \bar{\mathtt{m}} : \bar{F}]$, $A_2 = [\bar{\mathtt{f}'} : \bar{A_2'};\ \bar{\mathtt{m}'} : \bar{F'}]$ and $B = [\bar{\mathtt{g}} : \bar{B'};\ \bar{\mathtt{n}} : \bar{G}]$. Since $\Theta \vdash A_1 <: A_2$, it is either,

1. by rule $(s\text{-}record_1)$, $\bar{\mathtt{f}'} \subset \bar{\mathtt{f}}$ and $\bar{\mathtt{m}'} \subset \bar{\mathtt{m}}$, or
2. by rule $(s\text{-}record_2)$, $\bar{\mathtt{f}'} = \bar{\mathtt{f}}$ and $\bar{\mathtt{m}'} = \bar{\mathtt{m}}$, which imply that $\bar{A_1'} <: \bar{A_2'}$ and $\bar{F} <: \bar{F'}$,

for either case,

$$A_1 \bullet B = [(\bar{\mathtt{f}} : \bar{A_1'} \setminus \bar{\mathtt{g}} : \bar{B})\,\bar{\mathtt{g}} : \bar{B};\ (\bar{\mathtt{m}} : \bar{F} \setminus \bar{\mathtt{n}} : \bar{G})\,\bar{\mathtt{n}} : \bar{G}] <: [(\bar{\mathtt{f}'} : \bar{A_2'} \setminus \bar{\mathtt{g}} : \bar{B})\,\bar{\mathtt{g}} : \bar{B};\ (\bar{\mathtt{m}'} : \bar{F'} \setminus \bar{\mathtt{n}} : \bar{G})\,\bar{\mathtt{n}} : \bar{G}] = A_2 \bullet B$$

□

Theorem 2.5 Subject Reduction for Compiling If $\Theta;\Pi \vdash E : U$ and $E \rightsquigarrow^* E'$ then $\Theta;\Pi \vdash E' : U$.

Proof. We prove it by showing all compiling rules of E.

- $E = \mathtt{T} \diamond \mathtt{T}'$, where $\mathtt{T} = \mathtt{mixin\ C = D\ with\ R}$ and $\mathtt{T}' = \mathtt{class\ E\ R}'$. Lemma A.1.8 says U is a class type, say $U = B$, and there exists A such that

$$\Theta;\Pi \vdash \mathtt{T} : A \to B \tag{1}$$

and

$$\Theta;\Pi \vdash \mathtt{T}' : A \tag{2}$$

By applying Lemma A.1.7 to Eq. (1), we have $A = \mathtt{D}$ and $B = \mathtt{C}$ and there exists A'' such that $\Theta;\Pi \vdash \mathtt{R} : A''$ and

$$\Theta \vdash A \bullet A'' <: B \tag{3}$$

By applying Lemma A.1.5 to Eq. (2), we have $A = \mathtt{E}$ and there exists A' such that $\Theta;\Pi \vdash \mathtt{R}' : A'$ and $\Theta(\mathtt{E}) = A'$. According to rule (*s-map*),

$$A' <: \mathtt{E} = A \tag{4}$$

By applying rule (*t-merge*) to $\Theta;\Pi \vdash \mathtt{R} : A''$ and $\Theta;\Pi \vdash \mathtt{R}' : A'$, we have $\Theta;\Pi \vdash \mathtt{R}' \bullet \mathtt{R} : A' \bullet A''$. By applying Lemma A.2 and Eq. (3) and Eq. (4), we have

$$\Theta \vdash A' \bullet A'' <: \mathtt{E} \bullet A'' = A \bullet A'' <: B. \tag{5}$$

Thus by applying rule (*t-sub*) to Eq. (5), we derive

$$\Theta;\Pi \vdash \mathtt{R}' \bullet \mathtt{R} : B \tag{6}$$

By applying rule (*t-class*) to $\Theta;\Pi \vdash \mathtt{class\ C} : \mathtt{C} = B$, and Eq. (6), we derive

$$\Theta;\Pi \vdash \mathtt{class\ C}(\mathtt{R}' \bullet \mathtt{R}) : B.$$

- $E = \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \bar{\mathtt{M}}\} \bullet \{\bar{\mathtt{D}}\ \bar{\mathtt{g}};\ \bar{\mathtt{N}}\}$ and $\bar{\mathtt{E}}\ \bar{\mathtt{h}} \equiv (\bar{\mathtt{C}}\ \bar{\mathtt{f}} \backslash \bar{\mathtt{D}}\ \bar{\mathtt{g}})\ \bar{\mathtt{D}}\ \bar{\mathtt{g}}$ and $\bar{\mathtt{K}} \equiv (\bar{\mathtt{M}} \backslash \bar{\mathtt{N}})\ \bar{\mathtt{N}}$. Lemma A.1.4 says U is a class type, take $U = B$, and

$$\Theta;\Pi \vdash \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \bar{\mathtt{M}}\} : A_1 \text{ and } \Theta;\Pi \vdash \{\bar{\mathtt{D}}\ \bar{\mathtt{g}};\ \bar{\mathtt{N}}\} : A_2 \tag{7}$$

and $B = A_1 \bullet A_2$. According to Lemma A.1.3, take $A_1 = [\bar{\mathtt{f}} : \bar{A_1'};\ \bar{\mathtt{m}} : \bar{F}]$ for some $\bar{A_1'}$ and $\bar{F}$, then we have $\Theta;\Pi \vdash (\bar{\mathtt{C}}\ \bar{\mathtt{f}}) : \bar{A_1'}$ and $\Theta;\Pi \vdash \bar{\mathtt{M}} : \bar{F}$. Similarly, take $A_2 = [\bar{\mathtt{g}} : \bar{A_2'};\ \bar{\mathtt{n}} : \bar{G}]$ for some $\bar{A_2'}$ and $\bar{G}$, then we have $\Theta;\Pi \vdash (\bar{\mathtt{D}}\ \bar{\mathtt{g}}) : \bar{A_2'}$ and $\Theta;\Pi \vdash \bar{\mathtt{N}} : \bar{G}$. By definition of merge operation,

$$B = A \bullet A' = [(\bar{\mathtt{f}} : \bar{A_1'} \backslash \bar{\mathtt{g}} : \bar{A_2'})\bar{\mathtt{g}} : \bar{A_2'};\ (\bar{\mathtt{m}} : \bar{F} \backslash \bar{\mathtt{n}} : \bar{G})\bar{\mathtt{n}} : \bar{G}].$$

By applying rule (*t-merge*) to Eq. (7), we have

$$\Theta;\Pi \vdash \{(\bar{\mathtt{C}}\ \bar{\mathtt{f}} \backslash \bar{\mathtt{D}}\ \bar{\mathtt{g}})\ \bar{\mathtt{D}}\ \bar{\mathtt{g}}\ ;\ (\bar{\mathtt{M}} \backslash \bar{\mathtt{N}})\ \bar{\mathtt{N}} : B\},$$

thus

$$\Theta;\Pi \vdash \{\bar{\mathtt{E}}\ \bar{\mathtt{h}};\ \bar{\mathtt{K}}\} : B.$$

- $E = \texttt{class C R}$ and $\texttt{R} \rightsquigarrow \texttt{R}'$. Lemma A.1.5 says $U = \texttt{C}$ and there exists A such that $\Theta;\Pi \vdash \texttt{R} : A$ and $\Theta(\texttt{C}) = A$. By induction hypothesis on derivation of $\rightsquigarrow$, we have $\Theta;\Pi \vdash \texttt{R}' : A$. By applying rule (*t-class*) to $\Theta;\Pi \vdash \texttt{R}' : A$ and $\Theta(\texttt{C}) = A$, we have $\Theta;\Pi \vdash \texttt{class C R}' : \texttt{C}$.

□

Lemma A.3 (Generation Lemma) Let $\sigma \not\equiv \omega$.

1. $\Gamma \vdash x : \sigma \iff \Gamma(x) \leqslant \sigma$.
2. $\Gamma \vdash \lambda x.M : \sigma \iff \exists\, I, \sigma_i, \tau_i$ such that $\Gamma, x : \sigma_i \vdash M : \tau_i$ and $\bigwedge_{i\in I} \sigma_i \to \tau_i \leqslant \sigma$.
3. $\Gamma \vdash (M)N : \sigma \iff \exists\, I, \sigma_i, \tau_i$ such that $\Gamma \vdash M : \sigma_i \to \tau_i$ and $\Gamma \vdash N : \sigma_i$ and $\bigwedge_{i\in I} \tau_i \leqslant \sigma$.
4. $\Gamma \vdash \langle\rangle : \tau \iff \tau \equiv \omega$.
5. $\Gamma \vdash R.a : \sigma \iff \exists\, I, \tau_i$ such that $\Gamma \vdash R : \langle a : \tau_i, i \in I\rangle$ and $\bigwedge_{i\in I} \tau_i \leqslant \sigma$.
6. $\Gamma \vdash R.a := M : \sigma \iff (\exists\, \tau$ such that $\Gamma \vdash M : \tau$ and $\langle a : \tau\rangle \leqslant \sigma) \vee (\exists\, I, b_i, \tau_i$ such that $\Gamma \vdash R : \langle b_i : \tau_i, i \in I\rangle$ and $\langle b_i : \tau_i, i \in I\rangle \leqslant \sigma$ and $a \notin \{b_i, i \in I\})$.
7. $\Gamma \vdash R \oplus S : \sigma \iff (\Gamma \vdash S : \sigma) \vee (\exists\, I, a_i, \sigma_i$ such that $\Gamma \vdash R : \langle a_i : \sigma_i, i \in I\rangle$ and $\{a_i, i \in I\} \cap lbl(S) = \varnothing$ and $\langle a_i : \sigma_i, i \in I\rangle \leqslant \sigma)$.

Proof. Since 1, 2, and 3 are standard, here we only show the proofs for cases of 4, 5, 6, and 7.

- Case 4: For $\Rightarrow$, if $\Gamma \vdash \langle\rangle : \tau$ and $\tau \not\equiv \omega$ but $\tau = \langle a : \sigma\rangle$, then by rule (*sel*) we have $\Gamma \vdash \langle\rangle.a : \sigma$. But $\langle\rangle.a = \langle\rangle$, we thus have $\sigma = \langle a : \sigma\rangle = \tau$, which means we should have $\tau = \omega$ otherwise it is a conflict.

  For $\Leftarrow$, it is trivial.

- Case 5: For $\Rightarrow$, if $\Gamma \vdash R.a : \sigma$, let $\Gamma \vdash R : \tau$. If $\tau \equiv \omega$, then rule (*sel*) is not applicable. If $\tau \not\equiv \omega$, by Definition 3.3, without loss of generality, let $\tau = \bigwedge_i \langle a' : \tau_i\rangle$ for some $i \in I$. Rule (*sel*) is only applicable when $a' = a$ and $\bigwedge_{i\in I} \tau_i \leqslant \sigma$. Thus we derive that $\exists\, I, \tau_i$ such that $\Gamma \vdash R : \langle a : \tau_i, i \in I\rangle$ and $\bigwedge_{i\in I} \tau_i \leqslant \sigma$.

  For $\Leftarrow$, it is straightforward by applying Definition 3.2 to $\bigwedge_{i\in I} \tau_i \leqslant \sigma$ for some I and τ_i, we have $\bigwedge_{i\in I} \langle a : \tau_i\rangle \leqslant \langle a : \sigma\rangle$ and $\Gamma \vdash R : \langle a : \tau_i, i \in I\rangle \leqslant \langle a : \sigma\rangle$, thus we derive $\Gamma \vdash R : \langle a : \sigma\rangle$. By applying rule (*sel*) to $\Gamma \vdash R : \langle a : \sigma\rangle$, we have $\Gamma \vdash R.a : \sigma$.

- Case 6: For $\Rightarrow$, if $\Gamma \vdash (R.a := M) : \sigma$, let $R \equiv r.a_1 := M_1 ... a := M ... a_k := M_k$ and $\Gamma \vdash R : \tau$ for some τ. Then by Definition 3.3, it is either $\Gamma \vdash M : \tau$ and $\langle a : \tau\rangle \leqslant \sigma$, or for some I we have $\tau = \langle b_i : \tau_i, i \in I\rangle$ where $a \notin \{b_i, i \in I\}$.

  For $\Leftarrow$, if there exists τ such that $\Gamma \vdash M : \tau$, then by rule (upd_1), we have $\Gamma \vdash (R.a := M) : \langle a : \tau\rangle \leqslant \sigma$, thus we have $\Gamma \vdash (R.a := M) : \sigma$; if there exists I, b_i, τ_i such that $\Gamma \vdash R : \langle b_i : \tau_i, i \in I\rangle$ and $a \notin \{b_i, i \in I\}$, then by rule ($upd_2$), we have $\Gamma \vdash (R.a := M) : \langle b_i : \tau_i, i \in I\rangle \leqslant \sigma$, thus we have $\Gamma \vdash (R.a := M) : \sigma$.

- Case 7: For $\Rightarrow$, if $\Gamma \vdash R \oplus S : \sigma$, then by Definition 3.3, it is either $\Gamma \vdash S : \sigma$, or there exists I, a_i, σ_i such that $\Gamma \vdash R : \langle a_i : \sigma_i, i \in I\rangle$ and $\{a_i, i \in I\} \cap lbl(S) = \varnothing$ and $\langle a_i : \sigma_i, i \in I\rangle \leqslant \sigma$.

  For $\Leftarrow$, if $\Gamma \vdash S : \sigma$, then by rule ($\oplus_r$), we have $\Gamma \vdash R \oplus S : \sigma$; if there exists I, a_i, σ_i such that $\Gamma \vdash R : \langle a_i : \sigma_i, i \in I\rangle$ and $\{a_i, i \in I\} \cap lbl(S) = \varnothing$, by rule ($\oplus_l$), we have $\Gamma \vdash R \oplus S : \langle a_i : \sigma_i, i \in I\rangle \leqslant \sigma$, thus we have $\Gamma \vdash R \oplus S : \sigma$.

□

Lemma A.4 (Substitution) $\Gamma, x : \sigma \vdash M : \tau$ and $\Gamma \vdash N : \sigma \iff \Gamma \vdash M\{N/x\} : \tau$.

Proof. We prove it by induction hypothesis on the derivation of $\Gamma, x : \sigma \vdash M : \tau$ and $\Gamma \vdash N : \sigma \iff \Gamma \vdash M\{N/x\} : \tau$. The proofs for cases of $M = x$, $M = y \neq x$, $M = \lambda z.M'$ where $z \notin N$, and $M = (M')N'$ are standard. Here we only show the proofs for other interesting cases.

- Case $M = \langle\rangle$. Trivial.
  - For $\Rightarrow$: Then for some σ and τ, $\Gamma, x : \sigma \vdash \langle\rangle : \tau$ and $\Gamma \vdash N : \sigma$. By Lemma A.3.4, $\tau = \omega$. Since $x \notin \langle\rangle$, $\Gamma \vdash \langle\rangle : \omega$, and thus $\Gamma \vdash M\{N/x\} \equiv \langle\rangle : \omega$.
  - For $\Leftarrow$: Then for some τ, $\Gamma \vdash M\{N/x\} \equiv \langle\rangle : \tau$. By Lemma A.3.4, $\tau = \omega$. Thus it is trivial that $\Gamma, x : \sigma \vdash \langle\rangle : \omega$ and $\Gamma \vdash N : \omega$.
- Case $M \equiv (R.a := M')$.
  - For $\Rightarrow$: Then for some σ and τ, $\Gamma, x : \sigma \vdash (R.a := M') : \tau$ and $\Gamma \vdash N : \sigma$. By Lemma A.3.6, one of the cases below should hold:
    - $\Gamma, x : \sigma \vdash M' : \tau'$ for some τ' and $\langle a : \tau' \rangle \leqslant \tau$. By induction, we have $\Gamma \vdash M'\{N/x\} : \tau'$. By rule ($upd_1$), we have $\Gamma \vdash (R\{N/x\}.a := M'\{N/x\}) : \langle a, \tau' \rangle \leqslant \tau$. Thus we derive $\Gamma \vdash M\{N/x\} : \tau$.
    - $\Gamma, x : \sigma \vdash R : \langle b_i : \tau_i, i \in I \rangle$ for some I, b_i, τ_i and $\langle b_i : \tau'_i, i \in I \rangle \leqslant \tau$ and $a \notin \{b_i, i \in I\}$. By induction, we have $\Gamma \vdash R\{N/x\} : \langle b_i : \tau_i, i \in I \rangle$. By rule ($upd_2$), we have $\Gamma \vdash (R\{N/x\}.a := M'\{N/x\}) : \langle b_i : \tau_i, i \in I \rangle \leqslant \tau$. Thus we derive $\Gamma \vdash M\{N/x\} : \tau$.
  - For $\Leftarrow$: Then for some τ, $\Gamma \vdash (R\{N/x\}.a := M'\{N/x\}) : \tau$. By Lemma A.3.6, one of the cases below should hold:
    - $\Gamma \vdash M'\{N/x\} : \tau'$ for some τ' and $\langle a : \tau' \rangle \leqslant \tau$. By induction, $\Gamma, x : \sigma \vdash M' : \tau'$ and $\Gamma \vdash N : \sigma$ for some σ. By rule (upd_1), we have $\Gamma, x : \sigma \vdash (R.a := M') : \langle a : \tau' \rangle \leqslant \tau$, thus we derive $\Gamma, x : \sigma \vdash (R.a := M') : \tau$ and $\Gamma \vdash N : \sigma$.
    - $\Gamma \vdash R\{N/x\} : \langle b_i : \tau_i, i \in I \rangle$ for some I, b_i, τ_i and $\langle b_i : \tau'_i, i \in I \rangle \leqslant \tau$ and $a \notin \{b_i, i \in I\}$. By induction, $\forall i \in I$, there exists some σ'_i such that we have $\Gamma, x : \sigma'_i \vdash R : \langle b_i : \tau_i, i \in I \rangle$ and $\Gamma \vdash N : \sigma'_i$. They imply that $\Gamma, x : \bigwedge_{i \in I} \sigma'_i \vdash R : \bigwedge_{i \in I} \langle b_i : \tau_i, i \in I \rangle \leqslant \tau$ and $\Gamma \vdash N : \bigwedge_{i \in I} \sigma'_i$. By rule ($upd_2$), we have $\Gamma, x : \bigwedge_{i \in I} \sigma'_i \vdash (R.a := M') : \bigwedge_{i \in I} \langle b_i : \tau_i, i \in I \rangle \leqslant \tau$. Thus we derive $\Gamma, x : \sigma \vdash (R.a := M') : \tau$ and $\Gamma \vdash N : \sigma$, where $\sigma = \bigwedge_{i \in I} \sigma'_i$
- Case $M \equiv R \oplus S$ ($S \notin \mathbf{Var}$).
  - For $\Rightarrow$: Then for some σ and τ, $\Gamma, x : \sigma \vdash R \oplus S : \tau$ and $\Gamma \vdash N : \sigma$. By Lemma A.3.7, one of the cases below should hold:
    - $\Gamma, x : \sigma \vdash S : \tau$. Since $S\{N/x\} \equiv S$, by induction, we have $\Gamma \vdash S\{N/x\} : \tau$. By rule ($\oplus_r$), we have $\Gamma \vdash R\{N/x\} \oplus S : \tau$. Thus we derive $\Gamma \vdash R\{N/x\} \oplus S : \tau$, and derive $\Gamma \vdash M\{N/x\} : \tau$.
    - $\Gamma, x : \sigma \vdash R : \langle a_i : \tau'_i, i \in I \rangle$ for some I, a_i, τ'_i and $\{a_i, i \in I\} \cap lbl(S) = \varnothing$ and $\langle a_i : \tau'_i, i \in I \rangle \leqslant \tau$. By induction, we have $\Gamma \vdash R\{N/x\} : \langle a_i : \tau'_i, i \in I \rangle$. By rule ($\oplus_l$), we have $\Gamma \vdash R\{N/x\} \oplus S : \langle a_i : \tau'_i, i \in I \rangle \leqslant \tau$. Thus we derive $\Gamma \vdash R\{N/x\} \oplus S : \tau$, and derive $\Gamma \vdash M\{N/x\} : \tau$.
  - For $\Leftarrow$: Then for some τ, since $M\{N/x\} \equiv R\{N/x\}$, $\Gamma \vdash R\{N/x\} : \tau$. By Lemma A.3.7, one of the cases below should hold:

* $\Gamma \vdash S : \tau$. Let $\Gamma \vdash N : \sigma$ for some σ. By weakening, we have $\Gamma, x : \sigma \vdash S : \tau$. By rule $(\oplus_r)$, we have $\Gamma, x : \sigma \vdash R \oplus S : \tau$. Thus we derive $\Gamma, x : \sigma \vdash M : \tau$ and $\Gamma \vdash N : \sigma$.
* $\Gamma \vdash R\{N/x\} : \langle a_i : \tau'_i, i \in I \rangle$ for some I, a_i, τ'_i and $\{a_i, i \in I\} \cap lbl(S) = \varnothing$ and $\langle a_i : \tau'_i, i \in I \rangle \leqslant \tau$. By induction, we have $\Gamma, x : \sigma \vdash R : \langle a_i : \tau'_i, i \in I \rangle$ and $\Gamma \vdash N : \sigma$ for some σ. By rule $(\oplus_l)$, we have $\Gamma, x : \sigma \vdash R \oplus S : \langle a_i : \tau'_i, i \in I \rangle \leqslant \tau$. Thus we derive $\Gamma, x : \sigma \vdash M : \tau$ and $\Gamma \vdash N : \sigma$.

□

Theorem 3.5 Type Invariance for Λ_R $\quad \Gamma \vdash M : \sigma \ \& \ M = N \ \Rightarrow \ \Gamma \vdash N : \sigma$.

Proof. We prove it by showing that each case of M satisfies both (I) if $\Gamma \vdash M : \sigma$ and $M \longrightarrow N$, then $\Gamma \vdash N : \sigma$ and (II) if $\Gamma \vdash N : \sigma$ and $M \longrightarrow N$, then $\Gamma \vdash M : \sigma$.

1. $M \equiv R.a$.
   - if $R \equiv (R'.a := N')$, then by rule (r_1) we have $(R'.a := N').a \equiv M \longrightarrow N' \equiv N$.
     - For (I), as $\Gamma \vdash M : \sigma$, by Lemma A.3.5, there exist I, τ_i such that $\Gamma \vdash R : \langle a : \tau_i \rangle$ and $\bigwedge_{i \in I} \tau_i \leqslant \sigma$, thus we have $\Gamma \vdash R : \langle a : \tau_i \rangle \leqslant \langle a : \sigma \rangle$, which implies $\Gamma \vdash R : \langle a : \sigma \rangle$, thus we have $\Gamma \vdash (R'.a := N') : \langle a : \sigma \rangle$. By Lemma A.3.6, we have $\Gamma \vdash N' : \tau$ for some τ and $\langle a : \tau \rangle \leqslant \langle a, \sigma \rangle$. Thus we have $\tau \leqslant \sigma$, and we derive $\Gamma \vdash N : \sigma$.
     - For (II), as $\Gamma \vdash N : \sigma$, which means $\Gamma \vdash N' : \sigma$. By rule (upd_1), we have $\Gamma \vdash (R'.a := N') : \langle a : \sigma \rangle$. Since $R.a \equiv (R'.a := N).a$, by rule (sel), we have $\Gamma \vdash R.a : \sigma$.
   - if $R \equiv (R'.b := N')$ and $b \neq a$, then by rule (r_2) we have $M \equiv (R'.b := N').a \longrightarrow R'.a \equiv N$.
     - For (I), as $\Gamma \vdash M : \sigma$, by Lemma A.3.5, we derive $\Gamma \vdash (R'.b := N') : \langle a : \sigma \rangle$ (hereafter we shorten the proof for deriving $\Gamma \vdash R : \langle a : \sigma \rangle$ since it is as same as the one above). Since $\Gamma \vdash R : \langle a : \sigma \rangle$ and $R \equiv (R'.b := N')$ where $b \neq a$, by Lemma A.3.6, we have case $\Gamma \vdash R' : \langle a : \tau \rangle$ for some $\tau \leqslant \sigma$. Therefore, by rule (sel), $\Gamma \vdash R'.a : \tau \leqslant \sigma$, which implies that $\Gamma \vdash N : \sigma$.
     - For (II), as $\Gamma \vdash N : \sigma$, since $N \equiv R'.a$, by Lemma A.3.5, we derive $\Gamma \vdash R' : \langle a : \sigma \rangle$. By rule (upd_2), we have $\Gamma \vdash (R'.b := N') : \langle a : \sigma \rangle$. Since $M \equiv R.a \equiv (R'.b := N').a$, by rule (sel), we derive $\Gamma \vdash M : \sigma$.

2. $M = R \oplus \langle\rangle$. By rule $(\oplus_1)$, we have $M \equiv R \oplus \langle\rangle \longrightarrow R \equiv N$.
   - For (I), as $\Gamma \vdash M : \sigma$, by Lemma A.3.7, one of the cases below should hold:
     - $\Gamma \vdash \langle\rangle : \sigma$, and by Lemma A.3.4, $\sigma = \omega$, which contradict to the assumption of Lemma A.3 that we derive $\sigma \neq \omega$.
     - $\Gamma \vdash R : \langle a_i : \sigma_i, i \in I \rangle$ for some I, a_i, σ_i and $\{a_i, i \in I\} \cap lbl(\langle\rangle) = \varnothing$ and $\langle a_i : \sigma_i, i \in I \rangle \leqslant \sigma$. Then straightforwardly we have $\Gamma \vdash R : \langle a_i : \sigma_i, i \in I \rangle \leqslant \sigma$, thus $\Gamma \vdash N : \sigma$ as $N = R : \langle a_i : \sigma_i, i \in I \rangle$.
   - For (II), as $\Gamma \vdash N : \sigma$, where $N = R$. By rule $(\oplus_l)$, since $M = R \oplus \langle\rangle$, we have $\Gamma \vdash M : \sigma$.

3. $M \equiv R \oplus (R'.a := M')$. By rules $(\oplus_2)$, if $R' \notin \mathbf{Var}$, we have $M \longrightarrow ((R \oplus R').a := M') \equiv N$.

- For (I), as $\Gamma \vdash M : \sigma$, by Lemma A.3.7, one of the cases below should hold:
  - $\Gamma \vdash (R'.a := M') : \sigma$. By Lemma A.3.6, one of the cases below should hold:
    (a) $\Gamma \vdash M' : \tau$ for some τ and $\langle a : \tau \rangle \leqslant \sigma$. By rule (upd_1), we have $\Gamma \vdash ((R \oplus R').a := M') : \langle a : \tau \rangle \leqslant \sigma$, thus we derive $\Gamma \vdash N : \sigma$ as $N = ((R \oplus R').a := M')$.
    (b) $\Gamma \vdash R : \langle b_i : \tau_i, i \in I \rangle$ for some I, b_i, τ_i and $\langle b_i : \tau_i, i \in I \rangle \leqslant \sigma$ and $a \notin \{b_i, i \in I\}$). By rule (upd_2), we have $\Gamma \vdash ((R \oplus R').a := M') : \langle b_i : \tau_i, i \in I \rangle \leqslant \sigma$, thus we derive $\Gamma \vdash N : \sigma$ as $N = ((R \oplus R').a := M')$.
  - $\Gamma \vdash R : \langle a_i : \sigma_i, i \in I \rangle$ for some I, a_i, σ_i and $\{a_i, i \in I\} \cap lbl(R'.a := M') = \{a_i, i \in I\} \cap (lbl(R') \cup \{a\}) = \varnothing$ and $\langle a_i : \sigma_i, i \in I \rangle \leqslant \sigma$. By rule $(\oplus_l)$, we have $\Gamma \vdash R \oplus R' : \langle a_i : \sigma_i, i \in I \rangle$. By rule (upd_2), we have $\Gamma \vdash ((R \oplus R').a := M') : \langle a_i : \sigma_i, i \in I \rangle \leqslant \sigma$, thus we derive $\Gamma \vdash N : \sigma$ as $N = ((R \oplus R').a := M')$.
- For (II), as $\Gamma \vdash N : \sigma$, where $N \equiv ((R \oplus R').a := M')$. By Lemma A.3.6, one of the cases below should hold:
  - $\Gamma \vdash M' : \tau$ for some τ and $\langle a : \tau \rangle \leqslant \sigma$. By rule (upd_1), we have $\Gamma \vdash (R'.a := M') : \langle a : \tau \rangle$, and by rule $(\oplus_r)$, we have $\Gamma \vdash R \oplus (R'.a := M) : \langle a : \tau \rangle \leqslant \sigma$, thus we derive $\Gamma \vdash M = R \oplus (R'.a := M) : \sigma$.
  - $\Gamma \vdash R \oplus R' : \langle b_i : \tau_i, i \in I \rangle$ for some I, b_i, τ_i and $\langle b_i : \tau_i, i \in I \rangle \leqslant \sigma$ and $a \notin \{b_i, i \in I\}$. By Lemma A.3.7, one of the cases below should hold:
    (a) $\Gamma \vdash R' : \langle b_i : \tau_i, i \in I \rangle$. By rule (upd_2), we have $\Gamma \vdash (R'.a := M') : \langle b_i : \tau_i, i \in I \rangle$, and by rule $(\oplus_r)$, we have $\Gamma \vdash R \oplus (R'.a := M') : \langle b_i : \tau_i, i \in I \rangle \leqslant \sigma$, thus we derive $\Gamma \vdash M = R \oplus (R'.a := M') : \sigma$.
    (b) $\Gamma \vdash R : \langle b_i : \tau_i, i \in I \rangle$ and $\{b_i, i \in I\} \cap lbl(R') = \varnothing$. Since $a \notin \{b_i, i \in I\}$, by rule $(\oplus_l)$, we have $\Gamma \vdash R \oplus (R'.a := M') : \langle b_i : \tau_i, i \in I \rangle \leqslant \sigma$, thus we derive $\Gamma \vdash M = R \oplus (R'.a := M') : \sigma$.

□

Lemma 4.3 If $\bar{x}$ is suitably large, then

1. $\mathtt{R} \rightsquigarrow^* \mathtt{R}' \Rightarrow \langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}} \longrightarrow^* \langle\!\langle \mathtt{R}' \rangle\!\rangle_{\bar{x}}$;
2. $\mathtt{T} \rightsquigarrow^* \mathtt{T}' \Rightarrow \langle\!\langle \mathtt{T} \rangle\!\rangle_{\bar{x}} \longrightarrow^* \langle\!\langle \mathtt{T}' \rangle\!\rangle_{\bar{x}}$.

Proof. It is proved by examining all possible cases.

- $\mathtt{R} = \mathtt{R}_1 \bullet \mathtt{R}_2$ where $\mathtt{R}_1 = \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \bar{\mathtt{M}}\}$ and $\mathtt{R}_2 = \{\bar{\mathtt{D}}\ \bar{\mathtt{g}};\ \bar{\mathtt{N}}\}$. By the rule of compiling reduction, if $\bar{\mathtt{E}}\ \bar{\mathtt{h}} \equiv (\bar{\mathtt{C}}\ \bar{\mathtt{f}} \backslash \bar{\mathtt{D}}\ \bar{\mathtt{g}})\ \bar{\mathtt{D}}\ \bar{\mathtt{g}}$ and $\bar{\mathtt{K}} \equiv (\bar{\mathtt{M}} \backslash \bar{\mathtt{N}})\ \bar{\mathtt{N}}$, then $\mathtt{R} \rightsquigarrow \{\bar{\mathtt{E}}\ \bar{\mathtt{h}};\ \bar{\mathtt{K}}\} = \mathtt{R}'$.

$$
\begin{array}{rcl}
\langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x},\bar{y}} & = & \langle\!\langle \mathtt{R}_1 \bullet \mathtt{R}_2 \rangle\!\rangle_{\bar{x},\bar{y}} \quad \text{where } \bar{x} \cap \bar{y} = \varnothing \\
& = & \langle\!\langle \mathtt{R}_1 \rangle\!\rangle_{\bar{x}} \oplus \langle\!\langle \mathtt{R}_2 \rangle\!\rangle_{\bar{y}} \\
& = & \langle\!\langle \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \bar{\mathtt{M}}\} \rangle\!\rangle_{\bar{x}} \oplus \langle\!\langle \{\bar{\mathtt{D}}\ \bar{\mathtt{g}};\ \bar{\mathtt{N}}\} \rangle\!\rangle_{\bar{y}} \\
& = & \langle \bar{\mathtt{f}} = \bar{x_1}, \bar{\mathtt{m}} = \bar{x_2} \rangle \oplus \langle \bar{\mathtt{g}} = \bar{y_1}, \bar{\mathtt{n}} = \bar{y_2} \rangle \quad \text{where } \bar{x} = \bar{x_1}, \bar{x_2} \text{ and } \bar{y} = \bar{y_1}, \bar{y_2} \\
& \longrightarrow & \langle \bar{\mathtt{f}} \backslash \bar{\mathtt{g}} = \bar{x_1} \backslash \bar{y_1}, \bar{\mathtt{g}} = \bar{y_1}, \bar{\mathtt{m}} \backslash \bar{\mathtt{n}} = \bar{x_2} \backslash \bar{y_2}, \bar{\mathtt{n}} = \bar{y_2} \rangle \quad \text{by rule } (\oplus_3), \bar{x_1} \backslash \bar{y_1} = \bar{x_3} \text{ and } \bar{x_2} \backslash \bar{y_2} = \bar{x_4}
\end{array}
$$

$$
\begin{array}{rcl}
\langle\!\langle \mathtt{R}' \rangle\!\rangle_{\bar{x},\bar{y}} & = & \langle\!\langle \{\bar{\mathtt{E}}\ \bar{\mathtt{h}};\ \bar{\mathtt{K}}\} \rangle\!\rangle_{\bar{x},\bar{y}} \quad \text{where } \bar{x} \cap \bar{y} = \varnothing \\
& = & \langle \bar{\mathtt{h}} = \bar{x}, \bar{\mathtt{k}} = \bar{y} \rangle \quad \text{where } \bar{\mathtt{h}} = (\bar{\mathtt{f}} \backslash \bar{\mathtt{g}}) \bar{\mathtt{g}} \text{ and } \bar{\mathtt{k}} = (\bar{\mathtt{m}} \backslash \bar{\mathtt{n}}) \bar{\mathtt{n}} \text{ and} \\
& & \bar{\mathtt{f}} \backslash \bar{\mathtt{g}} = \bar{x_3}, \bar{\mathtt{g}} = \bar{y_1}, \bar{x_3} \cap \bar{y_1} = \varnothing \text{ and} \\
& & \bar{\mathtt{m}} \backslash \bar{\mathtt{n}} = \bar{x_4}, \bar{\mathtt{n}} = \bar{y_2}, \bar{x_4} \cap \bar{y_2} = \varnothing \text{ and } \bar{x} = \bar{x_3}, \bar{x_4} \text{ and } \bar{y} = \bar{y_1}, \bar{y_2}
\end{array}
$$

which means $\langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{z}} = \langle\!\langle \mathtt{R}' \rangle\!\rangle_{\bar{z}}$ where $\bar{z} = \bar{x}, \bar{y}$.

- $\mathtt{T} = \mathtt{T}_1 \diamond \mathtt{T}_2$, where $\mathtt{T}_1 = \mathtt{mixin\ C = D\ with\ R}$ and $\mathtt{T}_2 = \mathtt{class\ E\ R'}$. By the rule of compiling reduction, $\mathtt{T} \rightsquigarrow \mathtt{T'} = \mathtt{class\ C(R' \bullet R)}$.

$$\begin{array}{rcl} \langle\!\langle \mathtt{T} \rangle\!\rangle_{\bar{x},\bar{y}} & = & \langle\!\langle \mathtt{T}_1 \diamond \mathtt{T}_2 \rangle\!\rangle_{\bar{x},\bar{y}} \\ & = & (\langle\!\langle \mathtt{T}_1 \rangle\!\rangle_{\bar{x}})\langle\!\langle \mathtt{T}_2 \rangle\!\rangle_{\bar{y}} \quad \text{where } \bar{x} \cap \bar{y} = \varnothing \\ & = & (\langle\!\langle \mathtt{mixin\ C = D\ with\ R} \rangle\!\rangle_{\bar{x}})\langle\!\langle \mathtt{class\ E\ R'} \rangle\!\rangle_{\bar{y}} \\ & = & (\lambda z.z \oplus \langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}})\langle\!\langle \mathtt{R'} \rangle\!\rangle_{\bar{y}} \quad \text{where } z \notin \bar{x} \\ & = & \langle\!\langle \mathtt{R'} \rangle\!\rangle_{\bar{y}} \oplus \langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}} \end{array}$$

$$\begin{array}{rcl} \langle\!\langle \mathtt{class\ C(R' \bullet R)} \rangle\!\rangle_{\bar{x},\bar{y}} & = & \langle\!\langle \mathtt{R' \bullet R} \rangle\!\rangle_{\bar{x},\bar{y}} \quad \text{where } \bar{x} \cap \bar{y} = \varnothing \\ & = & \langle\!\langle \mathtt{R'} \rangle\!\rangle_{\bar{y}} \oplus \langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}} \end{array}$$

  which means $\langle\!\langle \mathtt{T} \rangle\!\rangle_{\bar{z}} = \langle\!\langle \mathtt{T'} \rangle\!\rangle_{\bar{z}}$ where $\bar{z} = \bar{x}, \bar{y}$.

- $\mathtt{T} = \mathtt{class\ C\ R}$ and $\mathtt{R} \rightsquigarrow \mathtt{R'}$. By the rule of compiling reduction, $\mathtt{T} \rightsquigarrow \mathtt{class\ CR'} = \mathtt{T'}$. $\langle\!\langle \mathtt{T} \rangle\!\rangle = \langle\!\langle \mathtt{class\ C\ R} \rangle\!\rangle = \langle\!\langle \mathtt{R} \rangle\!\rangle$ and $\langle\!\langle \mathtt{T'} \rangle\!\rangle = \langle\!\langle \mathtt{class\ CR'} \rangle\!\rangle = \langle\!\langle \mathtt{R'} \rangle\!\rangle$. By induction hypothesis on derivation of $\mathtt{R} \rightsquigarrow^* \mathtt{R'} \Rightarrow \langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}} \longrightarrow^* \langle\!\langle \mathtt{R'} \rangle\!\rangle_{\bar{x}}$, we have $\langle\!\langle \mathtt{T} \rangle\!\rangle_{\bar{x}} = \langle\!\langle \mathtt{R} \rangle\!\rangle_{\bar{x}} \longrightarrow^* \langle\!\langle \mathtt{R'} \rangle\!\rangle_{\bar{x}} = \langle\!\langle \mathtt{T'} \rangle\!\rangle_{\bar{x}}$.

□