Encoding Ownership Types in Java

Nicholas Cameron and James Noble

Victoria University of Wellington, New Zealand

Abstract. Ownership type systems organise the heap into a hierarchy which can be used to support encapsulation properties, effects, and invariants. Ownership types have many applications including parallelisation, concurrency, memory management, and security. In this paper, we show that several flavours and extensions of ownership types can be entirely encoded using the standard Java type system.
Ownership types systems usually require a sizable effort to implement and the relation of ownership types to standard type systems is poorly understood. Our encoding demonstrates the connection between ownership types and parametric and existential types. We formalise our encoding using a model for Java's type system, and prove that it is sound and enforces an ownership hierarchy. Finally, we leverage our encoding to produce lightweight compilers for Ownership Types and Universe Types — each compiler took only one day to implement.

1 Introduction

Ownership types describe the topology of the heap in a program's source code. They come in several varieties (context-parametric [16], Universes [17], Ownership Domains [3], OGJ [26], and more) and have many practical applications, including preventing data races [7,18], parallelisation [15,5], real-time memory management [4], and enforcing architectural constraints [2].

Ownership types usually require large, complicated type systems and compilers, and their relation to standard type theory is not well understood. We give a simple encoding from ownership types to standard generic Java by extending the previously identified relationship between ownership types and parametric types [25,26]. Previous work encoded ownership parameters as type parameters, but treated the the current object's ownership context (the `this` or `This` context) specially; we treat it as a standard type parameter, hidden externally by existential quantification [13]. With this technique we can encode ownership types (with generics and existential quantification), Ownership Domains, and Generic Universe Types. Furthermore, by unpacking the `This` parameter we can support a range of extensions, including inner classes [6], dynamic aliases [15], fields as contexts [12], and existential downcasting [30], within the same standard type system.

Contributions and Organisation The contributions of this paper are: a thorough discussion of how various flavours and extensions of ownership types can be encoded in a standard type system, such as Java's (Sect. 3); a formal type system which captures these concepts (including variations and extensions) and a soundness proof for this system which demonstrates that our encoding enforces the ownership hierarchy (Sect. 4); and compilers for Generic Universe Types and Ownership Types (Sect. 5).

Additionally, we give background on Java generics and ownership types in Sect. 2 and conclude and describe future work in Sect. 6.

2 Background

In this section, we describe features of the Java type system used in our encoding and ownership types.

2.1 Java Generics and Wildcards

Java has featured parametric and existential types since version 5.0, in the form of *generics* and *wildcards* [20]. Java types consist of a class name and a (possibly empty) list of actual type parameters. For example, we can describe a list of books as List<Book>, given a class (or interface) such as, class List<X> {...}. The formal type parameters (e.g., X) may be used in the body of the class; outside the class they must be instantiated with actual parameters (such as Book).

Generic types must be *invariant* with respect to subtyping. However, it is sometimes safe and desirable to make generic types co- or contravariant. To support this, Java has wildcards [27]: an object of type List<? extends Book> is a covariant list of books, that is, a list of some subtype of book. To remain sound, covariant lists must be 'read-only' and contravariant lists 'write-only'; wildcards enforce this. Formal models of Java typically use bounded existential types to represent wildcards [11]: our covariant list is denoted ∃X→[⊥ Book].List<X> (⊥, the bottom type, indicates no lower bound).

A wildcard *hides* a type parameter; for example, we can store (due to subtyping) an object of type List<Book> in a variable of type List<?>: the wildcard hides the *witness type* Book. Java allows the type to be temporarily named, but only as a fresh type variable, this is known as *wildcard capture* and corresponds to *unpacking* an existential type[1]. For example, List<?> can be *capture converted* to List<Z>, where Z is fresh; however, there is no relationship between Z and Book.

2.2 Ownership Types

Ownership types [16] are a mechanism for organising the heap into a hierarchy of contexts. The type system ensures that objects' positions in the hierarchy are reflected by their types. This property allows contexts to be used to specify encapsulation properties (for which ownership types are well known), such as

[1] Subsumption of concrete types to wildcard types corresponds to *packing*.

owners-as-dominators [16] and owners-as-modifiers [17], or to specify effects [15] or invariants [23]. Several mechanisms for reflecting the ownership hierarchy in types have been proposed, these can be separated into parameter-based systems, where types are parameterised by contexts (such as 'vanilla' ownership types [16,15], multiple ownership [12], and ownership domains [3]) and annotation-based systems, where types are annotated to describe relative position in the hierarchy (such as Universes [17]).

There have been several syntactic (but semantically equivalent) variations in the way that 'vanilla' ownership types are denoted. In our source language we prefix an object's type with its owner and parameterise it with actual context parameters. A class is declared without an explicit owner (only formal context parameters) and the `owner` keyword is added to the language for use as an actual context parameter (similarly to the `this` keyword); for example,

```
class List<d> {
    this:Node<d> first;
}
class Node<d> {
    owner:Node<d> next;
    d:Object datum;
}
```
source

Here, a list object owns all of its nodes and the context parameter `d` owns the data in the list. We will use this list as a running example.

Encapsulation and Effects Most ownership systems consist of a descriptive part (describing the topology of the heap) and a 'prescriptive' part, which uses the described topology to specify an encapsulation policy or effect system. Encapsulation properties can restrict aliasing (e.g., owners-as-dominators, associated with 'vanilla' ownership types [16]) or access (e.g., owners-as-modifiers, from Universes [17]). An effect system describes how objects are accessed, rather than restricting access. In this paper we concentrate on the descriptive aspects of ownership and so we will not describe these properties in detail.

2.3 OGJ

Ownership types and generics can be combined in an orthogonal fashion [19,10], giving the benefits and flexibility of both systems. They can also be integrated, as in Ownership Generic Java (OGJ [26]); the benefits of both systems are still gained, but with only a single kind of parameter because type parameters are used to represent context parameters. The only extra ingredient in OGJ (beyond standard Java generics) is a `This` type parameter which represents not a type, but the current context, it is treated specially by the formal type rules.

Our list example can be written in OGJ:

```
class List<D, Owner> {
    Node<D, This> first;
}
class Node<D, Owner> {
    Node<D, Owner> next;
    Object<D> datum;
}
```
OGJ

The syntax is almost identical to the standard ownership types version, other than that the owner of a type is specified as the last type parameter. The semantics, however, are different: all parameters are treated as type parameters by the type system, the usual rules for type checking Java are applied, rather than special ownership types rules. The exception is in dealing with the `This`, which is treated as a keyword, rather than a variable, and is thus a valid owner for `first` even though it is not declared.

Featherweight Generic Confinement (FGC [25]) uses the same representation of contexts as type parameters, but without any support for the `This` context. The result is a 'shallow ownership' system which supports encapsulation by package, but not on a per-object basis.

3 Encoding Ownership Types into Java

In this section we describe how we encode source ownership types programs into Java. As in FGC [25] and OGJ [26], we represent the owner of a class and its context parameters with type parameters. We create a formal type parameter (`This`) to represent the `this` context [13], bounded above by `Owner`. The inside relation (context ordering) is encoded by subtyping (as in OGJ). Since `this` cannot be named outside its class declaration, we must hide the corresponding `This` type parameter where it appears in types, using Java wildcards; conveniently, the wildcard will inherit the bound declared on `This`. Our basic list example (Sect. 2.2) is encoded as:

```
class List<D, Owner extends World, This extends Owner> {
    Node<D, This, ?> first;
}
class Node<D, Owner extends World, This extends Owner> {
    Node<D, Owner, ?> next;
    Object<D, ?> datum;
}
```
Java

Actual context parameters are either `World` (a class or interface which represents the root context) or formal context variables (either quantified or with class scope). The inherited or explicit bounds on these type variables produce a partial ordering on type parameters corresponding to the ownership hierarchy[2]. Because there are no concrete types representing contexts (other than `World`), the hierarchy is an illusion: an omniscient type checker would know that all type variables ultimately hold `World`. The 'opaqueness' of existential types ensures that the illusory hierarchy is respected during type checking.

Type systems must treat existentially quantified variables as hiding unique types; this gives the correct behaviour for ownership types in our encoding by treating each `This` context as unique. If we did not always hide the `This` parameter, ownership typing would not be effective[3]:

```
List<World, World, X> l1 = new List<World, World, X>();
List<World, World, X> l2 = new List<World, World, X>();
l1.first = l2.first;        //OK, but should be an error
```
Java

Universes Universes [17] are an annotation-based ownership system. Types may be annotated with `rep` (denoting that objects of this type are owned by `this`), `peer` (objects are in the same context as `this`), or `any` (objects are in an unknown context). Generic Universe Types [19] integrate type parametricity and universe modifiers; the programmer can write types such as `rep List<peer Book>`, which represents a list (owned by the current object) of books in the current context. Universe types and ownership types describe the same hierarchies [9].

Generic Universe Types can be encoded into ownership types [9], and then into Java using the above scheme. The only obstacle is that the universe modifier `any` corresponds to an existentially quantified owner (see below); `any` can be encoded as an unbounded wildcard.

Ownership Domains Ownership domains [3] support more flexible topologies and a more flexible encapsulation property than 'vanilla' ownership types. Topologically, ownership domains allow for multiple contexts (called domains) per object; all contexts are nested within the object's owner and other objects can belong to any of these contexts.

To support multiple contexts per object in our encoding we allow multiple parameters in place of the single `This` parameter. All these parameters are given the upper bound of `Owner` and all must be hidden with wildcards to create the phantom ownership hierarchy. Types are encoded in the same way as for 'vanilla' ownership types.

For example, the following class has two domains and a single domain parameter:

```
class C<domP> {  domain dom1, dom2;  }
```
ODs

[2] There are effectively two subtype hierarchies: one of real objects with `Object` at its root, and one of ownership contexts with `World` at its root.

[3] In this section we will use wildcards in `new` expressions, this is not allowed in Java and we describe how to avoid this in Sect. 5.

It is encoded as the Java class,

```
class C<DomP, Owner, Dom1 extends Owner, Dom2 extends Owner> {}
```
Java

3.1 Extensions to Ownership Types

There has been much work on making ownership type systems more descriptive and more flexible. Generally, the underlying ownership hierarchy is unchanged, but the language's types can describe it more precisely, usually combined with a relaxation of encapsulation properties in certain circumstances. In this section, we describe several extensions to ownership types and how they can be encoded in Java.

Bounds Context parameters may be given upper and lower bounds with respect to the ownership hierarchy [15,10]. These are usually denoted `inside` and `outside`, respectively. For example, `class C<a outside owner, b inside a>`.

Upper bounds on context parameters can easily be replicated using upper bounds on the corresponding type parameters (e.g. `B extends A`). The encoded bounds are with respect to the subtype hierarchy, within which the ownership hierarchy is encoded. Lower bounds cannot be encoded in Java without changing the type system to support lower bounds on type parameters.

Context Parametric Methods Methods may be parameterised by contexts [14,29] in the same way as they can be parameterised by types in Java. This allows for better code reuse. For example:

```
<a,b> a:Node<b> next(a:Node<b> n) {
    return n.next;
}
```
source

The `next` method will work for all possible nodes; without context parametric methods, such a method could not be written.

Context parametric methods are easily encoded as type parametric Java methods, upper bounds on context parameters can be handled as above:

```
<A,B> Node<B, A, ?> next(Node<B, A, ?> n) {
    return n.next;
}
```
Java

Inner Classes Ownership type systems can be made more flexible by giving inner classes access to the `this` and `owner` parameters of the surrounding class [6]. This increases the descriptiveness of the type system because more contexts can be named inside an inner class. Owners-as-dominators can be sensibly relaxed to allow instantiations of inner classes to hold references to their surrounding objects (e.g., the `curNode` field in the following example). This allows iterators to be implemented in an 'owners-as-dominators' system, an early obstacle to acceptance of ownership type systems. We extend our list example:

```
class List<d> {
    ...
    class Iterator {
        List.this:Node<d> curNode;
        d:Object next() { return curNode = curNode.next() }
    }
}

class Client {
    void m(this:List<world> l) {
        this:Iterator i = l.new this:Iterator()
        world:Object first = i.next();
    }
}
```

source

In the encoding, inner classes must be able to name the contexts of their surrounding class; this happens naturally in Java: an inner class can name type parameters of its surrounding class. However, we must be careful not to hide the generated type parameter by adding `This` parameters for both inner and outer classes. We accomplish this by appending the name of the class to the names of the `Owner` and `This` parameters (we elide some bounds in the example):

```
class List<D, Owner, This extends Owner> {
    ...
    class Iterator<It_Owner, It_This extends It_Owner> {
        Node<D, This, ?> curNode;
        Object<D, ?> next() { return curNode = curNode.next() }
    }
}

class Client<Owner, This extends Owner> {
    void m(List<World, This, ?> l) {
        Iterator<This, ?> i = l.new Iterator<This, ?>();
        Object<World, ?> first = i.next();
    }
}
```

Java

Dynamic Aliases An alternative solution to the iterators under owners-as-dominators problem is to allow *dynamic aliases* [15]; that is, allow variables on the stack to reference objects which break owners-as-dominators, and only enforce owners-as-dominators on the heap. Dynamic aliases achieve this by allowing local variables to be used as contexts. Extending the original list example:

```
class Iterator<d> {
    owner:Node<d> curNode;
    d:Object next() { return curNode = curNode.next() }
}

class Client {
    void m(final this:List<world> l) {
        l:Iterator<world> i = new l:Iterator<world>();
        world:Object first = i.next();
    }
}
```

source

The variable `l` cannot be named outside of `m`, and so the dynamic alias to `i` (owned by `l`) cannot be stored in the heap. It is only sound to use final variables to name contexts.

An object's context is represented by its hidden `This` argument; therefore, encoding dynamic aliases in Java requires naming that argument using a fresh, temporary type variable which is introduced as an extra type parameter to a method. Unpacking the hidden `This` argument to the fresh variable is achieved by wildcard capture:

```
class Iterator<D, Owner extends World, This extends Owner> {
    Node<D, Owner, ?> curNode;
    Object<D, ?> next() { return curNode = curNode.next() }
}

class Client<Owner extends World, This extends Owner> {
    void m(List<World, This, ?> l) {
        this.mAux(l)
    }

    <L> void mAux(List<World, This, L> l) {
        Iterator<World, L, ?> i = new Iterator<World, L, ?>;
        Object<World, ?> first = i.next();
    }
}
```

Java

The wildcard which hides `l`'s `This` argument is capture converted to the fresh type variable `L` when `mAux` is called. Using `l` as an owner in the source program is encoded to using `L` (`l`'s `This` argument) as an owner. `L` can only be named within the scope of `mAux`, and this corresponds to the scope of `l`.

Our example is simple because it does not require other state to be passed to `mAux`. In a more realistic example, we would need to pass any data accessed in `m` to `mAux`, and back again if it is not passed by reference. A simpler encoding is to modify the original method so that the `This` argument of `l` is captured by calling `m` (rather than when calling `mAux`). The simpler encoding only works if the variable being used as a context is an argument rather than a local variable. Note that the call-sites of `m` do not have to be modified, despite the extra type parameter, due to Java's type parameter inference. The simpler encoding of our encoding is

```
class Client<Owner, This> {
    <L> void m(List<World, This, L> l) { ... }  //body as mAux
}
```
Java

Fields as Contexts Similarly to local variables, final fields can be used to name contexts [12], this again improves flexibility. We can extend the list example:

```
class List<d> {
    this:Node<d> first;
    first:Object f2;    //owned by a field
}
```
source

Paths of final fields may also be used as contexts [12], e.g., one could allow the type `f3.first:Object`, where `f3` is a final field of type `List`.

We encode fields used as contexts by adding their hidden `This` parameters to the class's parameter list:

```
class List<D, Owner extends World, This extends Owner, First> {
    Node<D, This, ? extends First> first;
    Object<First, ?> f2;
}
```
Java

Instantiating this class requires that the value of `first` is passed into the constructor, wildcard capture is used to name `First` and then both `this` and `First` are hidden by wildcards.

Existential Quantification Just as type variables may be quantified existentially, so may context variables [10]. This allows for existential ownership types such as ∃`o.o:Object` or ∃`o.this:List<o>`. Such quantification has two benefits: context variance, that is subtyping which is variant with respect to the ownership hierarchy, and expressing partial knowledge about contexts (i.e., an unknown context or some unknown context within another known context). Existential quantification is the mechanism which underlies a number of proposals involving some kind of variance annotations on contexts [22,8].

Existentially quantified contexts can be encoded as wildcards. Since wildcards are syntactic sugar for existential types, this is not surprising. Both upper and lower bounds can be straightforwardly encoded. The only difficulty is if quantified contexts have both upper and lower bounds, which is not supported by

Java wildcards. However, because quantification is usually provided by variance annotations or wildcard-like syntax, this should not be a problem.

Existential Downcasting Downcasting is a common feature in programs, especially those that do not use generics. When downcasting from type `A` to type `B`, if `B` has context parameters which `A` does not, these must be synthesised. Wrigstad and Clarke propose the use of "existential owners" to handle these introduced context parameters [30]. For example:

```
void m(this:Object x) {
    this:List<d> l = (this:List<d>) x;
    d:Object first = l.first.datum;
    l.first.datum = new d:Object();
}
```
source

Here `x` is cast from type `this:Object` to `this:List<d>`, the `d` context parameter is a fresh context (an "existential owner") that can be named in the scope of the method, and allows operations on `l` to take place. Objects owned by `d` cannot be stored in the heap, outside of the original data structure, because `d` can only be named locally. Note that there is no explicit quantification, "existential owners" correspond to unpacked context existential types [8].

We can cast `x` to a type where `D` (the encoding of `d`) is hidden by a wildcard. We cannot cast directly to a type containing `D` because `D` is not in scope. We must split the method in order to name `D` using capture conversion:

```
void m(Object<This, ?> x) {
    this.mAux((List<?, This, ?>) x);
}
<D> void mAux(List<D, This, ?> l) {
    Object<D, ?> first = l.first.datum;
    l.first.datum = new Object<D, ?>();
}
```
Java

Owners-as-Dominators The owners-as-dominators property specifies that all reference paths from the root of the ownership hierarchy to any object pass through that object's owner: owners dominate reference paths. The property is enforced by restricting which contexts can be named: if only contexts outside the current context can be named, then no references can exist *into* contexts other than for the current `this` object.

We have previously sketched how owners-as-dominators can be supported in an encoding of ownership into Java [13]. This approach can be duplicated here with the same drawback: owners-as-dominators can only be guaranteed if the Java compiler is modified, it cannot be supported as a pre-processor step like the rest of the encodings discussed. The modifications are not major: a small change to the well-formedness rules for classes and types to ensure that context parameters are outside the declared owner (the usual requirement for ownership types to support owners-as-dominators). The issue is that at intermediate steps of computation the compiler might allow the `This` parameter to be named in types: this is not a problem for descriptive ownership because is is only temporary, but can allow owners-as-dominators to be violated.

4 Formalisation

To show that our encoding does in fact demonstrate the behaviour of an ownership types system, we extend a model for the Java type system with elements of our encoding and runtime ownership information. Our formalisation (Tame FJ_{Own}) follows the approach of OGJ [26], in representing context parameters as type parameters, but, by supporting existential types, we do not need any special machinery to deal with ownership issues.

The bulk of the formal system is relatively standard or follows Tame FJ [11]. Differences from Tame FJ to model ownership are highlighted in grey. We also add field assignment, `null`, a heap, and casting (to model dynamic downcasts), and make some small improvements elsewhere, these changes are not highlighted. For the sake of brevity, we do not describe the parts unchanged from Tame FJ. Parts of the operational semantics, well-formed environments and heaps, auxiliary functions, and rules for using the heap as an environment are relegated to the appendix.

Syntax The syntax of Tame FJ_{Own} is given in Fig. 1. For convenience, and following OGJ [26], we syntactically separate types and type parameters used to represent contexts from regular types: we use τ to denote types which represent contexts, T to denote regular types, and $\mathcal{T}$ to denote either type; likewise for parameters, we use `O` to denote type parameters which represent context parameters, `X` for regular type parameters, and $\mathcal{X}$ for either kind. Importantly, the two kinds of type are treated almost identically by the type system. We can do without this convenience by examining the type's top supertype: contexts will be bounded by `World`, other types by `Object`.

We allow values (v, which are addresses and `null`; the latter corresponds to `World`) to be context (and thus type) parameters at runtime so that we can prove enforcement of the ownership hierarchy (see Sect. 4.1); values are not allowed as parameters in source code.

We use a few shorthands for types: `C` for `C<>`, and R for $\exists\emptyset.R$.

Well-formed Types Well-formed types are defined in Fig. 2. In F-Class and F-Object, we do not check that the type parameter in the `This` position is well-formed. Instead we check that it is in the environment and is bounded below by bottom. This ensures that it is always an in-scope variable (in fact it is usually a quantified variable, although this does not need to be enforced) and that no other type can be derived to be a subtype of it (as would be the case if it had a lower bound). This ensures that the `This` context cannot be named by using subsumption.

e	$::=$	$\gamma \mid \texttt{null} \mid e.\texttt{f} \mid e.\texttt{f = } e \mid e.\texttt{<}\overline{P}\texttt{,}\,\overline{\mathcal{P}}\texttt{>m(}\overline{e}\texttt{)}$ $\mid \texttt{new C<}\overline{\mathcal{T}}\texttt{,}\star\texttt{>} \mid (T)e$	*expressions*
v	$::=$	$\iota \mid \texttt{null}$	*values*
Q	$::=$	$\texttt{class C<}\overline{\texttt{X}\lhd T}\texttt{,}\,\overline{\texttt{O}\lhd \tau}\texttt{,Owner}\lhd \mathcal{T}\texttt{, This}\lhd \mathcal{T}\texttt{>} \lhd N\ \{\overline{T\,\texttt{f;}}\ \overline{M}\}$	
M	$::=$	$\texttt{<}\overline{\texttt{X}\lhd T}\texttt{,}\,\overline{\texttt{O}\lhd \mathcal{T}}\texttt{>}\ T\,\texttt{m}(\overline{T\,\texttt{x}})\ \{\texttt{return}\ e\texttt{;}\}$	*method declarations*
N	$::=$	$\texttt{C<}\overline{T}\texttt{,}\,\overline{\mathcal{T}}\texttt{>} \mid \texttt{Object<}\mathcal{T}\texttt{, }\mathcal{T}\texttt{>} \mid \texttt{World<>}$	*class types*
R	$::=$	$N \mid \texttt{X}$	*non-existential types*

T, U	$::=$	$\exists\Delta.N \mid \exists\emptyset.\texttt{X}$	*types*	$\mathcal{T}$	$::=$	$T \mid \tau$	*types and contexts*
P	$::=$	$T \mid \star$	*method type parameters*	$\mathcal{P}$	$::=$	$\mathcal{T} \mid \star$	*method parameters*
$\mathcal{X},\mathcal{Y}$	$::=$	$\texttt{X} \mid \texttt{O} \mid v$	*type parameters*	τ	$::=$	$\texttt{World} \mid \texttt{O} \mid v$	*contexts*

Δ	$::=$	$\overline{\mathcal{X}\to[B_l\ \ B_u]}$	*type environments*	$\texttt{x}, \texttt{this}$	*variables*
B	$::=$	$\mathcal{T} \mid \bot$	*bounds*	$\texttt{X}, \texttt{Y}$	*type variables*
Γ	$::=$	$\overline{\gamma:T}$	*variable environments*	$\texttt{O}, \texttt{Owner}, \texttt{This}$	*context variables*
γ	$::=$	$\iota \mid \texttt{x}$	*locations or variables*	ι	*locations*
$\mathcal{H}$	$::=$	$\overline{\iota \to \{N;\ \overline{\texttt{f}\to v}\}}$	*heaps*	$\texttt{C}, \texttt{Object}, \texttt{World}$	*class names*
				$\texttt{f}, \texttt{g}$	*field names*
				$\texttt{m}$	*method names*

Fig. 1. Syntax of Tame FJ_{Own}.

Well-formed types: $\boxed{\Delta \vdash B\ \text{OK},\ \Delta \vdash \mathcal{P}\ \text{OK},\ \Delta \vdash R\ \text{OK}}$

$$\frac{\mathcal{X} \in \Delta}{\Delta \vdash \mathcal{X}\ \text{OK}}\ \text{(F-Var)} \qquad \frac{}{\Delta \vdash \texttt{World<>}\ \text{OK}}\ \text{(F-World)} \qquad \frac{}{\Delta \vdash \bot\ \text{OK}}\ \text{(F-Bottom)} \qquad \frac{}{\Delta \vdash \star\ \text{OK}}\ \text{(F-Star)} \qquad \frac{\Delta \vdash \Delta'\ \text{OK} \quad \Delta, \Delta' \vdash \texttt{N}\ \text{OK}}{\Delta \vdash \exists\Delta'.\texttt{N}\ \text{OK}}\ \text{(F-Exists)}$$

$$\frac{\begin{array}{c}\Delta \vdash \overline{T}, \overline{\tau}, \tau_o\ \text{OK} \\ \overline{\mathcal{T}} = \overline{T}, \overline{\tau}, \tau_o, \tau_t \qquad \Delta(\tau_t) = [\bot\ \ \mathcal{T}] \\ \texttt{class C<}\overline{\mathcal{X}\lhd \mathcal{T}_u}\texttt{>} \lhd N\{\ldots\} \\ \Delta \vdash \overline{\mathcal{T} <: [\overline{\mathcal{T}/\mathcal{X}}]\mathcal{T}_u}\end{array}}{\Delta \vdash \texttt{C<}\overline{\mathcal{T}}\texttt{>}\ \text{OK}}\ \text{(F-Class)} \qquad \frac{\Delta \vdash \tau_o\ \text{OK} \quad \Delta(\tau_t) = [\bot\ \ \mathcal{T}]}{\Delta \vdash \texttt{Object<}\tau_o\texttt{,}\tau_t\texttt{>}\ \text{OK}}\ \text{(F-Object)}$$

Fig. 2. Tame FJ_{Own} well-formed types, type environments, and heaps.

Type Checking Selected type rules are given in Fig. 3. Object creation (T-New) does not take any (value) parameters (i.e., we don't have constructors, at run-

Expression typing: $\boxed{\Delta;\Gamma \vdash e : T}$

$$\frac{\Delta \vdash T \text{ OK}}{\Delta;\Gamma \vdash \texttt{null} : T} \ (\text{T-NULL})$$

$$\frac{\Delta;\Gamma \vdash e : \exists\Delta'.N \quad fType(\texttt{f}, N) = T' \quad \Delta;\Gamma \vdash e' : T \quad \Delta,\Delta' \vdash T <: T'}{\Delta;\Gamma \vdash e.\texttt{f} \texttt{ = } e' : T} \ (\text{T-ASSIGN})$$

$$\frac{\Delta \vdash \overline{\mathcal{T}}, \mathcal{T} \text{ OK} \quad \Delta \vdash \exists \texttt{O} \rightarrow [\bot\ \mathcal{T}].\texttt{C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,O>} \text{ OK}}{\Delta;\Gamma \vdash \texttt{new C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\star\texttt{ >} : \exists \texttt{O} \rightarrow [\bot\ \mathcal{T}].\texttt{C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,O>}} \ (\text{T-NEW})$$

Class typing: $\boxed{\vdash Q \text{ OK}}$

$$\frac{\begin{array}{c}\Delta = \overline{\texttt{X}\rightarrow[\bot\ T_u]},\ \texttt{Owner}\rightarrow[\bot\ \tau_o]\texttt{,This}\rightarrow[\bot\ \texttt{Owner}]\texttt{,}\overline{\texttt{O}\rightarrow[\bot\ \tau_u]} \\ \emptyset \vdash \Delta \text{ OK} \qquad \Delta \vdash N, \overline{T} \text{ OK} \\ \overline{\mathcal{X}} = \overline{\texttt{X}}, \overline{\texttt{O}}, \texttt{Owner}, \texttt{This} \qquad \Delta; \texttt{this:C<}\overline{\mathcal{X}}\texttt{>} \vdash \overline{M} \text{ OK in C} \\ N = \texttt{D<}\overline{\mathcal{T}}\texttt{,Owner,This>} \qquad \Delta \vdash N <: \texttt{Object<Owner,This>}\end{array}}{\vdash \texttt{class C<}\overline{\texttt{X}\lhd\ T_u}\texttt{, }\overline{\texttt{O}\lhd\ \tau_u}\texttt{,Owner}\lhd\ \tau_o\texttt{,This}\lhd\ \texttt{Owner >} \lhd\ N\{\overline{\texttt{T f;}}\ \overline{M}\} \text{ OK}} \ (\text{T-CLASS})$$

Fig. 3. Selected Tame FJ_{Own} expression and class typing rules.

time all fields are initialised to `null`). This requires `null` and the T-NULL rule. Initialising objects in this way is necessary so that fields owned by `This` can be initialised. The actual type parameter in the `This` position of `new` expressions must always be $\star$, so no actual parameter is named at initialisation. New objects are given existential types, with the `This` parameter existentially quantified (bounded above by the `Owner` parameter), which ensures that the actual `This` parameter can never be named directly. The extra well-formedness premise in T-NEW is stricter than the usual well-formedness premise and ensures that the type parameters are well-formed without the extra, quantified parameter in the environment.

We add a rule for casting (T-CAST), which is standard. Unlike in Featherweight Java, we do not distinguish between up-, down-, and stupid casts.

In T-CLASS we enforce that the declared upper bound of `This` is `Owner`[4]. The last two premises ensure that declared classes fall under the `Object` hierarchy and are not subtypes of `World`, which means they cannot be used as context parameters, and that the `Owner` and `This` parameters are invariant with respect to inheritance. The latter is an important sanity condition of our encoding of ownership and corresponds to the well-known condition on inheritance and own-

[4] The re-ordering of type parameters is a hangover from supporting owners-as-dominators, where the lower bound of each `O` is `Owner`.

ership [15]. We assume that `Object` is declared with parameters `Owner` and `This` with the usual bounds.

Operational Semantics Operational semantics are mostly defined in the appendix; the most interesting change from Tame FJ is in object creation:

$$\frac{\iota \notin dom(\mathcal{H}) \qquad fields(\texttt{C}) = \overline{\texttt{f}} \qquad \mathcal{H}' = \mathcal{H}, \iota \rightarrow \{\texttt{C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\iota\texttt{>}; \overline{\texttt{f}\rightarrow\texttt{null}}\}}{\texttt{new C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\star\texttt{ >}; \mathcal{H} \leadsto \iota; \mathcal{H}'}$$

(R-New)

A new object's runtime type (stored in the heap) is formed by replacing the $\star$ used in the program source by the new object's address. Together with the usual rules of substitution (in method invocation), occurrences of both `this` and `This` in class declarations are replaced by the instantiation's address (ι), unifying the two representations of the object. Together with the quantification in T-New, objects are, in effect, packed into existential types, with the object's address as witness 'type'.

4.1 Discussion

Ownership types are intrinsically dependent because they reflect objects' positions in the heap. We have shown that ownership types can be encoded as parametric types in a Java-like type system, reminiscent of *phantom types* [21]. Phantom types are parametric types where type parameters are never used as types[5]. Phantom types are used in Haskell to simulate values in types, without the complexity and decidability issues of full dependent types [21]. This is exactly what our system is doing with respect to ownership information. We conclude then, that ownership type systems are, in some sense, no more complex than standard parametric type systems such as Java's. Despite their dependent character, the full power of dependent types is not required to support ownership type systems. However, we should not overstep the mark and assume that type parametricity is the only, or even the best, foundational model for ownership types.

Most of the ownership features described in Sect. 3 can be accommodated in Tame FJ_{Own}. Inner classes require encoding and are discussed below. Paths of final fields cannot easily by encoded in our formal system. Generic Universe Types [19] can be accommodated after encoding. Ownership domains would require a small extension to the formal system, which we have avoided for the sake of simplicity: each class has a list of `This` type parameters rather than a single parameter. Each parameter represents a domain. Since this change merely changes `This` to $\overline{\texttt{This}}$, we expect very few changes to be necessary to accommodate it.

The extensions to support ownership domains and inner classes (below) are fairly superficial changes, modifying only the restrictions on type parameters and which type parameters are hidden in T-New.

[5] More precisely, phantom type parameters are not used on the right hand side of the definition of a type constructor.

Inner Classes Encoding inner classes in Tame FJ_{Own} would require a small extension to our formalisation. References to the surrounding object and the type parameters of the surrounding object must be made available to objects of the inner class. Extending Tame FJ_{Own} could be done by adopting a nesting of classes and objects in the class table and heap or by adding a field to each class pointing to the surrounding object, and type parameters for the surrounding classes' type parameters; object creation becomes more complex, but otherwise the calculus is not changed too much. The iterator as inner class example from Sect. 3.1 is encoded as (we elide bounds):

```java
class Iterator<D, L_Owner, L_This, It_Owner, It_This> {
    List<D, L_Owner, L_This> out;
    Node<D, L_This, ?> curNode;
    Object<It_This, ?> privField;

    Object<D, ?> next() {...}
}

class Client<Owner, This> {
    <LT> void m(List<World, This, LT> l) {
        Iterator<World, This, LT, This, ?> i
            = new Iterator<World, This, LT, This, ?>();
        i.out = l;
        Object<World, ?> first = i.next();
    }
}
```

Java

We must use (a presumably capture converted) type variable (`LT`) for the `This` parameter of `l`, provide `l`'s type parameters to `i`, and must instantiate the `out` field of `i`.

Type Soundness We have proved type soundness for Tame FJ_{Own} in the usual way [28] by proving progress and preservation theorems. For the most part, our proofs follow those of Tame FJ [11]; they can be downloaded from [1].

In standard existential type systems, witness types are known at runtime, and type soundness guarantees that no type errors involving witness types occur, even though the type system has only partial knowledge of these types during type checking. Taking this approach with Tame FJ_{Own} would not be very informative, since all witness types (according to T-New) will be $\star$. Our static types hold *more* information (the ownership hierarchy) than is represented by the 'witness types'. Our soundness result proves that Tame FJ_{Own} does enforce the ownership hierarchy, i.e., Tame FJ_{Own} enforces not only strict type soundness (well-typed programs won't access non-existent fields or methods), but also that objects reside in the context described by their type. Ownership information is represented at runtime by storing the object's address into it's `This` position (in

R-New), the address propagates into other ownership positions by substitution (in R-Invk).

In proving type soundness for Tame FJ_{Own}, we have proved that a one-stage type checker (corresponding to an integration of our pre-processor and the Java type checker) is sound, rather than proving that a two-stage type checker (corresponding to pre-processing and then Java type checking, as in our implementation) is sound. Our approach is theoretically more direct and reflects what we envision to be the long term use of our techniques.

5 Implementation

We have implemented compilers for Java with ownership types and Generic Universe Types by using the techniques described in this paper. Our implementations are simple source to source translators which pre-process source code to plain Java; the Java compiler is then used to type check and compile the code. Most type errors are caught by the Java compiler, only a few are handled by our translators. Our translators are extensions to the parser and AST elements of the JKit Java compiler [24]. We encode one class at a time and do not need to be aware of the whole program. Generated classes will behave well together, but are incompatible with plain Java classes[6].

Our approach supports ownership and universe types on top of nearly the entire Java Language, including generics, arrays (including the various kinds of array initialisers), interfaces, inner classes (but not anonymous classes), statics, and wildcards.

Our implementations are very much prototypes, an industrial strength compiler would integrate the encoding with Java type checking, as opposed to our two-stage process. Integration would allow for meaningful error messages and support for effects and encapsulation properties. Furthermore, to be usable, a language requires more than a compiler, libraries must be supported, either by support for non-ownership aware classes (currently, all classes must be written with ownership types) or by producing a set of ownership annotated libraries (or a combination of the two approaches).

Our compilers can be downloaded from [1].

5.1 Ownership Types

Our source syntax is mostly similar to that used throughout this paper. We support ownership, context parameters, orthogonal generics, context- and type-parametric methods, final method parameters as contexts (for dynamic aliases), existential quantification in the form of context wildcards, and inner classes with access to the contexts and context parameters of the surrounding object. We do not support local variables (other than method parameters) or fields as contexts.

[6] Strictly, since we generate plain Java, one could write classes which behave well with the generated classes, but not in a way which behaves nicely with the source classes.

We support standard casting, including to wildcard owners, but do not directly support "existential owners".

Our compiler strips owners and context parameters and replaces them with type parameters, in both class declarations and in types; in the latter case, using wildcards in the This position.

The Java compiler does not permit wildcard parameters when objects are instantiated. To get around this, we use the Owner type parameter in the This position (because it is the only type parameter which satisfies the declared bound) and immediately cast to the required wildcard type (which inherits the upper bound):

```
new world:Object()                                    //source syntax
new Object<World, ?>()                                 //pseudo-Java
(OwnedObject<World, ?>) new OwnedObject<World, World>()   //Java
```

Note also that, as in OGJ, we have to add an OwnedObject which extends Object at the root of our class hierarchy to take the encoded ownership parameters. All classes must extend OwnedObject (rather than Object, which may happen implicitly) and all uses of Object changed to OwnedObject. In the source syntax, the object's owner is implicit in the extends clause, and so translation of the superclass type must be treated differently from other types. Because we add OwnedObject and World to our runtime, we must import these classes into each encoded class file.

5.2 Generic Universe Types

The source syntax is pretty standard for generic universes, e.g., rep List<any Object>. The translation is much simpler than for ownership types since we do not have to translate context parameters, only types. Most of the issues faced are similar, and simpler, than in the ownership types case: we must check for universe modifiers on all types (but not in extends clauses), Object is translated to OwnedObject, and care must be taken with array types.

6 Conclusion and Future Work

In this paper we have shown how ownership types, Generic Universe Types, Ownership Domains, and a range of extensions to ownership type systems can be encoded using Java Generics and wildcards. The key concepts are the representation of context parameters as type parameters, the reification of this as a type parameter, the hiding of that type parameter using wildcards, and the phantom ownership hierarchy thus created. Our developments shed light on the type-theoretic foundations of ownership types and offer a route for practical compilers constructed upon existing technology.

Future Work The main thrust of future work will be in supporting owners-as-dominators, and other encapsulation polices and effects, in our formal work and compilers. This will require integrating our translating compiler with an existing Java compiler, which will also allow for better error messages and more efficient type checking. We would also like to encode libraries with ownership type information for use with our compilers.

References

1. Accompanying webpage. https://ecs.victoria.ac.nz/Main/Encoding.
2. Marwan Abi-Antoun and Jonathan Aldrich. Ownership Domains in the Real World. In *International Workshop on Aliasing, Confinement and Ownership in object-oriented programming (IWACO)*, 2008.
3. Jonathan Aldrich and Craig Chambers. Ownership Domains: Separating Aliasing Policy from Mechanism. In *European Conference on Object Oriented Programming (ECOOP)*, 2004.
4. Austin Armbruster, Jason Baker, Antonio Cunei, Chapman Flack, David Holmes, Filip Pizlo, Edward Pla, Marek Prochazka, and Jan Vitek. A Real-Time Java Virtual Machine with Applications in Avionics. *Transactions on Embedded Computing Systems*, 7(1):1–49, 2007.
5. Robert L. Bocchino, Jr., Vikram S. Adve, Danny Dig, Sarita V. Adve, Stephen Heumann, Rakesh Komuravelli, Jeffrey Overbey, Patrick Simmons, Hyojin Sung, and Mohsen Vakilian. A type and effect system for deterministic parallel java. In *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 2009.
6. Boyapati, Liskov, and Shrira. Ownership Types for Object Encapsulation. In *Principles of Programming Languages (POPL)*, 2003.
7. Chandrasekhar Boyapati, Robert Lee, and Martin C. Rinard. Ownership Types for Safe Programming: Preventing Data Races and Deadlocks. In *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 2002.
8. Nicholas Cameron. *Existential Types for Variance — Java Wildcards and Ownership Types*. PhD thesis, Imperial College London, 2009.
9. Nicholas Cameron and Werner Dietl. Comparing Universes and Existential Ownership Types. In *International Workshop on Aliasing, Confinement and Ownership in object-oriented programming (IWACO)*, 2009.
10. Nicholas Cameron and Sophia Drossopoulou. Existential Quantification for Variant Ownership. In *European Symposium on Programming Languages and Systems (ESOP)*, 2009.
11. Nicholas Cameron, Sophia Drossopoulou, and Erik Ernst. A Model for Java with Wildcards. In *European Conference on Object Oriented Programming (ECOOP)*, 2008.
12. Nicholas Cameron, Sophia Drossopoulou, James Noble, and Matthew Smith. Multiple Ownership. In *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 2007.
13. Nicholas Cameron and James Noble. OGJ Gone Wild. In *International Workshop on Aliasing, Confinement and Ownership in object-oriented programming (IWACO)*, 2009.
14. David G. Clarke. *Object Ownership and Containment*. PhD thesis, School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia, 2001.

15. David G. Clarke and Sophia Drossopoulou. Ownership, Encapsulation and the Disjointness of Type and Effect. In *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 2002.
16. David G. Clarke, John M. Potter, and James Noble. Ownership Types for Flexible Alias Protection. In *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 1998.
17. David Cunningham, Werner Dietl, Sophia Drossopoulou, Adrian Francalanza, Peter Müller, and Alexander J. Summers. Universe Types for Topology and Encapsulation. In *Formal Methods for Components and Objects (FMCO)*, 2008.
18. David Cunningham, Sophia Drossopoulou, and Susan Eisenbach. Universe Types for Race Safety. In *Verification and Analysis of Multi-threaded Java-like Programs (VAMP)*, 2007.
19. Werner Dietl, Sophia Drossopoulou, and Peter Müller. Generic Universe Types. In *European Conference on Object Oriented Programming (ECOOP)*, 2007.
20. James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java Language Specification Third Edition*. Addison-Wesley, Boston, Mass., 2005.
21. Ralf Hinze. *The Fun of Programming*, pages 245–262. Palgrave Macmillan, 2003. Fun with phantom types.
22. Yi Lu and John Potter. On Ownership and Accessibility. In *European Conference on Object Oriented Programming (ECOOP)*, 2006.
23. Peter Müller, Arnd Poetzsch-Heffter, and Gary T. Leavens. Modular Invariants for Layered Object Structures. *Science of Computer Programming*, 62(3):253–286, October 2006.
24. David Pearce. Jkit compiler. http://www.ecs.vuw.ac.nz/~djp/jkit.
25. Alex Potanin, James Noble, Dave Clarke, and Robert Biddle. Featherweight generic confinement. *J. Funct. Program.*, 16(6):793–811, 2006.
26. Alex Potanin, James Noble, Dave Clarke, and Robert Biddle. Generic Ownership for Generic Java. In *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 2006.
27. Mads Torgersen, Christian Plesner Hansen, Erik Ernst, Peter von der Ahé, Gilad Bracha, and Neal Gafter. Adding Wildcards to the Java Programming Language. *Journal of Object Technology*, 3(11):97–116, 2004. Special issue: OOPS track at SAC 2004, Nicosia/Cyprus.
28. Andrew K. Wright and Matthias Felleisen. A Syntactic Approach to Type Soundness. *Information and Computation*, 115(1):38–94, 1994.
29. Tobias Wrigstad. *Ownership-Based Alias Managemant*. PhD thesis, KTH, Sweden, 2006.
30. Tobias Wrigstad and David G. Clarke. Existential Owners for Ownership Types. *Journal of Object Technology*, 6(4), 2007.

A Elided Figures

These sections will be provided in an accompanying technical report if the paper is accepted.

Lookup Functions

$$\frac{}{\mathit{fields}(\texttt{Object}) = \emptyset} \qquad \frac{\texttt{class C<}\overline{\mathcal{X} \lhd \mathcal{T}_u}\texttt{>} \lhd \texttt{D<...>} \; \{\overline{U\,\texttt{f;}} \;\; \overline{M}\}}{\mathit{fields}(\texttt{C}) = \mathit{fields}(\texttt{D}), \overline{\texttt{f}}}$$

$$\frac{\begin{array}{c}\texttt{class C<}\overline{\mathcal{X} \lhd \mathcal{T}_u}\texttt{>} \lhd N\{\overline{U\,\texttt{f;}} \;\; \overline{M}\} \\ \texttt{f} \notin \overline{\texttt{f}}\end{array}}{\mathit{fType}(\texttt{f}, \texttt{C<}\overline{\mathcal{T}}\texttt{>}) = \mathit{fType}(\texttt{f}, [\overline{\mathcal{T}/\mathcal{X}}]N)} \qquad \frac{\texttt{class C<}\overline{\mathcal{X} \lhd \mathcal{T}_u}\texttt{>} \lhd N\{\overline{U\,\texttt{f;}} \;\; \overline{M}\}}{\mathit{fType}(\texttt{f}_i, \texttt{C<}\overline{\mathcal{T}}\texttt{>}) = [\overline{\mathcal{T}/\mathcal{X}}]U_i}$$

$$\frac{\begin{array}{c}\texttt{class C<}\overline{\mathcal{X} \lhd \mathcal{T}_u}\texttt{>} \lhd N\{\overline{\texttt{U f;}} \;\; \overline{M}\} \\ \texttt{m} \notin \overline{M}\end{array}}{\mathit{mBody}(\texttt{m}, \texttt{C<}\overline{\mathcal{T}}\texttt{>}) = \mathit{mBody}(\texttt{m}, \texttt{[}\overline{\mathcal{T}\texttt{/}\mathcal{X}}\texttt{]}N)} \qquad \frac{\begin{array}{c}\texttt{class C<}\overline{\mathcal{X} \lhd \mathcal{T}_u}\texttt{>} \lhd N\{\overline{U'\,\texttt{f;}} \;\; \overline{M}\} \\ \texttt{<}\overline{\mathcal{Y} \lhd \mathcal{T}'_u}\texttt{>} \; U\,\texttt{m}(\overline{U\,\texttt{x}}) \;\; \{\texttt{return } e_0\texttt{;}\} \in \overline{M}\end{array}}{\mathit{mBody}(\texttt{m}, \texttt{C<}\overline{\mathcal{T}}\texttt{>}) = (\overline{\texttt{x}}, \texttt{[}\overline{\mathcal{T}\texttt{/}\mathcal{X}}\texttt{]}e_0)}$$

$$\frac{\begin{array}{c}\texttt{class C<}\overline{\mathcal{X} \lhd \mathcal{T}_u}\texttt{>} \lhd N\{\overline{U\,\texttt{f;}} \;\; \overline{M}\} \\ \texttt{m} \notin \overline{M}\end{array}}{\mathit{mType}(\texttt{m}, \texttt{C<}\overline{\mathcal{T}}\texttt{>}) = \mathit{mType}(\texttt{m}, \texttt{[}\overline{\mathcal{T}\texttt{/}\mathcal{X}}\texttt{]}N)} \qquad \frac{\begin{array}{c}\texttt{class C<}\overline{\mathcal{X} \lhd \mathcal{T}_u}\texttt{>} \lhd N\{\overline{U'\,\texttt{f;}} \;\; \overline{M}\} \\ \texttt{<}\overline{\mathcal{Y} \lhd \mathcal{T}'_u}\texttt{>} \; U\,\texttt{m}(\overline{U\,\texttt{x}}) \;\; \{\texttt{return } e_0\texttt{;}\} \in \overline{M}\end{array}}{\mathit{mType}(\texttt{m}, \texttt{C<}\overline{\mathcal{T}}\texttt{>}) = \texttt{[}\overline{\mathcal{T}\texttt{/}\mathcal{X}}\texttt{]}(\texttt{<}\overline{\mathcal{Y} \lhd \mathcal{T}'_u}\texttt{>}\overline{U} \rightarrow U)}$$

Fig. 4. Method and field lookup functions for Tame FJ_{Own}.

Auxiliary Functions: $\boxed{uBound_\Delta(B)}$ $\boxed{match(\langle\overline{R},\overline{U}\rangle,\overline{P},\overline{\mathcal{Y}},\overline{\mathcal{T}})}$ $\boxed{sift(\overline{R},\overline{T},\overline{\mathcal{X}})}$ $\boxed{\Downarrow_\Delta T}$

$$uBound_\Delta(B) = \begin{cases} uBound_\Delta(B_u), & if\ B = \mathcal{X}\ and\ \Delta(\mathcal{X}) = [B_l\ \ B_u] \\ B, & if\ B \neq \mathcal{X} \end{cases}$$

$$\frac{\begin{array}{c}\forall j\ where\ \mathcal{P}_j = \star : \mathcal{Y}_j \in fv(\overline{R'}) \qquad \forall i\ where\ \mathcal{P}_i \neq \star : \mathcal{T}_i = \mathcal{P}_i \\ \vdash R \sqsubset: [\overline{\mathcal{T}/\mathcal{Y}}, \overline{\mathcal{T}'/\mathcal{X}}]R' \\ dom(\overline{\Delta}) = \overline{\mathcal{X}} \qquad fv(\overline{\mathcal{T}}, \overline{\mathcal{T}'}) \cap \overline{\mathcal{Y}}, \overline{\mathcal{X}} = \emptyset \end{array}}{match(\langle\overline{R}, \exists\Delta.R'\rangle, \overline{\mathcal{P}}, \overline{\mathcal{Y}}, \overline{\mathcal{T}})}$$

$$\frac{\mathcal{X} \in \overline{\mathcal{Y}}}{sift((R,\overline{R}),\ (\mathcal{X},\overline{U}),\ \overline{\mathcal{Y}}) = sift(\overline{R},\ \overline{U},\ \overline{\mathcal{Y}})}$$

$$\frac{\mathcal{X} \notin \overline{\mathcal{Y}} \qquad sift(\overline{R},\ \overline{U},\ \overline{\mathcal{Y}}) = (\overline{R'},\ \overline{U'})}{sift((R,\overline{R}),\ (\mathcal{X},\overline{U}),\ \overline{\mathcal{Y}}) = \langle(R,\overline{R'}),\ (\mathcal{X},\overline{U'})\rangle}$$

$$\frac{}{sift(\emptyset,\ \emptyset,\ \overline{\mathcal{Y}}) = \langle\emptyset,\ \emptyset\rangle} \qquad \frac{sift(\overline{R},\ \overline{U},\ \overline{\mathcal{Y}}) = (\overline{R'},\ \overline{U'})}{sift((R,\overline{R}),\ (\exists\Delta.N,\overline{U}),\ \overline{\mathcal{Y}}) = \langle(R,\overline{R'}),\ (\exists\Delta.N,\overline{U'})\rangle}$$

$$\frac{\mathcal{X} \notin dom(\Delta)}{\Downarrow_\Delta \mathcal{X} = \mathcal{X}} \qquad \frac{\Delta(\mathcal{X}) = [B_l\ \ B_u]}{\Downarrow_\Delta \mathcal{X} = \Downarrow_\Delta B_u} \qquad \frac{}{\Downarrow_\Delta \exists\Delta'.N = \exists\Delta,\Delta'.N}$$

Fig. 5. Auxiliary functions for Tame FJ_{Own}.

Subclasses: $\boxed{\vdash R \sqsubset: R}$

$$\frac{\texttt{class C<}\overline{\mathcal{X} \triangleleft \mathcal{T}_u}\texttt{>} \triangleleft N\{\ldots\}}{\vdash \texttt{C<}\overline{\mathcal{T}}\texttt{>} \sqsubset: \texttt{[}\overline{\mathcal{T}/\mathcal{X}}\texttt{]}N} \text{ (SC-Sub-Class)}$$

$$\frac{}{\vdash R \sqsubset: R} \text{ (SC-Reflex)}$$

$$\frac{\vdash R \sqsubset: R'' \qquad \vdash R'' \sqsubset: R'}{\vdash R \sqsubset: R'} \text{ (SC-Trans)}$$

Extended subclasses: $\boxed{\Delta \vdash B \sqsubset: B}$

$$\frac{\texttt{class C<}\overline{\mathcal{X} \triangleleft \mathcal{T}_u}\texttt{>} \triangleleft N\{\ldots\}}{\Delta \vdash \exists \Delta' . \texttt{C<}\overline{\mathcal{T}}\texttt{>} \sqsubset: \exists \Delta' . \texttt{[}\overline{\mathcal{T}/\mathcal{X}}\texttt{]}N} \text{ (XS-Sub-Class)}$$

$$\frac{}{\Delta \vdash \bot \sqsubset: B} \text{ (XS-Bottom)}$$

$$\frac{}{\Delta \vdash B \sqsubset: B} \text{ (XS-Reflex)}$$

$$\frac{\Delta \vdash B \sqsubset: B'' \quad \Delta \vdash B'' \sqsubset: B'}{\Delta \vdash B \sqsubset: B'} \text{ (XS-Trans)}$$

$$\frac{\begin{array}{c} dom(\Delta') \cap fv(\exists \overline{\mathcal{X} \to \texttt{[}B_l\ B_u\texttt{]}} . N) = \emptyset \qquad fv(\overline{\mathcal{T}}) \subseteq dom(\Delta, \Delta') \\ \Delta, \Delta' \vdash \overline{\texttt{[}\overline{\mathcal{T}/\mathcal{X}}\texttt{]}B_l <: \mathcal{T}} \qquad \Delta, \Delta' \vdash \overline{\mathcal{T} <: \texttt{[}\overline{\mathcal{T}/\mathcal{X}}\texttt{]}B_u} \end{array}}{\Delta \vdash \exists \Delta' . \texttt{[}\overline{\mathcal{T}/\mathcal{X}}\texttt{]}N \sqsubset: \exists \overline{\mathcal{X} \to \texttt{[}B_l\ B_u\texttt{]}} . N} \text{ (XS-Env)}$$

Subtypes: $\boxed{\Delta \vdash B <: B}$

$$\frac{\Delta \vdash B \sqsubset: B'}{\Delta \vdash B <: B'} \text{ (S-SC)}$$

$$\frac{\Delta \vdash B <: B'' \qquad \Delta \vdash B'' <: B'}{\Delta \vdash B <: B'} \text{ (S-Trans)}$$

$$\frac{\Delta(\mathcal{X}) = \texttt{[}B_l\ B_u\texttt{]}}{\begin{array}{c}\Delta \vdash \mathcal{X} <: B_u \\ \Delta \vdash B_l <: \mathcal{X}\end{array}} \text{ (S-Bound)}$$

Fig. 6. Tame FJ_{Own} subclasses, extended subclasses, and subtypes.

Well-formed type environments: $\boxed{\Delta \vdash \Delta \text{ OK}}$

$$\frac{}{\Delta \vdash \emptyset \text{ OK}} \text{ (F-Env-Empty)}$$

$$\frac{\begin{array}{c} \Delta, \mathcal{X} \to \texttt{[}B_l\ B_u\texttt{]}, \Delta' \vdash B_l \text{ OK} \qquad \Delta, \mathcal{X} \to \texttt{[}B_l\ B_u\texttt{]}, \Delta' \vdash B_u \text{ OK} \\ \Delta \vdash uBound_\Delta(B_l) \sqsubset: uBound_\Delta(B_u) \\ \Delta \vdash B_l <: B_u \qquad \Delta, \mathcal{X} \to \texttt{[}B_l\ B_u\texttt{]} \vdash \Delta' \text{ OK} \end{array}}{\Delta \vdash \mathcal{X} \to \texttt{[}B_l\ B_u\texttt{]}, \Delta' \text{ OK}} \text{ (F-Env)}$$

Well-formed heaps: $\boxed{\Delta \vdash \mathcal{H} \text{ OK}}$

$$\frac{\begin{array}{c} \forall \iota \to \{N;\ \overline{\texttt{f} \to v}\} \in \mathcal{H} : \\ \emptyset \vdash N \text{ OK} \\ \overline{fType(\texttt{f}, N) = T} \qquad \emptyset, \mathcal{H} \vdash \overline{\texttt{v} : T} \end{array}}{\vdash \mathcal{H} \text{ OK}} \text{ (F-Heap)}$$

Fig. 7. Tame FJ_{Own} well-formed type environments and heaps.

Expression typing: $\boxed{\Delta; \Gamma \vdash e : T}$

$$\frac{}{\Delta; \Gamma \vdash \gamma : \Gamma(\gamma)} \quad \text{(T-VAR)}$$

$$\frac{\Delta \vdash T \text{ OK}}{\Delta; \Gamma \vdash \texttt{null} : T} \quad \text{(T-NULL)}$$

$$\frac{\begin{array}{c}\Delta; \Gamma \vdash e : \exists \Delta' . N \\ fType(\texttt{f}, N) = T\end{array}}{\Delta; \Gamma \vdash e\texttt{.f} : \Downarrow_{\Delta'} T} \quad \text{(T-FIELD)}$$

$$\frac{\begin{array}{c}\Delta; \Gamma \vdash e : \exists \Delta' . N \qquad fType(\texttt{f}, N) = T' \\ \Delta; \Gamma \vdash e' : T \qquad \Delta, \Delta' \vdash T <: T'\end{array}}{\Delta; \Gamma \vdash e\texttt{.f = } e' : T} \quad \text{(T-ASSIGN)}$$

$$\frac{\begin{array}{c}\Delta; \Gamma \vdash e : U \\ \Delta \vdash T <: U \qquad \Delta \vdash T \text{ OK}\end{array}}{\Delta; \Gamma \vdash \texttt{(}T\texttt{)}\, e : T} \quad \text{(T-CAST)}$$

$$\frac{\begin{array}{c}\Delta; \Gamma \vdash e : U \\ \Delta \vdash U <: T \qquad \Delta \vdash T \text{ OK}\end{array}}{\Delta; \Gamma \vdash e : T} \quad \text{(T-SUBS)}$$

$$\frac{\begin{array}{c}\Delta \vdash \overline{\mathcal{T}}, \mathcal{T} \text{ OK} \\ \Delta \vdash \exists \texttt{O} \rightarrow \texttt{[}\bot\ \mathcal{T}\texttt{].C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,O>} \text{ OK}\end{array}}{\Delta; \Gamma \vdash \texttt{new C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\star\texttt{ >} : \exists \texttt{O} \rightarrow \texttt{[}\bot\ \mathcal{T}\texttt{].C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,O>}} \quad \text{(T-NEW)}$$

$$\frac{\begin{array}{c}\Delta; \Gamma \vdash e : \exists \Delta' . N \qquad mType(\texttt{m},\ N) = \texttt{<}\overline{\mathcal{X} \lhd\ B}\texttt{>}\overline{U} \rightarrow U \\ \Delta \vdash \overline{\mathcal{P}} \text{ OK} \qquad \Delta; \Gamma \vdash \overline{e : \exists \Delta . R} \\ match(sift(\overline{R}, \overline{U}, \overline{\mathcal{X}}), \overline{\mathcal{P}}, \overline{\mathcal{X}}, \overline{\mathcal{T}}) \\ \Delta, \Delta', \overline{\Delta} \vdash \overline{\mathcal{T} <: \texttt{[}\overline{\mathcal{T}/\mathcal{X}}\texttt{]}B} \qquad \Delta, \Delta', \overline{\Delta} \vdash \overline{R <: \texttt{[}\overline{\mathcal{T}/\mathcal{X}}\texttt{]}U}\end{array}}{\Delta; \Gamma \vdash e\texttt{.<}\overline{\mathcal{P}}\texttt{>m(}\overline{e}\texttt{)} : \Downarrow_{\Delta', \overline{\Delta}} \texttt{[}\overline{\mathcal{T}/\mathcal{X}}\texttt{]}U} \quad \text{(T-INVK)}$$

Fig. 8. Tame FJ_{Own} expression typing rules.

Method typing: $\boxed{\Delta \vdash M \text{ OK } \texttt{in C}}$

$$\frac{\begin{array}{c}\Delta' = \overline{\mathcal{X} \rightarrow \texttt{[}\bot\ \ \mathcal{T}_u\texttt{]}} \qquad \Delta \vdash \Delta' \text{ OK} \qquad \Delta, \Delta' \vdash T, \overline{T} \text{ OK} \\ \texttt{class C<...>} \lhd N\{\ldots\} \\ \Delta, \Delta'; \Gamma, \overline{\texttt{x}:T} \vdash e : T \\ override(\texttt{m}, \texttt{N}, \texttt{<}\overline{\mathcal{X} \lhd\ \mathcal{T}_u}\texttt{>}\overline{\texttt{T}} \rightarrow T)\end{array}}{\Delta; \Gamma \vdash \texttt{<}\overline{\mathcal{X} \lhd\ \mathcal{T}_u}\texttt{>}T\texttt{m(}\overline{T\ \texttt{x}}\texttt{)}\ \{\texttt{return } e\} \text{ OK } \texttt{in C}} \quad \text{(T-METHOD)}$$

$$\frac{mType(\texttt{m}, N) = \texttt{<}\overline{\mathcal{X} \lhd\ \mathcal{T}}\texttt{>}\overline{T} \rightarrow T}{override(\texttt{m}, N, \texttt{<}\overline{\mathcal{X} \lhd\ \mathcal{T}}\texttt{>}\overline{T} \rightarrow T)} \quad \text{(T-OVERRIDE)}$$

$$\frac{mType(\texttt{m}, N) \quad undefined}{override(\texttt{m}, N, \texttt{<}\overline{\mathcal{X} \lhd\ \mathcal{T}}\texttt{>}\overline{T} \rightarrow T)} \quad \text{(T-OVERRIDEUNDEF)}$$

Fig. 9. Tame FJ_{Own} method typing rules.

Computation: $\boxed{e;\mathcal{H} \rightsquigarrow e;\mathcal{H}}$

$$\frac{\mathcal{H}(\iota) = \{N; \overline{\texttt{f}\to v}\}}{\iota.\texttt{f}_i;\mathcal{H} \rightsquigarrow v_i;\mathcal{H}}$$

(R-Field)

$$\frac{\begin{array}{c}\mathcal{H}(\iota) = \{N; \overline{\texttt{f}\to v}\} \\ \mathcal{H}' = \mathcal{H}[\iota \mapsto \{N; v'; \overline{\texttt{f}\to v}[\texttt{f}_i \mapsto v]\}]\end{array}}{\iota.\texttt{f}_i\ \texttt{=}\ v;\mathcal{H} \rightsquigarrow v;\mathcal{H}'}$$

(R-Assign)

$$\frac{\begin{array}{c}\iota \notin dom(\mathcal{H}) \qquad fields(\texttt{C}) = \overline{\texttt{f}} \\ \mathcal{H}' = \mathcal{H}, \iota \to \{\texttt{C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\iota\texttt{>}; \overline{\texttt{f}\to\texttt{null}}\}\end{array}}{\texttt{new C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\star\texttt{ >};\mathcal{H} \rightsquigarrow \iota;\mathcal{H}'}$$

(R-New)

$$\frac{\begin{array}{c}\mathcal{H}(\iota) = \{N; v; ...\} \qquad \overline{\mathcal{H}(\iota) = \{N...\}} \\ mBody(\texttt{m}, N) = (\overline{\texttt{x}}, e_0) \\ mType(\texttt{m}, N) = \texttt{<}\overline{\mathcal{X} \lhd B}\texttt{>}\overline{U} \ \to\ U \\ match(sift(\overline{N}, \overline{U}, \overline{\mathcal{X}}), \overline{\mathcal{P}}, \overline{\mathcal{X}}, \overline{\mathcal{T}})\end{array}}{\iota\texttt{.<}\overline{\mathcal{P}}\texttt{>m(}\overline{\iota}\texttt{)};\mathcal{H} \rightsquigarrow \texttt{[}\overline{\iota\texttt{/x}}\texttt{, }\iota\texttt{/this, }\overline{\mathcal{T}\texttt{/}\mathcal{X}}\texttt{]}e_0;\mathcal{H}}$$

(R-Invk)

$$\frac{\mathcal{H}(\iota) = \{N...\} \qquad \emptyset \vdash N <: T}{\texttt{(}T\texttt{)}\iota;\mathcal{H} \rightsquigarrow; \iota;\mathcal{H}}$$

(R-Cast)

$$\frac{}{\texttt{(}T\texttt{)null};\mathcal{H} \rightsquigarrow; \texttt{null};\mathcal{H}}$$

(R-Cast-Null)

Fig. 10. Tame FJ_{Own} reduction rules.

Congruence: $\boxed{e;\mathcal{H} \rightsquigarrow e;\mathcal{H}}$

$$\frac{e;\mathcal{H} \rightsquigarrow e';\mathcal{H}' \qquad e' \neq \texttt{err}}{e\texttt{.f};\mathcal{H} \rightsquigarrow e'\texttt{.f};\mathcal{H}'}$$

(RC-Field)

$$\frac{e_1;\mathcal{H} \rightsquigarrow e_1';\mathcal{H}' \qquad e_1' \neq \texttt{err}}{\mathcal{H}; e_1\texttt{.f = }e_2 \rightsquigarrow \mathcal{H}'; e_1'\texttt{.f = }e_2}$$

(RC-Assign-1)

$$\frac{e_2;\mathcal{H} \rightsquigarrow e_2';\mathcal{H}' \qquad e_2' \neq \texttt{err}}{\iota\texttt{.f = }e_2;\mathcal{H} \rightsquigarrow \iota\texttt{.f = }e_2';\mathcal{H}'}$$

(RC-Assign-2)

$$\frac{e;\mathcal{H} \rightsquigarrow e';\mathcal{H}' \qquad e' \neq \texttt{err}}{e\texttt{.<}\overline{\mathcal{P}}\texttt{>m(}\overline{e}\texttt{)};\mathcal{H} \rightsquigarrow e'\texttt{.<}\overline{\mathcal{P}}\texttt{>m(}\overline{e}\texttt{)};\mathcal{H}'}$$

(RC-Inv-Recv)

$$\frac{e;\mathcal{H} \rightsquigarrow e';\mathcal{H}' \qquad e' \neq \texttt{err}}{\iota\texttt{.<}\overline{\mathcal{P}}\texttt{>m(}\overline{v}\texttt{,}e\texttt{,}\overline{e}\texttt{)};\mathcal{H} \rightsquigarrow \iota\texttt{.<}\overline{\mathcal{P}}\texttt{>m(}\overline{v}\texttt{,}e'\texttt{,}\overline{e}\texttt{)};\mathcal{H}'}$$

(RC-Inv-Arg)

$$\frac{e;\mathcal{H} \rightsquigarrow e';\mathcal{H}' \qquad e' \neq \texttt{err}}{\texttt{(}T\texttt{)}e;\mathcal{H} \rightsquigarrow \texttt{(}T\texttt{)}e';\mathcal{H}'}$$

(RC-Cast)

Fig. 11. Tame FJ_{Own} reduction rules.

Exceptional Computation and Error Propogation: $\boxed{e;\mathcal{H} \leadsto e;\mathcal{H}}$

$$\frac{}{\texttt{null.f};\mathcal{H} \leadsto \texttt{err};\mathcal{H}}\ (\text{R-Field-Null}) \qquad \frac{}{\texttt{null.f = } e;\mathcal{H} \leadsto \texttt{err};\mathcal{H}}\ (\text{R-Assign-Null}) \qquad \frac{}{\texttt{null.<}\overline{\mathcal{P}}\texttt{>m(}\overline{e}\texttt{)};\mathcal{H} \leadsto \texttt{err};\mathcal{H}}\ (\text{R-Invk-Null})$$

$$\frac{\mathcal{H}(\iota) = \{N...\} \qquad \emptyset \nvdash N <: T}{\texttt{(}T\texttt{)}\iota;\mathcal{H} \leadsto;\texttt{err};\mathcal{H}}\ (\text{R-Bad-Cast})$$

$$\frac{e;\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}{e\texttt{.f};\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}\ (\text{RC-Field-Err}) \qquad \frac{e_1;\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}{e_1\texttt{.f = }e_2;\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}\ (\text{RC-Assign-1-Err}) \qquad \frac{e_2;\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}{e_1\texttt{.f = }e_2;\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}\ (\text{RC-Assign-2-Err})$$

$$\frac{e;\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}{e\texttt{.<}\overline{\mathcal{P}}\texttt{>m(}\overline{e}\texttt{)};\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}\ (\text{RC-Invk-Recv-Err}) \qquad \frac{e;\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}{\iota\texttt{.<}\overline{\mathcal{P}}\texttt{>m(}\overline{v}\texttt{,}e\texttt{,}\overline{e}\texttt{)};\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}\ (\text{RC-Invk-Arg-Err}) \qquad \frac{e;\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}{\texttt{(}T\texttt{)}e;\mathcal{H} \leadsto \texttt{err};\mathcal{H}'}\ (\text{RC-Cast-Err})$$

Fig. 12. Tame FJ_{Own} reduction rules.

$$\frac{\begin{array}{c}\mathcal{H} = \overline{\iota \rightarrow \{\texttt{C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\iota'\texttt{>; ...}\}} \\ \overline{\iota \rightarrow \texttt{[}\bot\ \mathcal{T}\texttt{]}}, \Delta; \overline{\iota : \overline{N}.\texttt{C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\iota'\texttt{>}} \vdash e : T\end{array}}{\Delta;\mathcal{H} \vdash e : T}\ (\text{H-T}) \qquad \frac{\begin{array}{c}\mathcal{H} = \overline{\iota \rightarrow \{\texttt{C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\iota'\texttt{>; ...}\}} \\ \overline{\iota \rightarrow \texttt{[}\bot\ \mathcal{T}\texttt{]}}, \Delta \vdash T <: T'\end{array}}{\mathcal{H}, \Delta \vdash T <: T'}\ (\text{H-S})$$

$$\frac{\begin{array}{c}\mathcal{H} = \overline{\iota \rightarrow \{\texttt{C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\iota'\texttt{>; ...}\}} \\ \overline{\iota \rightarrow \texttt{[}\bot\ \mathcal{T}\texttt{]}}, \Delta \vdash T\ \text{OK}\end{array}}{\mathcal{H}, \Delta \vdash T\ \text{OK}}\ (\text{H-F})$$

Fig. 13. Using the heap as an environment in Tame FJ_{Own}.

B Proofs in Detail

For all lemmas and theorems we require the additional premise that the program is well-formed, i.e., for all class declarations, Q, in the program, $\vdash$ Q OK. Throughout, we assume the Barendregt convention, i.e., bound and free variables are distinct.

To use the premises of a judgement in a proof where we have the conclusion an inversion lemma is required. However, where a judgement is syntax directed we reduce trivial overhead by using the inversion of the judgment directly in the proof.

Lemma 1 (Substitution preserves subclassing).

If:

a. $\vdash \mathtt{R} \sqsubset: \mathtt{R}'$

then:

$\vdash [\overline{\mathtt{T/X}}]\mathtt{R} \sqsubset: [\overline{\mathtt{T/X}}]\mathtt{R}'$

Proof *by structural induction on the derivation of* $\vdash \mathtt{R} \sqsubset: \mathtt{R}'$ *with a case analysis on the last step:*

Case 1 (SC-Trans)

1.	$\vdash \mathtt{R} \sqsubset: \mathtt{R}''$	*by premises of* SC-Trans
2.	$\vdash \mathtt{R}'' \sqsubset: \mathtt{R}'$	*by premises of* SC-Trans
3.	$\vdash [\overline{\mathtt{T/X}}]\mathtt{R} \sqsubset: [\overline{\mathtt{T/X}}]\mathtt{R}''$	*by ind hyp,* **1**
4.	$\vdash [\overline{\mathtt{T/X}}]\mathtt{R}'' \sqsubset: [\overline{\mathtt{T/X}}]\mathtt{R}'$	*by ind hyp,* **2**
5.	$\vdash [\overline{\mathtt{T/X}}]\mathtt{R} \sqsubset: [\overline{\mathtt{T/X}}]\mathtt{R}'$	*by* **3**, **4**, SC-Trans

Case 2 (SC-Reflex)

trivial

Case 3 (SC-Sub-Class)

1.	$\mathtt{R} = \mathtt{C}\langle\overline{\mathtt{U}}\rangle$	*by def* SC-Sub-Class
2.	$\mathtt{R}' = [\overline{\mathtt{U/Y}}]\mathtt{N}$	*by def* SC-Sub-Class
3.	$\mathtt{class\ C}\langle\overline{\mathtt{Y...}}\rangle \lhd \mathtt{N...}$	*by premise* SC-Sub-Class
4.	$[\overline{\mathtt{T/X}}]\mathtt{R} = \mathtt{C}\langle\overline{[\overline{\mathtt{T/X}}]\mathtt{U}}\rangle$	*by* **1**, *def subst*
5.	$\vdash \mathtt{C}\langle\overline{[\overline{\mathtt{T/X}}]\mathtt{U}}\rangle \sqsubset: [\overline{[\overline{\mathtt{T/X}}]\mathtt{U/Y}}]\mathtt{N}$	*by* **3**, SC-Sub-Class
6.	$\vdash \mathtt{class\ C}\langle\overline{\mathtt{Y...}}\rangle \lhd \mathtt{N...}$ OK	*by* **3**, *wf-prog*
7.	$\overline{\mathtt{Y...}} \vdash \mathtt{N}$ OK	*by* **6**, *def* T-Class
8.	$[\overline{\mathtt{T/X}}][\overline{\mathtt{U/Y}}]\mathtt{N} = [\overline{[\overline{\mathtt{T/X}}]\mathtt{U/Y}}]\mathtt{N}$	*by* **7**
9.	$\vdash [\overline{\mathtt{T/X}}]\mathtt{R} \sqsubset: [\overline{\mathtt{T/X}}]\mathtt{R}'$	*by* **5**, **8**, **2**, **4**

$\square$

Lemma 2 (Subsititution preserves *match*ing).

If:

a. $match(\langle \overline{\mathtt{R}}, \overline{\exists\Delta.\mathtt{R}'}\rangle, \overline{\mathtt{P}}, \overline{\mathtt{Y}}, \overline{\mathtt{U}})$

b. $(\overline{\mathtt{X}} \cup fv(\overline{\mathtt{T}})) \cap \overline{\mathtt{Y}}) = \emptyset$

then:

$match(\langle \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{R}}, \overline{[\overline{\mathtt{T}/\mathtt{X}}]\exists\Delta.\mathtt{R}'}\rangle, \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{P}}, \overline{\mathtt{Y}}, \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{U}})$

Proof

1.	$\forall i$ *where* $\mathtt{P}_i \neq \star : \mathtt{U}_i = \mathtt{P}_i$	*by premises of* match
2.	$\forall j$ *where* $\mathtt{P}_j = \star : \mathtt{Y}_j \in fv(\overline{\mathtt{R}'})$	
3.	$\vdash \mathtt{R} \sqsubset: \overline{[\overline{\mathtt{U}/\mathtt{Y}}, \overline{\mathtt{U}'/\mathtt{Z}}]\mathtt{R}'}$	
4.	$dom(\overline{\Delta}) = \overline{\mathtt{Z}}$	
5.	$fv(\overline{\mathtt{U}}, \overline{\mathtt{U}'}) \cap \overline{\mathtt{Y}}, \overline{\mathtt{Z}} = \emptyset$	
6.	$\overline{\mathtt{Z}}$ *are fresh*	*by* **4**, *Barendregt*
7.	$\vdash \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{R} \sqsubset: [\overline{\mathtt{T}/\mathtt{X}}][\overline{\mathtt{U}/\mathtt{Y}}, \overline{\mathtt{U}'/\mathtt{Z}}]\mathtt{R}'}$	*by* **3**, *lemma 1*
8.	$\vdash \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{R} \sqsubset: [\overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{U}/\mathtt{Y}}, \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{U}'/\mathtt{Z}}][\overline{\mathtt{T}/\mathtt{X}}]\mathtt{R}'}$	*by* **7**, **6**, **b**
9.	$\forall i$ *where* $[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{P}_i \neq \star : [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{U}_i = [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{P}_i$	*by* **1**, *def subst*
10.	$\forall j$ *where* $[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{P}_j = \star : \mathtt{Y}_j \in fv(\overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{R}'})$	*by* **2**, **b**, *def subst*
11.	$fv(\overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{U}}, \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{U}'}) \cap \overline{\mathtt{Y}}, \overline{\mathtt{Z}} = \emptyset$	*by* **5**, **6**, **b**
12.	$match(\langle \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{R}}, \overline{[\overline{\mathtt{T}/\mathtt{X}}]\exists\Delta.\mathtt{R}'}\rangle, \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{P}}, \overline{\mathtt{Y}}, \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{U}})$	*by* **9**, **10**, **8**, **4**, **11**

$\square$

Lemma 3 (Substitution on $\overline{\mathtt{U}}$ preserves *sift*).

If:

a. $sift(\overline{\mathtt{R}}, \overline{\mathtt{U}}, \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}_r}, \overline{\mathtt{T}_r}\rangle$

b. $(fv(\overline{\mathtt{T}}) \cup \overline{\mathtt{X}}) \cap \overline{\mathtt{Y}} = \emptyset$

then:

$sift(\overline{\mathtt{R}}, \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{U}}, \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}_r}, \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{T}_r}\rangle$

Proof *by structural induction on the derivation of* $sift(\overline{\mathtt{R}}, \overline{\mathtt{U}}, \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}_r}, \overline{\mathtt{T}_r}\rangle$ *with a case analysis on the last step:*

Case 1 $\overline{\mathtt{U}} = \emptyset$

trivial

Case 2 $\overline{\mathtt{U}} = \exists\Delta.\mathtt{N}, \overline{\mathtt{U}'}$

1. $\overline{\mathtt{R}} = \mathtt{R}, \overline{\mathtt{R}'}$

2. $\langle \overline{\mathtt{R}_r}, \overline{\mathtt{T}_r} \rangle = \langle \mathtt{R}, \overline{\mathtt{R}''},\ \exists \Delta\,.\,\mathtt{N}, \overline{\mathtt{U}''} \rangle$

} *by def* sift

3. $sift(\overline{\mathtt{R}'},\ \overline{\mathtt{U}'},\ \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}''},\ \overline{\mathtt{U}''} \rangle$ *by premise* sift

4. $\overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}} = \exists\,[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\Delta\,.\,[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{N}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}'}$ *by def subst*

5. $sift(\overline{\mathtt{R}'},\ \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}'},\ \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}''},\ \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}''} \rangle$ *by* **3**, **b**, *ind hyp*

6. $sift(\overline{\mathtt{R}}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}}, \overline{\mathtt{Y}}) =$ *by* **5**, **4**, **1**, sift
$\langle \mathtt{R}, \overline{\mathtt{R}''},\ \exists\,[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\Delta\,.\,[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{N}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}''} \rangle$

7. $sift(\overline{\mathtt{R}}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}}, \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}_r}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{T}_r} \rangle$ *by* **6**, **2**

Case 3 $\overline{\mathtt{U}} = \exists\emptyset\,.\,\mathtt{Z}, \overline{\mathtt{U}'} \wedge \mathtt{Z} \notin \overline{\mathtt{Y}}$

1. $\overline{\mathtt{R}} = \mathtt{R}, \overline{\mathtt{R}'}$

2. $\langle \overline{\mathtt{R}_r}, \overline{\mathtt{T}_r} \rangle = \langle \mathtt{R}, \overline{\mathtt{R}''},\ \exists\emptyset\,.\,\mathtt{Z}, \overline{\mathtt{U}''} \rangle$

} *by def* sift

3. $sift(\overline{\mathtt{R}'},\ \overline{\mathtt{U}'},\ \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}''},\ \overline{\mathtt{U}''} \rangle$ *by premise* sift

4. $\overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}} = [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\exists\emptyset\,.\,\mathtt{Z}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}'}$ *by def subst*

5. $sift(\overline{\mathtt{R}'},\ \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}'},\ \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}''},\ \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}''} \rangle$ *by* **3**, **b**, *ind hyp*

Case analysis on Z:

Case 1 $\mathtt{Z} \notin \overline{\mathtt{X}}$

1.1. $\overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}} = \exists\emptyset\,.\,\mathtt{Z}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}'}$ *by* **4**

1.2. $sift(\overline{\mathtt{R}}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}}, \overline{\mathtt{Y}}) =$ *by* **5**, **1.1**, **1**, sift
$\langle \mathtt{R}, \overline{\mathtt{R}''},\ \exists\emptyset\,.\,\mathtt{Z}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}''} \rangle$

1.3. $sift(\overline{\mathtt{R}}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}}, \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}_r}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{T}_r} \rangle$ *by* **1.2**, **2**

Case 2 $\mathtt{Z} \in \overline{\mathtt{X}}$

2.1. $\mathtt{Z} = \mathtt{X}_i$

2.2. $[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\exists\emptyset\,.\,\mathtt{Z} = \mathtt{T}_i$ *by* **2.1**, *def subst*

2.3. $\mathtt{T}_i = \exists\emptyset\,.\,\mathtt{Z}' \wedge \mathtt{Z}' \notin \overline{\mathtt{Y}} \vee \mathtt{T}_i = \exists\Delta\,.\,\mathtt{N}$ *by* **b**

2.4. $sift(\overline{\mathtt{R}}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}}, \overline{\mathtt{Y}}) =$ *by* **5**, **2.3**, **2.2**, **4**, **1**, sift
$\langle \mathtt{R}, \overline{\mathtt{R}''},\ \mathtt{T}_i, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}''} \rangle$

2.5. $sift(\overline{\mathtt{R}}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{U}}, \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}_r}, \overline{[\overline{\mathtt{T}}/\overline{\mathtt{X}}]\,\mathtt{T}_r} \rangle$ *by* **2.3**, **2.2**, **2**

Case 4 $\overline{\mathtt{U}} = \exists\emptyset\,.\,\mathtt{Z}, \overline{\mathtt{U}'} \wedge \mathtt{Z} \in \overline{\mathtt{Y}}$

1. $\overline{\mathtt{R}} = \mathtt{R}, \overline{\mathtt{R}'}$

2. $\langle \overline{\mathtt{R}_r}, \overline{\mathtt{T}_r} \rangle = \langle \overline{\mathtt{R}''},\ \overline{\mathtt{U}''} \rangle$

} *by def* sift

3.	$\overline{\texttt{[T/X]U}} = \texttt{[}\overline{\texttt{T/X}}\texttt{]}\exists\emptyset\texttt{.Z}, \overline{\texttt{[T/X]U}'}$	*by def subst*
4.	$\overline{\texttt{[T/X]U}} = \exists\emptyset\texttt{.Z}, \overline{\texttt{[T/X]U}'}$	*by* **3**, **b**
5.	$sift(\overline{\texttt{R}}, \overline{\texttt{[T/X]U}}, \overline{\texttt{Y}}) = \langle\overline{\texttt{R}''},\ \overline{\texttt{[T/X]U}''}\rangle$	*by* **4**, **1**, sift
6.	*done*	*by* **5**, **2**

□

Lemma 4 (Substitution on $\overline{\texttt{R}}$ preserves *sift*).

If:

a. $sift(\overline{\texttt{R}}, \overline{\texttt{U}}, \overline{\texttt{Y}}) = \langle\overline{\texttt{R}_r}, \overline{\texttt{T}_r}\rangle$

b. *f is a mapping from and to types in the syntactic category* $\texttt{R}$.

then:

$sift(\overline{f(\texttt{R})}, \overline{\texttt{U}}, \overline{\texttt{Y}}) = \langle\overline{f(\texttt{R}_r)}, \overline{\texttt{T}_r}\rangle$

Proof *by structural induction on the derivation of* $sift(\overline{\texttt{R}}, \overline{\texttt{U}}, \overline{\texttt{Y}}) = \langle\overline{\texttt{R}_r}, \overline{\texttt{T}_r}\rangle$ *with a case analysis on the last step:*

Case 1 $\overline{\texttt{U}} = \emptyset$

trivial

Case 2 $\overline{\texttt{U}} = \exists\Delta\texttt{.N}, \overline{\texttt{U}'}$

1.	$\overline{\texttt{R}} = \texttt{R}, \overline{\texttt{R}'}$	*by def* sift
2.	$(\overline{\texttt{R}_r}, \overline{\texttt{T}_r}) = (\texttt{R}, \overline{\texttt{R}''},\ \exists\Delta\texttt{.N}, \overline{\texttt{U}''})$	*by def* sift
3.	$sift(\overline{\texttt{R}'},\ \overline{\texttt{U}'},\ \overline{\texttt{Y}}) = \langle\overline{\texttt{R}''},\ \overline{\texttt{U}''}\rangle$	*by premise* sift
4.	$\overline{f(\texttt{R})} = f(\texttt{R}), \overline{f(\texttt{R}')}$	*by* **1**, **c**
5.	$sift(\overline{f(\texttt{R}')},\ \overline{\texttt{U}'},\ \overline{\texttt{Y}}) = \langle\overline{f(\texttt{R}'')},\ \overline{\texttt{U}''}\rangle$	*by* **3**, *ind hyp*
6.	$sift(\overline{f(\texttt{R})}, \overline{\texttt{U}}, \overline{\texttt{Y}}) =$ $\langle f(\texttt{R}), \overline{f(\texttt{R}'')},\ \exists\Delta\texttt{.N}, \overline{\texttt{U}''}\rangle$	*by* **5**, **4**, sift
7.	$sift(\overline{f(\texttt{R})}, \overline{\texttt{U}}, \overline{\texttt{Y}}) = \langle\overline{f(\texttt{R}_r)}, \overline{\texttt{T}_r}\rangle$	*by* **6**, **2**

Case 3 $\overline{\texttt{U}} = \exists\emptyset\texttt{.Z}, \overline{\texttt{U}'} \wedge \texttt{Z} \notin \overline{\texttt{Y}}$

1.	$\overline{\texttt{R}} = \texttt{R}, \overline{\texttt{R}'}$	*by def* sift
2.	$\langle\overline{\texttt{R}_r}, \overline{\texttt{T}_r}\rangle = \langle\texttt{R}, \overline{\texttt{R}''},\ \exists\emptyset\texttt{.Z}, \overline{\texttt{U}''}\rangle$	*by def* sift
3.	$sift(\overline{\texttt{R}'},\ \overline{\texttt{U}'},\ \overline{\texttt{Y}}) = \langle\overline{\texttt{R}''},\ \overline{\texttt{U}''}\rangle$	*by premise* sift
4.	$\overline{f(\texttt{R})} = f(\texttt{R}), \overline{f(\texttt{R}')}$	*by* **1**, **c**
5.	$sift(\overline{f(\texttt{R}')},\ \overline{\texttt{U}'},\ \overline{\texttt{Y}}) = \langle\overline{f(\texttt{R}'')},\ \overline{\texttt{U}''}\rangle$	*by* **3**, **b**, *ind hyp*
6.	$sift(\overline{f(\texttt{R})}, \overline{\texttt{U}}, \overline{\texttt{Y}}) =$ $\langle f(\texttt{R}), \overline{f(\texttt{R}'')},\ \exists\emptyset\texttt{.Z}, \overline{\texttt{U}''}\rangle$	*by* **5**, **4**, sift
7.	$sift(\overline{f(\texttt{R})}, \overline{\texttt{U}}, \overline{\texttt{Y}}) = \langle\overline{f(\texttt{R}_r)}, \overline{\texttt{T}_r}\rangle$	*by* **6**, **2**

Case 4 $\overline{\mathtt{U}} = \exists\emptyset\,.\,\mathtt{Z},\overline{\mathtt{U}'} \wedge \mathtt{Z} \in \overline{\mathtt{Y}}$

1.	$\overline{\mathtt{R}} = \mathtt{R},\overline{\mathtt{R}'}$	*by def* sift
2.	$\langle\overline{\mathtt{R}_r},\overline{\mathtt{T}_r}\rangle = \langle\overline{\mathtt{R}''},\ \overline{\mathtt{U}''}\rangle$	(by def sift)
3.	$\overline{f(\mathtt{R})} = f(\mathtt{R}),\overline{f(\mathtt{R}')}$	*by* **1**, **c**
4.	$sift(\overline{f(\mathtt{R})},\overline{\mathtt{U}},\overline{\mathtt{Y}}) = \langle\overline{f(\mathtt{R}'')},\ \overline{\mathtt{U}''}\rangle$	*by* **3**, sift
5.	*done*	*by* **4**, **2**

□

Lemma 5 (Substitution preserves field type).

If:

a. $fType(\mathtt{f},\mathtt{C}\texttt{<}\overline{\mathtt{U}}\texttt{>}) = \mathtt{U}$

then:

$fType(\mathtt{f},\,[\overline{\mathtt{T/X}}]\mathtt{C}\texttt{<}\overline{\mathtt{U}}\texttt{>}) = [\overline{\mathtt{T/X}}]\mathtt{U}$

Proof *by induction on the derivation of* $fType(\mathtt{f},\mathtt{C}\texttt{<}\overline{\mathtt{U}}\texttt{>}) = \mathtt{U}$ *with a case analysis on the last step:*

Case 1 *base case*

1.	$\mathtt{f} = \mathtt{f}_i$	*by def fType*
2.	$\mathtt{U} = [\overline{\mathtt{U/Y}}]\mathtt{U}_i^{'}$	(*by def fType*)
3.	$\mathtt{class}\ \mathtt{C}\texttt{<}\overline{\mathtt{Y}\triangleleft\ \mathtt{B}_u}\texttt{>}\ \triangleleft\ \mathtt{N}\ \{\overline{\mathtt{U}'\ \mathtt{f;}}\ \overline{\mathtt{M}}\}$	*by premise of fType*
4.	$fType(\mathtt{f}_i,\mathtt{C}\texttt{<}[\overline{\mathtt{T/X}}]\overline{\mathtt{U}}\texttt{>}) = [\overline{[\overline{\mathtt{T/X}}]\mathtt{U/Y}}]\mathtt{U}_i^{'}$	*by* **3**, *def fType*
5.	$\overline{\mathtt{Y}\triangleleft\ \mathtt{B}_u} \vdash \mathtt{U}_i'\ \text{OK}$	*by* **3**, *wf prog,* T-Class
6.	$[\overline{\mathtt{T/X}}]\mathtt{U}_i^{'} = \mathtt{U}_i'$	*by* **5**
7.	$fType(\mathtt{f}_i,\mathtt{C}\texttt{<}[\overline{\mathtt{T/X}}]\overline{\mathtt{U}}\texttt{>}) = [\overline{\mathtt{T/X}}]([\overline{\mathtt{U/Y}}]\mathtt{U}_i')$	*by* **4**, **6**, *def subst*
8.	$fType(\mathtt{f},\mathtt{C}\texttt{<}[\overline{\mathtt{T/X}}]\overline{\mathtt{U}}\texttt{>}) = [\overline{\mathtt{T/X}}]\mathtt{U}$	*by* **7**, **1**, **2**

Case 2 *inductive case*

1.	$\mathtt{U} = fType(\mathtt{f},\,[\overline{\mathtt{U/Y}}]\mathtt{N})$	*by def fType*
2.	$\mathtt{class}\ \mathtt{C}\texttt{<}\overline{\mathtt{Y}\triangleleft\ \mathtt{B}_u}\texttt{>}\ \triangleleft\ \mathtt{N}\ \{\overline{\mathtt{U}'\ \mathtt{f;}}\ \overline{\mathtt{M}}\}$	*by premises of fType*
3.	$\mathtt{f} \notin \overline{\mathtt{f}}$	(*by premises of fType*)
4.	$[\overline{\mathtt{T/X}}]\mathtt{U} = fType(\mathtt{f},\,[\overline{\mathtt{T/X}}]\,[\overline{\mathtt{U/Y}}]\mathtt{N})$	*by* **1**, *ind hyp*
5.	$fType(\mathtt{f},\mathtt{C}\texttt{<}[\overline{\mathtt{T/X}}]\overline{\mathtt{U}}\texttt{>}) =$ $fType(\mathtt{f},\,[\overline{[\overline{\mathtt{T/X}}]\mathtt{U/Y}}]\mathtt{N})$	*by* **2**, **3**, *def fType*
6.	$\overline{\mathtt{Y}\triangleleft\ \mathtt{B}_u} \vdash \mathtt{N}\ \text{OK}$	*by* **2**, *wf prog,* T-Class
7.	$[\overline{\mathtt{T/X}}]\mathtt{N} = \mathtt{N}$	*by* **6**
8.	$fType(\mathtt{f},\mathtt{C}\texttt{<}[\overline{\mathtt{T/X}}]\overline{\mathtt{U}}\texttt{>}) =$ $fType(\mathtt{f},\,[\overline{\mathtt{T/X}}]\,[\overline{\mathtt{U/Y}}]\mathtt{N})$	*by* **5**, **7**
9.	$fType(\mathtt{f},\mathtt{C}\texttt{<}[\overline{\mathtt{T/X}}]\overline{\mathtt{U}}\texttt{>}) = [\overline{\mathtt{T/X}}]\mathtt{U}$	*by* **8**, **4**

$\square$

Lemma 6 (Substitution preserves method type).

If:

a. $mType(\texttt{m}, \texttt{C<}\overline{\texttt{U}}\texttt{>}) = \texttt{<}\overline{\texttt{X}'\lhd\ \texttt{T}'}\texttt{>}\overline{\texttt{U}'} \rightarrow \texttt{U}$

then:

$mType(\texttt{m}, \texttt{C<}[\overline{\texttt{T/X}}]\overline{\texttt{U}}\texttt{>}) = [\overline{\texttt{T/X}}](\texttt{<}\overline{\texttt{X}'\lhd\ \texttt{T}'}\texttt{>}\overline{\texttt{U}'} \rightarrow \texttt{U})$

Proof *by induction on the derivation of* $mType(\texttt{m}, \texttt{C<}\overline{\texttt{U}}\texttt{>}) = \texttt{<}\overline{\texttt{X}'\lhd\ \texttt{T}'}\texttt{>}\overline{\texttt{U}'} \rightarrow \texttt{U}$ *with a case analysis on the last step:*

Case 1 *base case*

1.	$\texttt{<}\overline{\texttt{X}'\lhd\ \texttt{T}'}\texttt{>}\overline{\texttt{U}'} \rightarrow \texttt{U} = [\overline{\texttt{U/Y}}](\texttt{<}\overline{\texttt{X}''\lhd\ \texttt{T}'}\texttt{>}\overline{\texttt{U}''} \rightarrow \texttt{U}')$	*by def mType*
2.	$\texttt{class C<}\overline{\texttt{Y}\lhd\ \texttt{B}_u}\texttt{>}\ \lhd\ \texttt{N}\ \{\overline{\texttt{U}'\ \texttt{f;}}\ \overline{\texttt{M}}\}$	*by premises of mType*
3.	$\texttt{<}\overline{\texttt{X}''\lhd\ \texttt{T}'}\texttt{>U}'\ \texttt{m(}\overline{\texttt{U}''\ \texttt{x}}\texttt{)}\ \{\texttt{return e;}\} \in \overline{\texttt{M}}$	*by premises of mType*
4.	*let* $\texttt{S} = \texttt{<}\overline{\texttt{X}''\lhd\ \texttt{T}'}\texttt{>}\overline{\texttt{U}''} \rightarrow \texttt{U}'$	
5.	$mType(\texttt{m}, \texttt{C<}[\overline{\texttt{T/X}}]\overline{\texttt{U}}\texttt{>}) = [\overline{[\overline{\texttt{T/X}}]\texttt{U/Y}}]\texttt{S}$	*by* **2**, **3** *def mType*
6.	$\overline{\texttt{Y}\lhd\ \texttt{B}_u} \vdash \texttt{S}$ OK	*by* **2**, *wf prog,* T-Class
7.	$[\overline{\texttt{T/X}}]\texttt{S} = \texttt{S}$	*by* **6**
8.	$mType(\texttt{m}, \texttt{C<}[\overline{\texttt{T/X}}]\overline{\texttt{U}}\texttt{>}) = [\overline{\texttt{T/X}}]([\overline{\texttt{U/Y}}]\texttt{S})$	*by* **5**, **7**, *def subst*
9.	$mType(\texttt{m}, \texttt{C<}[\overline{\texttt{T/X}}]\overline{\texttt{U}}\texttt{>}) =$ $[\overline{\texttt{T/X}}](\texttt{<}\overline{\texttt{X}'\lhd\ \texttt{T}'}\texttt{>}\overline{\texttt{U}'} \rightarrow \texttt{U}))$	*by* **8**, **1**, **4**

Case 2 *inductive case*

1.	$\texttt{<}\overline{\texttt{X}'\lhd\ \texttt{T}'}\texttt{>}\overline{\texttt{U}'} \rightarrow \texttt{U} = mType(\texttt{m}, [\overline{\texttt{U/Y}}]\texttt{N})$	*by def mType*
2.	$\texttt{class C<}\overline{\texttt{Y}\lhd\ \texttt{B}_u}\texttt{>}\ \lhd\ \texttt{N}\ \{\overline{\texttt{U}'\ \texttt{f;}}\ \overline{\texttt{M}}\}$	*by premises of mType*
3.	$\texttt{m} \notin \overline{\texttt{M}}$	*by premises of mType*
4.	$[\overline{\texttt{T/X}}](\texttt{<}\overline{\texttt{X}'\lhd\ \texttt{T}'}\texttt{>}\overline{\texttt{U}'} \rightarrow \texttt{U}) =$ $mType(\texttt{m}, [\overline{\texttt{T/X}}][\overline{\texttt{U/Y}}]\texttt{N})$	*by* **1**, *ind hyp*
5.	$mType(\texttt{m}, \texttt{C<}[\overline{\texttt{T/X}}]\overline{\texttt{U}}\texttt{>}) =$ $mType(\texttt{m}, [\overline{[\overline{\texttt{T/X}}]\texttt{U/Y}}]\texttt{N})$	*by* **2**, **3**, *def mType*
6.	$\overline{\texttt{Y}\lhd\ \texttt{B}_u} \vdash \texttt{N}$ OK	*by* **2**, *wf prog,* T-Class
7.	$[\overline{\texttt{T/X}}]\texttt{N} = \texttt{N}$	*by* **6**
8.	$mType(\texttt{m}, \texttt{C<}[\overline{\texttt{T/X}}]\overline{\texttt{U}}\texttt{>}) =$ $mType(\texttt{m}, [\overline{\texttt{T/X}}][\overline{\texttt{U/Y}}]\texttt{N})$	*by* **5**, **7**
9.	$mType(\texttt{m}, \texttt{C<}[\overline{\texttt{T/X}}]\overline{\texttt{U}}\texttt{>}) =$ $[\overline{\texttt{T/X}}](\texttt{<}\overline{\texttt{X}'\lhd\ \texttt{T}'}\texttt{>}\overline{\texttt{U}'} \rightarrow \texttt{U})$	*by* **8**, **4**

$\square$

Lemma 7 (Weakening of *uBound*).

If:

a. $uBound_{\Delta,\Delta'}(\mathtt{B}) = \mathtt{B}'$

b. $dom(\Delta,\Delta') \cap dom(\Delta'') = \emptyset$

then:

$uBound_{\Delta,\Delta'',\Delta'}(\mathtt{B}) = \mathtt{B}'$

Proof *by structural induction on the derivation of* $uBound_{\Delta,\Delta'}(\mathtt{B}) = \mathtt{B}'$ *with a case analysis on the last step:*

Case 1 $\mathtt{B} = \exists\emptyset.\mathtt{X}$

1.	$uBound_{\Delta,\Delta'}(\mathtt{B}) = uBound_{\Delta,\Delta'}(\mathtt{B})_u$	*by def uBound* (1, 2)
2.	$\Delta,\Delta'(\mathtt{X}) = [\mathtt{B}_l\ \mathtt{B}_u]$	
3.	$\Delta,\Delta'',\Delta'(\mathtt{X}) = [\mathtt{B}_l\ \mathtt{B}_u]$	*by* **2**, **b**
4.	$uBound_{\Delta,\Delta'}(\mathtt{B})_u = uBound_{\Delta,\Delta'',\Delta'}(\mathtt{B})_u$	*by* **1**, **b**, *ind hyp*
5.	$uBound_{\Delta,\Delta'',\Delta'}(\mathtt{B}) = \mathtt{B}'$	*by* **3**, **4**

Case 2 *otherwise*

1.	$\mathtt{B}' = \mathtt{B}$	*by def uBound*

□

Lemma 8 (Weakening of subtyping).

If:

a. $dom(\Delta,\Delta') \cap dom(\Delta'') = \emptyset$

and if:

b. $\Delta,\Delta' \vdash \mathtt{B} \sqsubset: \mathtt{B}'$

then:

c. $\Delta,\Delta'',\Delta' \vdash \mathtt{B} \sqsubset: \mathtt{B}'$

and if:

d. $\Delta,\Delta' \vdash \mathtt{B} <: \mathtt{B}'$

then:

$\Delta,\Delta'',\Delta' \vdash \mathtt{B} <: \mathtt{B}'$

Proof *by structural induction on* $\Delta,\Delta' \vdash \mathtt{B} \ll \mathtt{B}'$ *where* $\Delta \vdash \mathtt{B} \ll \mathtt{B}'$ *is defined to hold if either* $\Delta \vdash \mathtt{B} \sqsubset: \mathtt{B}'$ *or* $\Delta \vdash \mathtt{B} <: \mathtt{B}'$ *holds. There is a case analysis on the last step:*

Case 1 (XS-Reflex, XS-Sub-Class, XS-Bottom, XS-Empty)

trivial

Case 2 (XS-Trans, S-SC, S-Trans)

easy, by ind hyp

Case 3 (XS-Env)

1.	$\mathtt{B} = \exists\Delta'''.[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{N}$	*by def* XS-Env
2.	$\mathtt{B}' = \exists\overline{\mathcal{X}\to[B_l\ \ B_u]}.\mathtt{N}$	
3.	$\Delta, \Delta', \Delta''' \vdash \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{B}_l <: \mathtt{T}}$	*by premises of* XS-Env
4.	$\Delta, \Delta', \Delta''' \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{B}_u}$	
5.	$dom(\Delta''') \cap fv(\exists\overline{\mathcal{X}\to[B_l\ \ B_u]}.\mathtt{N}) = \emptyset$	
6.	$fv(\overline{\mathtt{T}}) \subseteq dom(\Delta, \Delta', \Delta''')$	
7.	$dom(\Delta, \Delta') \cap dom(\Delta''') = \emptyset$	*by Barendregt convention*
8.	$dom(\Delta''') \cap dom(\Delta'') = \emptyset$	
9.	$dom(\Delta, \Delta', \Delta''') \cap dom(\Delta'') = \emptyset$	*by* **a**, **7**, **8**
10.	$\Delta, \Delta'', \Delta', \Delta''' \vdash \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{B}_l <: \mathtt{T}}$	*by* **3**, **9**, *ind hyp*
11.	$\Delta, \Delta'', \Delta', \Delta''' \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{B}_u}$	*by* **4**, **9**, *ind hyp*
12.	$fv(\overline{\mathtt{T}}) \subseteq dom(\Delta, \Delta'', \Delta', \Delta''')$	*by* **6**, *def* $\subseteq$
13.	$\Delta, \Delta'', \Delta' \vdash \mathtt{B} <: \mathtt{B}'$	*by* **1**, **2**, **5**, **10**, **11**, **12**, XS-Env

Case 4 (S-Bound)

1.	$(\Delta, \Delta')(\mathtt{X}) = [\mathtt{B}_l\ \ \mathtt{B}_u]$	*by premise of* S-Bound
2.	$(\Delta, \Delta'', \Delta')(\mathtt{X}) = [\mathtt{B}_l\ \ \mathtt{B}_u]$	*by* **1**, **a**
3.	*done*	*by* **2**, S-Bound

□

Lemma 9 (Weakening of well-formedness).

If:

a. $dom(\Delta, \Delta') \cap dom(\Delta'') = \emptyset$

b. $\Delta, \Delta' \vdash \psi$ OK

where:

c. $\psi = \Delta'''$ *or* B *or* R *or* $\star$

and if:

d. $\psi = \Delta'''$ *then* $dom(\Delta, \Delta', \Delta''') \cap dom(\Delta'') = \emptyset$

then:

$\Delta, \Delta'', \Delta' \vdash \psi$ OK

Proof. structural induction on the derivation of $\Delta, \Delta' \vdash \psi$ OK *with a case analysis on the last step:*

Case 1 (F-Bottom, F-Env-Empty, F-World)

Trivial

Case 2 (F-Var, F-Var-O)

easy by **a**

Case 3 (F-Exist)

1.	$\psi = \exists\Delta'''.\mathtt{N}$	*by def* F-Exist
2.	$\Delta, \Delta' \vdash \Delta'''$ OK	*by premises of* F-Exist
3.	$\Delta, \Delta', \Delta''' \vdash \mathtt{N}$ OK	
4.	$dom(\Delta, \Delta', \Delta''') \cap dom(\Delta'') = \emptyset$	*by* **a**, *Barendregt*
5.	$\Delta, \Delta'', \Delta' \vdash \Delta'''$ OK	*by* **2**, **4**, *ind hyp*
6.	$\Delta, \Delta'', \Delta', \Delta''' \vdash \mathtt{N}$ OK	*by* **3**, **4**, *ind hyp*
7.	$\Delta, \Delta'', \Delta' \vdash \exists\Delta'''.\mathtt{N}$ OK	*by* **5**, **6**, F-Exist

Case 4 (F-Class)

1.	$\psi = \mathtt{C}\texttt{<}\overline{\mathcal{T}}\texttt{>}$	*by def* F-Class
2.	$\overline{\mathcal{T}} = \overline{\mathtt{T}}, \overline{\tau}, \tau_o, \tau_t$	*by premises of* F-Class
3.	$\Delta, \Delta(\tau_t) = [\bot\ \ \mathcal{T}]$	
4.	$\mathtt{class\ C}\texttt{<}\overline{\mathtt{X}\lhd\ \tau_u}\texttt{>}\ldots$	
5.	$\Delta, \Delta' \vdash \overline{\mathtt{T}}, \overline{\tau}, \tau_o$ OK	
6.	$\Delta, \Delta' \vdash \overline{\mathcal{T} <: [\overline{\mathcal{T}/\mathtt{X}}]\mathcal{T}_u}$	
7.	$\forall\tau \in \overline{\tau}.\ \Delta, \Delta' \vdash \tau_o <: \tau$	
8.	$\Delta, \Delta'', \Delta(\tau_t) = [\bot\ \ \mathcal{T}]$	*by* **3**, **a**
9.	$\Delta, \Delta'', \Delta' \vdash \overline{\mathtt{T}}, \overline{\tau}, \tau_o$ OK	*by* **5**, **a**, *ind hyp*
10.	$\Delta, \Delta'', \Delta' \vdash \overline{\mathcal{T} <: [\overline{\mathcal{T}/\mathtt{X}}]\mathtt{T}_u}$	*by* **6**, **a**, *lemma 8*
11.	$\forall\tau \in \overline{\tau}.\ \Delta, \Delta'', \Delta' \vdash \tau_o <: \tau$	*by* **7**, **a**, *ind hyp*
12.	$\Delta, \Delta'', \Delta' \vdash \psi$ OK	*by* **2**, **1**, **8**, **4**, **9**, **10**, **11**, F-Class

Case 5 (F-Object)

1.	$\psi = \mathtt{Object}\texttt{<}\tau_o,\ \tau_t\texttt{>}$	*by def* F-Object
2.	$\Delta, \Delta' \vdash \tau_o$ OK	*by premises of* F-Object
3.	$\Delta, \Delta(\tau_t) = [\bot\ \ \mathcal{T}]$	
4.	$\Delta, \Delta'', \Delta(\tau_t) = [\bot\ \ \mathcal{T}]$	*by* **3**, **a**
5.	$\Delta, \Delta'', \Delta' \vdash \tau_o$ OK	*by* **2**, **a**, *ind hyp*
6.	$\Delta, \Delta'', \Delta' \vdash \psi$ OK	*by* **1**, **4**, **5**, F-Object

Case 6 (F-Env)

1.	$\psi = \mathcal{X}\to[B_l\ \ B_u], \Delta'''$	*by def* F-Env
2.	$\Delta, \Delta', \mathcal{X}\to[B_l\ \ B_u], \Delta''' \vdash \mathtt{B}_l, \mathtt{B}_u$ OK	*by premises of* F-Env
3.	$\Delta, \Delta' \vdash uBound_{\Delta,\Delta'}(\mathtt{B}_l) \sqsubset: uBound_{\Delta,\Delta'}(\mathtt{B}_u)$	
4.	$\Delta, \Delta' \vdash \mathtt{B}_l <: \mathtt{B}_u$	
5.	$\Delta, \Delta', \mathcal{X}\to[B_l\ \ B_u] \vdash \Delta'''$ OK	
6.	$\Delta, \Delta' \vdash_{\mathcal{X}} B_u$ SC	
7.	$\Delta, \Delta'', \Delta', \mathcal{X}\to[B_l\ \ B_u], \Delta''' \vdash \mathtt{B}_l, \mathtt{B}_u$ OK	*by* **2**, **d**, *ind hyp*
8.	$\Delta, \Delta'', \Delta' \vdash uBound_{\Delta,\Delta'}(\mathtt{B}_l) \sqsubset:$ $uBound_{\Delta,\Delta'}(\mathtt{B}_u)$	*by* **3**, **a**, *lemma 8*
9.	$\Delta, \Delta'', \Delta' \vdash uBound_{\Delta,\Delta'',\Delta'}(\mathtt{B}_l) \sqsubset:$	*by* **8**, **a**, *lemma 7*

$uBound_{\Delta,\Delta'',\Delta'}(\mathtt{B}_u)$

10.	$\Delta, \Delta'', \Delta' \vdash \mathtt{B}_l <: \mathtt{B}_u$	*by* **4**, **a**, *lemma 8*
11.	$\Delta, \Delta'', \Delta', \mathcal{X}{\rightarrow}[B_l \;\; B_u] \vdash \Delta'''$ OK	*by* **5**, **d**, *ind hyp*
12.	$\Delta, \Delta'', \Delta' \vdash_{\mathcal{X}} B_u$ SC	*by* **6**, **a**, *lemma 8*
13.	$\Delta, \Delta'', \Delta' \vdash \mathcal{X}{\rightarrow}[B_l \;\; B_u], \Delta'''$ OK	*by* **7**, **9**, **10**, **11**, F-ENV

□

Lemma 10 (Weakening of Typing).

If:

a. $dom(\Delta, \Delta') \cap dom(\Delta'') = \emptyset$
b. $dom(\Gamma, \Gamma'') \cap dom(\Gamma') = \emptyset$
c. $\Delta, \Delta'; \Gamma, \Gamma'' \vdash \mathtt{e} : \mathtt{T}$

then:

$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \mathtt{e} : \mathtt{T}$

Proof *by structural induction on the derivation of* $\Delta, \Delta'; \Gamma, \Gamma'' \vdash \mathtt{e} : \mathtt{T}$ *with a case analysis on the last step:*

Case 1 (T-NULL)

trivial

Case 2 (T-CAST)

1.	$\mathtt{e} = (T)e'$	*by def* T-CAST
2.	$\Delta, \Delta'; \Gamma, \Gamma'' \vdash e' : U$	*by premises of* T-CAST
3.	$\Delta, \Delta' \vdash T <: U$	*by premises of* T-CAST
4.	$\Delta, \Delta' \vdash T$ OK	*by premises of* T-CAST
5.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash e' : U$	*by* **2**, **a**, **b**, *ind hyp*
6.	$\Delta, \Delta'', \Delta' \vdash T <: U$	*by* **3**, **a**, **b**, *ind hyp*
7.	$\Delta, \Delta'', \Delta' \vdash T$ OK	*by* **4**, **a**, **b**, *ind hyp*
8.	*done*	*by* **1**, **5**, **6**, **7**, T-CAST

Case 3 (T-VAR)

1.	$\mathtt{e} = \mathtt{x}$	*by def* T-VAR
2.	$\mathtt{T} = (\Gamma, \Gamma'')(\mathtt{x})$	*by def* T-VAR
3.	$(\Gamma, \Gamma', \Gamma'')(\mathtt{x}) = \mathtt{T}$	*by* **2**, **b**
4.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \mathtt{x} : \mathtt{T}$	*by* **3**, T-VAR

Case 4 (T-NEW)

1.	$\mathtt{e} = \mathtt{new\ C<}\overline{\mathcal{T}}\mathtt{,}\ \mathcal{T}\mathtt{,}\ \star\ \mathtt{>}$	*by def* T-NEW
2.	$\mathtt{T} = \exists\mathtt{O}{\rightarrow}[\bot\;\; \mathcal{T}]\mathtt{.C<}\overline{\mathcal{T}}\mathtt{,}\ \mathcal{T}\mathtt{,}\ \mathtt{O>}$	*by def* T-NEW

3.	$\Delta, \Delta' \vdash \overline{\mathcal{T}}, \mathcal{T}$ OK	by premises of T-New
4.	$\Delta, \Delta' \vdash \exists \mathtt{O} \to [\bot\ \mathcal{T}]\mathtt{.C<}\overline{\mathcal{T}}\mathtt{,}\ \mathcal{T}\mathtt{,}\ \mathtt{O>}$ OK	
5.	$\Delta, \Delta'', \Delta' \vdash \overline{\mathcal{T}}, \mathcal{T}$ OK	by **3**, *lemma 9*
6.	$\Delta, \Delta'', \Delta' \vdash \exists \mathtt{O} \to [\bot\ \mathcal{T}]\mathtt{.C<}\overline{\mathcal{T}}\mathtt{,}\ \mathcal{T}\mathtt{,}\ \star\ \mathtt{>}$ OK	by **4**, *lemma 9*
7.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \mathtt{e} : \mathtt{T}$	by **5**, **6**, **1**, **2**, T-New

Case 5 (T-Field)

1.	$\mathtt{e} = \mathtt{e}'\mathtt{.f}$	by *def* T-Field
2.	$\mathtt{T} = \Downarrow_{\Delta'''} \mathtt{U}$	
3.	$\Delta, \Delta'; \Gamma, \Gamma'' \vdash \mathtt{e}' : \exists \Delta'''\mathtt{.N}$	by *premises of* T-Field
4.	$fType(\mathtt{f}, \mathtt{N}) = \mathtt{U}$	
5.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \mathtt{e}' : \exists \Delta'''\mathtt{.N}$	by **3**, **a**, **b**, *ind hyp*
6.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \mathtt{e} : \mathtt{T}$	by **5**, **4**, **2**, T-Field

Case 6 (T-Assign)

1.	$\mathtt{e} = \mathtt{e}'\mathtt{.f\ =\ e}''$	by *def* T-Assign
2.	$\Delta, \Delta'; \Gamma, \Gamma'' \vdash \mathtt{e}' : \exists \Delta'''\mathtt{.N}$	by *premises of* T-Assign
3.	$fType(\mathtt{f}, \mathtt{N}) = \mathtt{U}$	
4.	$\Delta, \Delta'; \Gamma, \Gamma'' \vdash \mathtt{e}'' : \mathtt{T}$	
5.	$\Delta, \Delta', \Delta''' \vdash \mathtt{T} <: \mathtt{U}$	
6.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \mathtt{e}' : \exists \Delta'''\mathtt{.N}$	by **2**, **a**, **b**, *ind hyp*
7.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \mathtt{e}'' : \mathtt{T}$	by **4**, **a**, **b**, *ind hyp*
8.	$\Delta, \Delta'', \Delta', \Delta''' \vdash \mathtt{T} <: \mathtt{U}$	by **5**, **a**, *lemma 8*
9.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \mathtt{e} : \mathtt{T}$	by **6**, **3**, **7**, **8**, **1**, T-Field

Case 7 (T-Subs)

1.	$\Delta, \Delta'; \Gamma, \Gamma'' \vdash \mathtt{e} : \mathtt{U}$	by *premises of* T-Subs
2.	$\Delta, \Delta' \vdash \mathtt{U} <: \mathtt{T}$	
3.	$\Delta, \Delta' \vdash \mathtt{T}$ OK	
4.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \mathtt{e} : \mathtt{U}$	by **1**, **a**, **b**, *ind hyp*
5.	$\Delta, \Delta'', \Delta' \vdash \mathtt{U} <: \mathtt{T}$	by **2**, **a**, *lemma 8*
6.	$\Delta, \Delta'', \Delta' \vdash \mathtt{T}$ OK	by **3**, **a**, *lemma 9*
7.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \mathtt{e} : \mathtt{T}$	by **4**, **5**, **6**, T-Subs

Case 8 (T-Invk)

1.	$\mathtt{e} = \mathtt{e}'\mathtt{.<}\overline{\mathtt{P}}\mathtt{>m(}\overline{\mathtt{e}}\mathtt{)}$	by *def* T-Invk
2.	$\Delta''' = \Delta'''', \overline{\Delta}$	
3.	$\mathtt{T} = \Downarrow_{\Delta'''', \overline{\Delta}} [\overline{\mathtt{T/Y}}]\mathtt{U}$	

4.	$\Delta, \Delta'; \Gamma, \Gamma'' \vdash \texttt{e}' : \exists\Delta''''.\texttt{N}$	*by premises of* T-INVK
5.	$mType(\texttt{m}, \texttt{N}) = \texttt{<}\overline{\texttt{Y}\lhd\ \texttt{B}}\texttt{>}\overline{\texttt{U}}\to\texttt{U}$	
6.	$\Delta, \Delta'; \Gamma, \Gamma'' \vdash \overline{\texttt{e} : \exists\Delta.\texttt{R}}$	
7.	$match(sift(\overline{\texttt{R}}, \overline{\texttt{U}}, \overline{\texttt{Y}}), \overline{\texttt{P}}, \overline{\texttt{Y}}, \overline{\texttt{T}})$	
8.	$\Delta, \Delta' \vdash \overline{\texttt{P}}$ OK	
9.	$\Delta, \Delta', \Delta'''', \overline{\Delta} \vdash \overline{\texttt{T} <: [\overline{\texttt{T/Y}}]\texttt{B}}$	
10.	$\Delta, \Delta', \Delta'''', \overline{\Delta} \vdash \overline{\texttt{R} <: [\overline{\texttt{T/Y}}]\texttt{U}}$	
11.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \texttt{e}' : \exists\Delta''''.\texttt{N}$	*by* **4**, **a**, **b**, *ind hyp*
12.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \overline{\texttt{e} : \exists\Delta.\texttt{R}}$	*by* **6**, **a**, **b**, *ind hyp*
13.	$\Delta, \Delta'', \Delta' \vdash \overline{\texttt{P}}$ OK	*by* **8**, **a**, *lemma 9*
14.	$\Delta, \Delta'', \Delta', \Delta'''', \overline{\Delta} \vdash \overline{\texttt{T} <: [\overline{\texttt{T/Y}}]\texttt{B}}$	*by* **9**, **a**, *lemma 8*
15.	$\Delta, \Delta'', \Delta', \Delta'''', \overline{\Delta} \vdash \overline{\texttt{R} <: [\overline{\texttt{T/Y}}]\texttt{U}}$	*by* **10**, **a**, *lemma 8*
16.	$\Delta, \Delta'', \Delta'; \Gamma, \Gamma', \Gamma'' \vdash \texttt{e} :\Downarrow_{\Delta'''', \overline{\Delta}} [\overline{\texttt{T/Y}}]\texttt{U}$	*by* **11**, **5**, **12**, **7**, **13**, **14**, **15**, T-INVK

□

Lemma 11 (Well-formed type environments are disjoint).

If:

a. $\Delta \vdash \Delta'$ OK

then:

$dom(\Delta) \cap dom(\Delta') = \emptyset$

Proof *by structural induction on the derivation of* $\Delta \vdash \Delta'$ OK *with a case analysis on the last step:*

Case 1 (F-ENV-EMPTY)

trivial

Case 2 (F-ENV)

1.	$\Delta' = \mathcal{X}\to[B_l\ \ B_u], \Delta'''$	*by def* F-ENV
2.	$\Delta, \mathcal{X}\to[B_l\ \ B_u] \vdash \Delta'''$ OK	*by premises of* F-ENV
3.	$\texttt{X} \notin dom(\Delta)$	*by* **2**, *def concatenation*
4.	$dom(\Delta, \mathcal{X}\to[B_l\ \ B_u]) \cap dom(\Delta''') = \emptyset$	*by* **2**, *ind hyp*
5.	$dom(\Delta) \cap dom(\Delta''') = \emptyset$	*by* **4**
6.	$dom(\Delta) \cap dom(\mathcal{X}\to[B_l\ \ B_u], \Delta''') = \emptyset$	*by* **5**,**3**
7.	$dom(\Delta) \cap dom(\Delta') = \emptyset$	*by* **6**,**1**

□

Lemma 12 (Extension of type environments preserves well-formedness).

If:

a. $\Delta \vdash \Delta'$ OK

b. $\Delta, \Delta' \vdash \Delta''$ OK

then:

$\Delta \vdash \Delta', \Delta''$ OK

Proof *by structural induction on the derivation of* $\Delta \vdash \Delta'$ OK *with a case analysis on the last step:*

Case 1 (F-Env-Empty)

1.	$\Delta' = \emptyset$	*by def* F-Env-Empty
2.	$\Delta \vdash \Delta''$ OK	*by* **1**, **b**
3.	$\Delta \vdash \Delta', \Delta''$ OK	*by* **1**, **2**

Case 2 (F-Env)

1.	$\Delta' = \mathcal{X} \rightarrow [B_l \;\; B_u], \Delta'''$	*by def* F-Env-Empty
2.	$\Delta, \mathcal{X} \rightarrow [B_l \;\; B_u], \Delta''' \vdash \mathtt{B}_l, \mathtt{B}_u$ OK	*by premises of* F-Env
3.	$\Delta \vdash uBound_{\Delta}(\mathtt{B}_l) \sqsubset: uBound_{\Delta}(\mathtt{B}_u)$	*by premises of* F-Env
4.	$\Delta \vdash \mathtt{B}_l <: \mathtt{B}_u$	*by premises of* F-Env
5.	$\Delta, \mathcal{X} \rightarrow [B_l \;\; B_u] \vdash \Delta'''$ OK	*by premises of* F-Env
6.	$dom(\Delta, \Delta') \cap dom(\Delta'') = \emptyset$	*by* **b**, *lemma 11*
7.	$\Delta, \mathcal{X} \rightarrow [B_l \;\; B_u], \Delta''', \Delta'' \vdash \mathtt{B}_l, \mathtt{B}_u$ OK	*by* **2**, **6**, *lemma 9*
8.	$\Delta, \mathcal{X} \rightarrow [B_l \;\; B_u] \vdash \Delta''', \Delta''$ OK	*by* **5**, **b**, **1**, *ind hyp*
9.	$\Delta \vdash \Delta', \Delta''$ OK	*by* **7**, **3**, **4**, **8**, *def* F-Env

□

Lemma 13 (Concatenation of type environments preserves well-formedness).

If:

a. $\Delta \vdash \Delta'$ OK

b. $\Delta \vdash \Delta''$ OK

c. $dom(\Delta') \cap dom(\Delta'') = \emptyset$

then:

$\Delta \vdash \Delta', \Delta''$ OK

Proof *by induction on the size of* Δ'

Case 1 $\Delta' = \emptyset$

trivial

Case 2 $\Delta' \neq \emptyset$

1. *let* $\Delta' = \mathcal{X} \rightarrow [B_l \;\; B_u], \Delta'''$

2.	$\Delta, \texttt{X}{\to}[\texttt{B}_l\ \texttt{B}_u], \Delta''' \vdash \texttt{B}_l$ OK	by **1**, **a**, *premises* F-ENV
3.	$\Delta, \texttt{X}{\to}[\texttt{B}_l\ \texttt{B}_u], \Delta''' \vdash \texttt{B}_u$ OK	
4.	$\Delta \vdash uBound_\Delta(\texttt{B}_l) \sqsubset: uBound_\Delta(\texttt{B}_u)$	
5.	$\Delta \vdash \texttt{B}_l <: \texttt{B}_u$	
6.	$\Delta, \texttt{X}{\to}[\texttt{B}_l\ \texttt{B}_u] \vdash \Delta'''$ OK	
7.	$\Delta, \texttt{X}{\to}[\texttt{B}_l\ \texttt{B}_u], \Delta''', \Delta'' \vdash \texttt{B}_l$ OK	*by* **2**, **c**, *lemma 9*
8.	$\Delta, \texttt{X}{\to}[\texttt{B}_l\ \texttt{B}_u], \Delta''', \Delta'' \vdash \texttt{B}_u$ OK	*by* **3**, **c**, *lemma 9*
9.	$dom(\Delta''') \cap dom(\Delta'') = \emptyset$	*by* **c**, **1**
10.	$\Delta, \texttt{X}{\to}[\texttt{B}_l\ \texttt{B}_u] \vdash \Delta''', \Delta''$ OK	*by* **6**, **b**, **9**, *ind hyp*
11.	$\Delta \vdash \mathcal{X}{\to}[B_l\ \ B_u], \Delta''', \Delta''$ OK	*by* **7**, **8**, **4**, **5**, **10**, F-ENV
12.	$\Delta \vdash \Delta', \Delta''$ OK	*by* **11**, **1**

□

Lemma 14 (Limited commutativity of substitution).

If:

a. $[\overline{\texttt{U}/\texttt{X}}]\,[\overline{\texttt{U}'/\texttt{X}'}]\,\texttt{T} = \texttt{T}'$

b. $\overline{\texttt{X}} \cap fv(\overline{\texttt{U}'}) = \emptyset$

c. $\overline{\texttt{X}'} \cap fv(\overline{\texttt{U}}) = \emptyset$

d. $\overline{\texttt{X}} \cap \overline{\texttt{X}'} = \emptyset$

then:

$[\overline{\texttt{U}'/\texttt{X}'}]\,[\overline{\texttt{U}/\texttt{X}}]\,\texttt{T} = \texttt{T}'$

Proof *by structural induction on the form of* T*:*

Case 1 $\texttt{T} = \exists\Delta.\texttt{R}$

1.	$\texttt{T}' = \exists[\overline{\texttt{U}/\texttt{X}}]\,[\overline{\texttt{U}'/\texttt{X}'}]\,\Delta.\,[\overline{\texttt{U}/\texttt{X}}]\,[\overline{\texttt{U}'/\texttt{X}'}]\texttt{R}$	*by def subst*
2.	$\texttt{T}' = \exists[\overline{\texttt{U}'/\texttt{X}'}]\,[\overline{\texttt{U}/\texttt{X}}]\,\Delta.\,[\overline{\texttt{U}'/\texttt{X}'}]\,[\overline{\texttt{U}/\texttt{X}}]\texttt{R}$	*by* **1**, **b**, **c**, **d**, *ind hyp*
3.	$\texttt{T}' = [\overline{\texttt{U}'/\texttt{X}'}]\,[\overline{\texttt{U}/\texttt{X}}]\,\texttt{T}$	*by* **2**, *def subst*

Case 2 $\texttt{T} = \texttt{C<}\overline{\texttt{T}}\texttt{>}$

1.	$\texttt{T}' = \texttt{C<}[\overline{\texttt{U}/\texttt{X}}]\,[\overline{\texttt{U}'/\texttt{X}'}]\overline{\texttt{T}}\texttt{>}$	*by def subst*
2.	$\texttt{T}' = \texttt{C<}[\overline{\texttt{U}'/\texttt{X}'}]\,[\overline{\texttt{U}/\texttt{X}}]\overline{\texttt{T}}\texttt{>}$	*by* **1**, **b**, **c**, **d**, *ind hyp*
3.	$\texttt{T}' = [\overline{\texttt{U}'/\texttt{X}'}]\,[\overline{\texttt{U}/\texttt{X}}]\,\texttt{T}$	*by* **2**, *def subst*

Case 3 $\texttt{T} = \texttt{Y}$

Case analysis on Y**:**

Case 1 $\texttt{Y} \in \overline{\texttt{X}}$

1.1. $[\overline{\mathtt{U'/X'}}]\mathtt{Y} = \mathtt{Y}$ *by def subst*
1.2. $[\overline{\mathtt{U/X}}][\overline{\mathtt{U'/X'}}]\mathtt{Y} = \mathtt{U}_i$ *by* **1.1**, *def subst*
1.3. $[\overline{\mathtt{U/X}}]\mathtt{Y} = \mathtt{U}_i$ *by def subst*
1.4. $[\overline{\mathtt{U'/X'}}][\overline{\mathtt{U/X}}]\mathtt{Y} = \mathtt{U}_i$ *by* **1.3**, **c**
1.5. *done with* $\mathtt{T'} = \mathtt{U}_i$ *by* **1.2**, **1.4**

Case 2 $\mathtt{Y} \in \overline{\mathtt{X'}}$

2.1. $[\overline{\mathtt{U'/X'}}]\mathtt{Y} = \mathtt{U}_i^{'}$ *by def subst*
2.2. $[\overline{\mathtt{U/X}}][\overline{\mathtt{U'/X'}}]\mathtt{Y} = \mathtt{U}_i^{'}$ *by* **2.1**, **b**
2.3. $[\overline{\mathtt{U/X}}]\mathtt{Y} = \mathtt{Y}$ *by def subst*
2.4. $[\overline{\mathtt{U'/X'}}][\overline{\mathtt{U/X}}]\mathtt{Y} = \mathtt{U}_i^{'}$ *by* **2.3**
2.5. *done with* $\mathtt{T'} = \mathtt{U}_i$ *by* **2.2**, **2.4**

Case 3 $\mathtt{Y} \notin (\overline{\mathtt{X}}, \overline{\mathtt{X'}})$

3.1. $\mathtt{T'} = \mathtt{Y}$ *by def subst*
3.2. $[\overline{\mathtt{U'/X'}}][\overline{\mathtt{U/X}}]\mathtt{T} = \mathtt{Y}$ *by* **b**, **c**, **d**

□

Lemma 15 (Subclassing preserves class type).

If:
a. $\vdash \mathtt{R} \sqsubset: \mathtt{N}$
then:
$\mathtt{R} = \mathtt{N'}$

Proof *by structural induction on the derivation of* $\vdash \mathtt{R} \sqsubset: \mathtt{N}$ *with a case analysis on the last step:*

Case 1 (SC-Sub-Class, SC-Reflex)

trivial

Case 2 (SC-Trans)

1. $\vdash \mathtt{R} \sqsubset: \mathtt{R'}$
2. $\vdash \mathtt{R'} \sqsubset: \mathtt{N}$
} *by premises of* SC-Trans

3. $\mathtt{R'} = \mathtt{N''}$ *by* **2**, *ind hyp*
4. $\mathtt{R} = \mathtt{N'}$ *by* **1**, **3**, *ind hyp*

□

Lemma 16 (*uBound* refines subtyping).

If:

a. $\vdash \Delta$ OK

and if:

b. $\Delta \vdash \mathtt{T} \sqsubset: \mathtt{T}'$

or:

c. $\Delta \vdash \mathtt{T} <: \mathtt{T}'$

then:

$\Delta \vdash uBound_\Delta(\mathtt{T}) \sqsubset: uBound_\Delta(\mathtt{T}')$

Proof *by structural induction on* $\Delta \vdash \mathtt{T} \ll \mathtt{T}'$ *where* $\Delta \vdash \mathtt{T} \ll \mathtt{T}'$ *is defined to hold if either* $\Delta \vdash \mathtt{T} \sqsubset: \mathtt{T}'$ *or* $\Delta \vdash \mathtt{T} <: \mathtt{T}'$ *holds. There is a case analysis on the last step:*

Case 1 (XS-Reflex)

trivial

Case 2 (XS-Sub-Class, XS-Env)

easy since $\mathtt{T} = \exists\Delta'.\mathtt{N}$ *and* $\mathtt{T}' = \exists\Delta''.\mathtt{N}'$ *and* $\forall\exists\Delta'.\mathtt{N}:\ uBound_\Delta(\exists\Delta'.\mathtt{N}) = \exists\Delta'.\mathtt{N}$

Case 3 (XS-Bottom)

N/A

Case 4 (S-SC)

easy, by ind hyp.

Case 5 S-Bound *upper bound*

1.	$\mathtt{T} = \exists\emptyset.\mathtt{X}$	} *by def* S-Bound
2.	$\mathtt{T}' = \mathtt{B}_u$	
3.	$\Delta(\mathtt{X}) = \mathcal{X}\to[B_l\ \ B_u]$	*by premise of* S-Bound
4.	$uBound_\Delta(\mathtt{X}) = uBound_\Delta(\mathtt{B}_u)$	*by def uBound,* **3**
5.	*done*	*by* **4**, XS-Reflex

Case 6 S-Bound *lower bound*

1.	$\mathtt{T} = \mathtt{B}_l$	} *by def* S-Bound
2.	$\mathtt{T}' = \exists\emptyset.\mathtt{X}$	
3.	$\Delta(\mathtt{X}) = \mathcal{X}\to[B_l\ \ B_u]$	*by premise of* S-Bound
4.	$uBound_\Delta(\mathtt{X}) = uBound_\Delta(\mathtt{B}_u)$	*by def uBound,* **3**
5.	$\Delta \vdash uBound_\Delta(\mathtt{B}_l) \sqsubset: uBound_\Delta(\mathtt{B}_u)$	*by* **3**, **a**, *def F-Env*
6.	*done*	*by* **5**, **4**, **2**, **1** SC-Reflex

Case 7 (XS-Trans)

1.	$\Delta \vdash \mathtt{T} \sqsubset: \mathtt{T}''$	} *by premises of* XS-Trans/S-Trans
2.	$\Delta \vdash \mathtt{T}'' \sqsubset: \mathtt{T}'$	

3.	$\Delta \vdash uBound_{\Delta}(\mathtt{T}) \sqsubset: uBound_{\Delta}(\mathtt{T}'')$	*by* **1**, **a**, *ind hyp*
4.	$\Delta \vdash uBound_{\Delta}(\mathtt{T}'') \sqsubset: uBound_{\Delta}(\mathtt{T}')$	*by* **2**, **a**, *ind hyp*
5.	$\Delta \vdash uBound_{\Delta}(\mathtt{T}) \sqsubset: uBound_{\Delta}(\mathtt{T}')$	*by* **3**, **4**, XS-Trans

Case 8 (S-Trans)

similar to case XS-Trans

Corollary *If* $\Delta \vdash \exists\Delta'.\mathtt{N} <: \exists\Delta''.\mathtt{N}'$ *and* $\vdash \Delta$ OK *then* $\Delta \vdash \exists\Delta'.\mathtt{N} \sqsubset: \exists\Delta''.\mathtt{N}'$.

□

Lemma 17 (Subsititution preserves subtyping).

If:

a.	$\Delta_1 \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T/X}}]\mathtt{B}_u}$
b.	$\Delta_1 \vdash \overline{[\overline{\mathtt{T/X}}]\mathtt{B}_l <: \mathtt{T}}$
c.	$\Delta = \Delta_1, \overline{\mathtt{X}\to[\mathtt{B}_l\ \mathtt{B}_u]}, \Delta_2$
d.	$\Delta' = \Delta_1, [\overline{\mathtt{T/X}}]\Delta_2$
e.	$\overline{\mathtt{X}} \cap fv(\Delta_1) = \emptyset$
f.	$fv(\overline{\mathtt{T}}) \subseteq dom(\Delta')$

and if:

g. $\Delta \vdash \mathtt{B} <: \mathtt{B}'$

then:

$\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B} <: [\overline{\mathtt{T/X}}]\mathtt{B}'$

and if:

h. $\Delta \vdash \mathtt{B} \sqsubset: \mathtt{B}'$

then:

$\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B} \sqsubset: [\overline{\mathtt{T/X}}]\mathtt{B}'$

Proof *by structural induction on* $\Delta \vdash \mathtt{B} \ll \mathtt{B}'$ *where* $\Delta \vdash \mathtt{B} \ll \mathtt{B}'$ *is defined to hold if either* $\Delta \vdash \mathtt{B} \sqsubset: \mathtt{B}'$ *or* $\Delta \vdash \mathtt{B} <: \mathtt{B}'$ *holds. There is a case analysis on the last step:*

Case 1 (XS-Reflex, XS-Bottom)

trivial

Case 2 (XS-Sub-Class)

1.	$\mathtt{B} = \exists\Delta''.\mathtt{C}\texttt{<}\overline{\mathtt{U}}\texttt{>}$	*by def* XS-Sub-Class
2.	$\mathtt{B}' = \exists\Delta''.[\overline{\mathtt{U/Y}}]\mathtt{N}$	*by def* XS-Sub-Class
3.	$\mathtt{class\ C}\texttt{<}\overline{\mathtt{Y} \lhd \mathtt{T}_u}\texttt{>} \lhd \mathtt{N}\ \{\ldots\}$	*by premise of* XS-Sub-Class
4.	$[\overline{\mathtt{T/X}}]\mathtt{B} = \exists[\overline{\mathtt{T/X}}]\Delta''.\mathtt{C}\texttt{<}[\overline{\mathtt{T/X}}]\overline{\mathtt{U}}\texttt{>}$	*by* **1**, *def subst*
5.	$[\overline{\mathtt{T/X}}]\mathtt{B}' = \exists[\overline{\mathtt{T/X}}]\Delta''.[\overline{\mathtt{T/X}}][\overline{\mathtt{U/Y}}]\mathtt{N}$	*by* **2**, *def subst*
6.	$\overline{\mathtt{Y}\to[\bot\ \mathtt{T}_u]} \vdash \mathtt{N}$ OK	*by* **3**, *wf prog*, T-Class
7.	$[\overline{\mathtt{T/X}}][\overline{\mathtt{U/Y}}]\mathtt{N} = [\overline{[\overline{\mathtt{T/X}}]\mathtt{U/Y}}]\mathtt{N}$	*by* **6**
8.	$[\overline{\mathtt{T/X}}]\mathtt{B}' = \exists[\overline{\mathtt{T/X}}]\Delta''.[\overline{[\overline{\mathtt{T/X}}]\mathtt{U/Y}}]\mathtt{N}$	*by* **5**, **7**
9.	$\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B} \sqsubset: [\overline{\mathtt{T/X}}]\mathtt{B}'$	*by* **3**, **4**, **8**, XS-Sub-Class

Case 3 (XS-Trans, S-SC, S-Trans)

easy, by ind hyp

Case 4 (XS-Env)

1.	$\mathtt{B} = \exists\Delta''.\,[\overline{\mathtt{U/Y}}]\mathtt{N}$	*by def* XS-Env
2.	$\mathtt{B}' = \exists\overline{\mathtt{Y}\rightarrow[\mathtt{B}'_l\ \mathtt{B}'_u]}.\mathtt{N}$	*by def* XS-Env
3.	$\Delta, \Delta'' \vdash \overline{[\overline{\mathtt{U/Y}}]\mathtt{B}'_l <: \mathtt{U}}$	*by premises of* XS-Env
4.	$\Delta, \Delta'' \vdash \overline{\mathtt{U} <: [\overline{\mathtt{U/Y}}]\mathtt{B}'_u}$	*by premises of* XS-Env
5.	$dom(\Delta'') \cap fv(\exists\overline{\mathtt{Y}\rightarrow[\mathtt{B}'_l\ \mathtt{B}'_u]}.\mathtt{N}) = \emptyset$	*by premises of* XS-Env
6.	$fv(\overline{\mathtt{U}}) \subseteq dom(\Delta, \Delta'')$	*by premises of* XS-Env
7.	$\Delta', [\overline{\mathtt{T/X}}]\Delta'' \vdash \overline{[\overline{\mathtt{T/X}}]\,[\overline{\mathtt{U/Y}}]\mathtt{B}'_l <: [\overline{\mathtt{T/X}}]\mathtt{U}}$	*by* **3**, **b**–**g**, *ind hyp*
8.	$\Delta', [\overline{\mathtt{T/X}}]\Delta'' \vdash \overline{[\overline{\mathtt{T/X}}]\mathtt{U} <: [\overline{\mathtt{T/X}}]\,[\overline{\mathtt{U/Y}}]\mathtt{B}'_u}$	*by* **4**, **b**–**g**, *ind hyp*
9.	$\overline{\mathtt{Y}} \cap fv(\overline{\mathtt{T}}) = \emptyset$	*by* **2**, *Barendregt's convention*
10.	$\overline{\mathtt{Y}} \cap \overline{\mathtt{X}} = \emptyset$	*by* **2**, *Barendregt's convention*
11.	$\overline{\mathtt{X}} \cap dom(\Delta'') = \emptyset$	*by* **1**, *Barendregt's convention*
12.	$\Delta', [\overline{\mathtt{T/X}}]\Delta'' \vdash \overline{[\overline{[\overline{\mathtt{T/X}}]\mathtt{U/Y}}]\,[\overline{\mathtt{T/X}}]\mathtt{B}'_l <: [\overline{\mathtt{T/X}}]\mathtt{U}}$	*by* **7**, **9**
13.	$\Delta', [\overline{\mathtt{T/X}}]\Delta'' \vdash \overline{[\overline{\mathtt{T/X}}]\mathtt{U} <: [\overline{[\overline{\mathtt{T/X}}]\mathtt{U/Y}}]\,[\overline{\mathtt{T/X}}]\mathtt{B}'_u}$	*by* **7**, **9**
14.	$fv(\overline{[\overline{\mathtt{T/X}}]\mathtt{U}}) \subseteq dom(\Delta', [\overline{\mathtt{T/X}}]\Delta'')$	*by* **6**, **f**
15.	$\Delta' \vdash \exists[\overline{\mathtt{T/X}}]\Delta''.\,[\overline{[\overline{\mathtt{T/X}}]\mathtt{U/Y}}]\,[\overline{\mathtt{T/X}}]\mathtt{N} \sqsubset:$ $\exists\overline{\mathtt{Y}\rightarrow[[\overline{\mathtt{T/X}}]\mathtt{B}'_l\ \ [\overline{\mathtt{T/X}}]\mathtt{B}'_u]}.\,[\overline{\mathtt{T/X}}]\mathtt{N}$	*by* **12**, **13**, **5**, **14**, XS-Env
16.	$\Delta' \vdash [\overline{\mathtt{T/X}}]\exists\Delta''.\,[\overline{\mathtt{U/Y}}]\mathtt{N} \sqsubset:$ $[\overline{\mathtt{T/X}}]\exists\overline{\mathtt{Y}\rightarrow[\mathtt{B}'_l\ \mathtt{B}'_u]}.\mathtt{N}$	*by* **15**, **9**, **10**, **11**, *def subst*
17.	$\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B} \sqsubset: [\overline{\mathtt{T/X}}]\mathtt{B}'$	*by* **16**, **1**, **2**

Case 5 S-Bound *lower bound*

1.	$\mathtt{B} = \mathtt{B}_l$	*by def* S-Bound
2.	$\mathtt{B}' = \mathtt{Y}$	*by def* S-Bound
3.	$\Delta(\mathtt{Y}) = [\mathtt{B}_l\ \mathtt{B}_u]$	*by premise of* S-Bound

Case analysis on $\mathtt{Y}$**:**

Case 1 $\mathtt{Y} \in dom(\Delta_1)$

1.1.	$\Delta'(\mathtt{Y}) = [\mathtt{B}_l\ \mathtt{B}_u]$	*by* **3**, **e**, $\mathtt{Y} \in dom(\Delta_1)$
1.2.	$\Delta' \vdash \mathtt{B}_l <: \mathtt{Y}$	*by* **1.1**, S-bound
1.3.	$[\overline{\mathtt{T/X}}]\mathtt{B} = \mathtt{B}_l$	*by* **1**, **3**, **e**, $\mathtt{Y} \in dom(\Delta_1)$
1.4.	$[\overline{\mathtt{T/X}}]\mathtt{B}' = \mathtt{Y}$	*by* **2**, **e**, $\mathtt{Y} \in dom(\Delta_1)$
1.5.	$\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B} <: [\overline{\mathtt{T/X}}]\mathtt{B}'$	*by* **1.2**, **1.4**, **1.3**

Case 2 $\mathtt{Y} \in dom(\Delta_2)$

2.1. $\Delta'(\mathtt{Y}) = \mathtt{[}[\overline{\mathtt{T/X}}]\mathtt{B}_l\ \ [\overline{\mathtt{T/X}}]\mathtt{B}_u\mathtt{]}$ *by* **3**, **e**, $\mathtt{Y} \in dom(\Delta_2)$
2.2. $\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B}_l <: \mathtt{Y}$ *by* **2.1**, S-BOUND
2.3. $[\overline{\mathtt{T/X}}]\mathtt{B} = [\overline{\mathtt{T/X}}]\mathtt{B}_l$ *by* **1**, **3**, $\mathtt{Y} \in dom(\Delta_2)$
2.4. $[\overline{\mathtt{T/X}}]\mathtt{B}' = \mathtt{Y}$ *by* **2**, **e**, $\mathtt{Y} \in dom(\Delta_2)$
2.5. $\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B} <: [\overline{\mathtt{T/X}}]\mathtt{B}'$ *by* **2.2**, **2.4**, **2.3**

Case 3 $\mathtt{Y} \in \overline{\mathtt{X}}$

3.1. *let* $\mathtt{Y} = \mathtt{X}_i$
3.2. $[\overline{\mathtt{T/X}}]\mathtt{B} = \mathtt{T}_i$ *by* **1**, **3.1**
3.3. $\Delta_1 \vdash [\overline{\mathtt{T/X}}]\mathtt{B}_l <: \mathtt{T}_i$ *by* **b**, **3**, **3.1**
3.4. $\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B}_l <: \mathtt{T}_i$ *by* **3.3**, **d**, *lemma 8*
3.5. $\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B} <: [\overline{\mathtt{T/X}}]\mathtt{B}'$ *by* **3.4**, **3.2**, **2**

Case 6 S-BOUND *upper bound*

1. $\mathtt{B} = \mathtt{Y}$
2. $\mathtt{B}' = \mathtt{B}_u$
} *by def* S-BOUND (1, 2)
3. $\Delta(\mathtt{Y}) = \mathtt{[B}_l\ \ \mathtt{B}_u\mathtt{]}$ *by premise of* S-BOUND

Case analysis on Y:

Case 1 $\mathtt{Y} \in dom(\Delta_1)$

1.1. $\Delta'(\mathtt{Y}) = \mathtt{[B}_l\ \ \mathtt{B}_u\mathtt{]}$ *by* **3**, **e**, $\mathtt{Y} \in dom(\Delta_1)$
1.2. $\Delta' \vdash \mathtt{Y} <: \mathtt{B}_u$ *by* **1.1**, S-BOUND
1.3. $[\overline{\mathtt{T/X}}]\mathtt{B} = \mathtt{Y}$ *by* **1**, **e**, $\mathtt{Y} \in dom(\Delta_1)$
1.4. $[\overline{\mathtt{T/X}}]\mathtt{B}' = \mathtt{B}_u$ *by* **2**, **3**, **e**, $\mathtt{Y} \in dom(\Delta_1)$
1.5. $\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B} <: [\overline{\mathtt{T/X}}]\mathtt{B}'$ *by* **1.2**, **1.3**, **1.4**

Case 2 $\mathtt{Y} \in dom(\Delta_2)$

2.1. $\Delta'(\mathtt{Y}) = \mathtt{[}[\overline{\mathtt{T/X}}]\mathtt{B}_l\ \ [\overline{\mathtt{T/X}}]\mathtt{B}_u\mathtt{]}$ *by* **3**, **e**, $\mathtt{Y} \in dom(\Delta_2)$
2.2. $\Delta' \vdash \mathtt{Y} <: [\overline{\mathtt{T/X}}]\mathtt{B}_u$ *by* **2.1**, S-BOUND
2.3. $[\overline{\mathtt{T/X}}]\mathtt{B} = \mathtt{Y}$ *by* **1**, **e**, $\mathtt{Y} \in dom(\Delta_2)$
2.4. $[\overline{\mathtt{T/X}}]\mathtt{B}' = [\overline{\mathtt{T/X}}]\mathtt{B}_u$ *by* **2**, **3**, $\mathtt{Y} \in dom(\Delta_2)$
2.5. $\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B} <: [\overline{\mathtt{T/X}}]\mathtt{B}'$ *by* **2.2**, **2.3**, **2.4**

Case 3 $\mathtt{Y} \in \overline{\mathtt{X}}$

3.1. *let* $\mathtt{Y} = \mathtt{X}_i$
3.2. $[\overline{\mathtt{T/X}}]\mathtt{B} = \mathtt{T}_i$ *by* **1**, **3.1**
3.3. $\Delta_1 \vdash \mathtt{T}_i <: [\overline{\mathtt{T/X}}]\mathtt{B}_u$ *by* **a**, **3**, **3.1**
3.4. $\Delta' \vdash \mathtt{T}_i <: [\overline{\mathtt{T/X}}]\mathtt{B}_u$ *by* **3.3**, **d**, *lemma 8*
3.5. $\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B} <: [\overline{\mathtt{T/X}}]\mathtt{B}'$ *by* **3.4**, **3.2**, **2**

□

Lemma 18 (Subsititution preserves well-formedness).

If:

a. $\Delta \vdash \psi$ OK

b. $\Delta_1 \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T/X}}]\mathtt{B}_u}$

c. $\Delta_1 \vdash \overline{[\overline{\mathtt{T/X}}]\mathtt{B}_l <: \mathtt{T}}$

d. $\Delta = \Delta_1, \overline{\mathtt{X} \to [\mathtt{B}_l\ \ \mathtt{B}_u]}, \Delta_2$

e. $\Delta' = \Delta_1, [\overline{\mathtt{T/X}}]\Delta_2$

f. $\overline{\mathtt{X}} \cap fv(\Delta_1) = \emptyset$

g. $\Delta_1 \vdash \overline{\mathtt{T}}$ OK

h. $\emptyset \vdash \Delta'$ OK

where:

i. $\psi ::= \Delta_p \mid \mathtt{B} \mid \mathtt{R} \mid \star$

then:

$\Delta' \vdash [\overline{\mathtt{T/X}}]\psi$ OK

Proof *by structural induction on the derivation of* $\Delta \vdash \psi$ OK *with a case analysis on the last step:*

Case 1 (F-Bottom, F-Env-Empty, F-Star, F-World)

trivial

Case 2 (F-Var, F-Var-O)

1.	$\psi = \mathtt{Y}$	*by def* F-Var
2.	$\mathtt{Y} \in dom(\Delta)$	*by premise of* F-Var

Case analysis on $\mathtt{Y}$**:**

Case 1 $\mathtt{Y} \in \overline{\mathtt{X}}$

1.1.	*let* $\mathtt{Y} = \mathtt{X}_i$	
1.2.	$[\overline{\mathtt{T/X}}]\psi = \mathtt{T}_i$	*by* **1.1**
1.3.	$\Delta' \vdash \mathtt{T}_i$ OK	*by* **g**, **d**, *lemma 9*
1.4.	*done*	*by* **1.2**, **1.3**

Case 2 $\mathtt{Y} \in dom(\Delta_1, \Delta_2)$

2.1.	$\mathtt{Y} \in dom\Delta'$	*by* **2**, $\mathtt{Y} \notin \overline{\mathtt{X}}$
2.2.	$\Delta' \vdash [\overline{\mathtt{T/X}}]\psi$ OK	*by* **2.1**, **1**

Case 3 (F-Exist)

1.	$\psi = \exists\Delta''.\texttt{N}$	*by def* F-Exist
2.	$\Delta \vdash \Delta''$ OK	*by premises of* F-Exist
3.	$\Delta, \Delta'' \vdash \texttt{N}$ OK	
4.	$\Delta' \vdash [\overline{\texttt{T/X}}]\Delta''$ OK	*by* **2**, **b**–**g**, *ind hyp*
5.	$\Delta'[\overline{\texttt{T/X}}]\Delta'' \vdash [\overline{\texttt{T/X}}]\texttt{N}$ OK	*by* **3**, **b**–**h**, *ind hyp*
6.	$dom(\Delta'') \cap \overline{\texttt{T}} = \emptyset$	*by* **h**, *Barendregt*
7.	$\Delta' \vdash [\overline{\texttt{T/X}}]\psi$ OK	*by* **4**, **5**, **6**, F-Exist, **1**

Case 4 (F-Class)

1.	$\psi = \texttt{C<}\overline{\mathcal{T}}\texttt{>}$	*by def* F-Class
2.	$\overline{\mathcal{T}} = \overline{T}, \overline{\tau}, \tau_o, \tau_t$	*by premises of* F-Class
3.	$\Delta(\tau_t) = [\bot\ \ \mathcal{T}]$	
4.	$\texttt{class C<}\overline{\texttt{Y} \lhd \mathcal{T}_u}\texttt{>...}$	
5.	$\Delta \vdash \overline{\texttt{T}}, \overline{\tau}, \tau_o$ OK	
6.	$\Delta \vdash \overline{\mathcal{T} <: [\overline{\mathcal{T}/\texttt{X}}]\mathcal{T}_u}$	
7.	$\forall\tau \in \overline{\tau}.\ \Delta \vdash \tau_o <: \tau$	
8.	$\Delta'([\overline{\texttt{T/X}}]\tau_t) = [\bot\ \ [\overline{\texttt{T/X}}]\mathcal{T}]$	*by* **3**, **1**
9.	$\Delta' \vdash \overline{[\overline{\texttt{T/X}}]\overline{T}, \overline{\tau}, \tau_o}$ OK	*by* **5**, **b**–**h**, *ind hyp*
10.	$\Delta' \vdash \overline{[\overline{\texttt{T/X}}]\mathcal{T} <: [[\overline{\texttt{T/X}}]\overline{\mathcal{T}}/\overline{\texttt{X}}][\overline{\texttt{T/X}}]\mathcal{T}_u}$	*by* **6**, **b**–**g**, *lemma 17*
11.	$\overline{\texttt{Y}\to[\texttt{...}]} \vdash \overline{\mathcal{T}_u}$ OK	*by* **4**, *wf prog*, T-Class
12.	$\Delta' \vdash \overline{[\overline{\texttt{T/X}}]\mathcal{T} <: [[\overline{\texttt{T/X}}]\overline{\mathcal{T}}/\overline{\texttt{X}}]\mathcal{T}_u}$	*by* **10**, **11**
13.	$\forall\tau' \in [\overline{\texttt{T/X}}]\overline{\tau}.\ \Delta' \vdash [\overline{\texttt{T/X}}]\tau_o <: \tau'$	*by* **7**, **b**–**h**, *ind hyp*
14.	$\Delta' \vdash [\overline{\texttt{T/X}}]\psi$ OK	*by* **8**, **4**, **9**, **12**, **13**, F-Class, **1**

Case 5 (F-Object)

1.	$\psi = \texttt{Object<}\tau_o\texttt{,}\ \tau_t\texttt{>}$	*by def* F-Object
2.	$\Delta \vdash \tau_o$ OK	*by premises of* F-Object
3.	$\Delta(\tau_t) = [\bot\ \ \mathcal{T}]$	
4.	$\Delta'([\overline{\texttt{T/X}}]\tau_t) = [\bot\ \ [\overline{\texttt{T/X}}]\mathcal{T}]$	*by* **3**, **1**
5.	$\Delta' \vdash \overline{[\overline{\texttt{T/X}}]\texttt{T}}, [\overline{\texttt{T/X}}]\tau_o$ OK	*by* **2**, **b**–**h**, *ind hyp*
6.	$\Delta' \vdash [\overline{\texttt{T/X}}]\psi$ OK	*by* **4**, **5**, F-Object, **1**

Case 6 (F-Env)

1.	$\psi = \texttt{Y}\to[\texttt{B}_l\ \ \texttt{B}_u], \Delta''$	*by def* F-Env
2.	$\Delta, \texttt{Y}\to[\texttt{B}_l\ \ \texttt{B}_u], \Delta'' \vdash \texttt{B}_l, \texttt{B}_u$ OK	*by premises of* F-Env
3.	$\Delta \vdash uBound_\Delta(\texttt{B}_l) \sqsubset: uBound_\Delta(\texttt{B}_u)$	
4.	$\Delta \vdash \texttt{B}_l <: \texttt{B}_u$	
5.	$\Delta, \texttt{Y}\to[\texttt{B}_l\ \ \texttt{B}_u] \vdash \Delta''$ OK	
6.	$\Delta \vdash_{\mathcal{X}} B_u$ SC	

7.	$\Delta', \mathtt{Y}{\rightarrow}[\,[\overline{\mathtt{T/X}}]\mathtt{B}_l\ \ [\overline{\mathtt{T/X}}]\mathtt{B}_u\,], [\overline{\mathtt{T/X}}]\Delta'' \vdash$ $[\overline{\mathtt{T/X}}]\mathtt{B}_l, [\overline{\mathtt{T/X}}]\mathtt{B}_u$ OK	*by* **2**, **b**–**h**, *ind hyp*
8.	$\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{B}_l <: [\overline{\mathtt{T/X}}]\mathtt{B}_u$	*by* **4**, **b**–**g**, *lemma 17*
9.	$\Delta' \vdash uBound_{\Delta'}([\overline{\mathtt{T/X}}]\mathtt{B}_l) \sqsubset: uBound_{\Delta'}([\overline{\mathtt{T/X}}]\mathtt{B}_u)$	*by* **8**, **h**, *lemma 16*
10.	$\Delta', \mathtt{Y}{\rightarrow}[\,[\overline{\mathtt{T/X}}]\mathtt{B}_l\ \ [\overline{\mathtt{T/X}}]\mathtt{B}_u\,] \vdash [\overline{\mathtt{T/X}}]\Delta''$ OK	*by* **5**, **b**–**h**, *ind hyp*
11.	$\Delta \vdash_{\mathcal{X}} [\overline{\mathtt{T/X}}]B_u$ SC	*by* **6**, **b**–**g**, *lemma 17*
12.	$\Delta' \vdash [\overline{\mathtt{T/X}}]\psi$ OK	*by* **7**, **9**, **8**, **10**, F-Env, **1**

□

Lemma 19 (Corrolorary to lemma 18).

If:

- **a.** $\Delta \vdash \psi$ OK
- **b.** $\Delta_1 \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T/X}}]\mathtt{B}_u}$
- **c.** $\Delta_1 \vdash \overline{[\overline{\mathtt{T/X}}]\mathtt{B}_l <: \mathtt{T}}$
- **d.** $\Delta = \Delta_1, \overline{\mathtt{X}{\rightarrow}[\mathtt{B}_l\ \ \mathtt{B}_u]}, \Delta_2$
- **e.** $\Delta' = \Delta_1, [\overline{\mathtt{T/X}}]\Delta_2$
- **f.** $\overline{\mathtt{X}} \cap fv(\Delta_1) = \emptyset$
- **g.** $\Delta_1 \vdash \overline{\mathtt{T}}$ OK
- **h.** $\emptyset \vdash \Delta_1$ OK
- **i.** $\Delta_1, \overline{\mathtt{X}{\rightarrow}[\mathtt{B}_l\ \ \mathtt{B}_u]} \vdash \Delta_2$ OK

where:

- **j.** $\psi ::= \Delta_p \mid \mathtt{B} \mid \mathtt{R} \mid \star$

then:

$\Delta' \vdash [\overline{\mathtt{T/X}}]\psi$ OK

Proof

1.	$\Delta_1 \vdash [\overline{\mathtt{T/X}}]\Delta_2$ OK	*by* **b**–**i**, *lemma 18*
2.	$\emptyset \vdash \Delta_1, [\overline{\mathtt{T/X}}]\Delta_2$ OK	*by* **h**, **1**, *lemma 12*
3.	$\Delta' \vdash [\overline{\mathtt{T/X}}]\psi$ OK	*by* **a**–**g**, **2**, *lemma 18*

□

Lemma 20 (Substitution preserves *close*).

If:

- **a.** $\Downarrow_{\Delta} \mathtt{T} = \mathtt{U}$
- **b.** $fv(\overline{\mathtt{T}}) \cap dom(\Delta) = \emptyset$
- **c.** $\overline{\mathtt{X}} \cap dom(\Delta) = \emptyset$

then:

$\Downarrow_{[\overline{\mathtt{T/X}}]\Delta} [\overline{\mathtt{T/X}}]\mathtt{T} = [\overline{\mathtt{T/X}}]\mathtt{U}$

Proof *by structural induction on the derivation of* $\Downarrow_{\Delta} \mathtt{T}$ *with a case analysis on the last step:*

Case 1

1. $\mathtt{T} = \exists\emptyset.\mathtt{X}$
2. $\mathtt{U} = \exists\emptyset.\mathtt{X}$
3. $\mathtt{X} \notin dom(\Delta)$

} *by def* close

Case analysis on X:

Case 1 $\mathtt{X} \notin \overline{\mathtt{X}}$

1.1.	$[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{U} = \mathtt{U}$	*by* **2**, *def subcase*
1.2.	$\Downarrow_{[\overline{\mathtt{T}/\mathtt{X}}]\Delta}\ [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{T} = \Downarrow_{[\overline{\mathtt{T}/\mathtt{X}}]\Delta}\ \mathtt{T}$	*by* **1**, *def subcase*
1.3.	$\Downarrow_{[\overline{\mathtt{T}/\mathtt{X}}]\Delta}\ \mathtt{T} = \Downarrow_{\Delta}\ \mathtt{T}$	*by* **1.2**, *def* close
1.4.	*done*	*by* **1.1**, **1.3**, **a**

Case 2 $\mathtt{X} = \mathtt{X}_1$

2.1.	$[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{T} = \mathtt{T}_i$	*by def subcase*, **1**
2.2.	$[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{U} = \mathtt{T}_i$	*by def subcase*, **2**
2.3.	$\Downarrow_{\Delta}\ \mathtt{T}_i = \mathtt{T}_i$	*by* **b**, *def* close
2.4.	$\Downarrow_{[\overline{\mathtt{T}/\mathtt{X}}]\Delta}\ \mathtt{T}_i = \mathtt{T}_i$	*by* **2.3**, *irrelevance of range of Δ in cases*
2.5.	*done*	*by* **2.4**, **2.1**, **2.2**

Case 2

1. $\mathtt{T} = \exists\emptyset.\mathtt{X}$
2. $\mathtt{U} = \Downarrow_{\Delta}\ \mathtt{B}_u$
3. $\Delta(\mathtt{X}) = [\mathtt{B}_l\ \ \mathtt{B}_u]$

} *by def* close

4.	$\mathtt{X} \notin \overline{\mathtt{X}}$	*by* **c**, **3**
5.	$[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{T} = \mathtt{T}$	*by* **1**, **4**
6.	$\Downarrow_{\Delta}\ [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{T} = \Downarrow_{\Delta}\ \mathtt{T}$	*by* **5**
7.	$\Downarrow_{\Delta}\ [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{T} = \Downarrow_{\Delta}\ \mathtt{B}_u$	*by* **6**, **2**, **a**
8.	$\Downarrow_{[\overline{\mathtt{T}/\mathtt{X}}]\Delta}\ [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{T} = \Downarrow_{[\overline{\mathtt{T}/\mathtt{X}}]\Delta}\ [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{B}_u$	*by* **7**, **3**
9.	$\Downarrow_{[\overline{\mathtt{T}/\mathtt{X}}]\Delta}\ [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{T} = [\overline{\mathtt{T}/\mathtt{X}}]\ \Downarrow_{\Delta}\ \mathtt{B}_u$	*by* **8**, **b**, **c**, *ind hyp*
10.	*done*	*by* **9**, **2**

Case 3

1. $\mathtt{T} = \exists\Delta'.\mathtt{N}$
2. $\mathtt{U} = \exists\Delta,\Delta'.\mathtt{N}$

} *by def* close

3.	$[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{T} = \exists[\overline{\mathtt{T}/\mathtt{X}}]\Delta'.[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{N}$	*by* **1**
4.	$\Downarrow_{[\overline{\mathtt{T}/\mathtt{X}}]\Delta}\ [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{T} = \exists[\overline{\mathtt{T}/\mathtt{X}}]\Delta, [\overline{\mathtt{T}/\mathtt{X}}]\Delta'.[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{N}$	*by* **3**, *def* close
5.	$\Downarrow_{[\overline{\mathtt{T}/\mathtt{X}}]\Delta}\ [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{T} = [\overline{\mathtt{T}/\mathtt{X}}]\exists\Delta,\Delta'.\mathtt{N}$	*by* **4**
6.	*done*	*by* **5**, **2**

□

Lemma 21 (Subsititution preserves typing).

If:

a.	$\Delta;\Gamma \vdash \mathtt{e} : \mathtt{T}$
b.	$\Delta_1 \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T/X}}]\mathtt{B}_u}$
c.	$\Delta_1 \vdash \overline{[\overline{\mathtt{T/X}}]\mathtt{B}_l <: \mathtt{T}}$
d.	$\Delta = \Delta_1, \overline{\mathtt{X}\rightarrow[\mathtt{B}_l\ \ \mathtt{B}_u]}, \Delta_2$
e.	$\Delta' = \Delta_1, [\overline{\mathtt{T/X}}]\Delta_2$
f.	$\overline{\mathtt{X}} \cap fv(\Delta_1) = \emptyset$
g.	$\Delta_1 \vdash \overline{\mathtt{T}}$ OK
h.	$\emptyset \vdash \Delta_1$ OK
i.	$\Delta_1, \overline{\mathtt{X}\rightarrow[\mathtt{B}_l\ \ \mathtt{B}_u]} \vdash \Delta_2$ OK

then:

$\Delta'; [\overline{\mathtt{T/X}}]\Gamma \vdash [\overline{\mathtt{T/X}}]\mathtt{e} : [\overline{\mathtt{T/X}}]\mathtt{T}$

Proof *by structural induction on the derivation of* $\Delta;\Gamma \vdash \mathtt{e} : \mathtt{T}$ *with a case analysis on the last step:*

Case 1 (T-VAR)

1.	$\mathtt{e} = \gamma$	*by def* T-VAR
2.	$\mathtt{T} = \Gamma(\gamma)$	
3.	$[\overline{\mathtt{T/X}}]\mathtt{e} = \gamma$	*by* **1**
4.	$([\overline{\mathtt{T/X}}]\Gamma)(\gamma) = [\overline{\mathtt{T/X}}](\Gamma(\gamma))$	*by def subst*
5.	$\Delta'; [\overline{\mathtt{T/X}}]\Gamma \vdash [\overline{\mathtt{T/X}}]\mathtt{e} : [\overline{\mathtt{T/X}}]\mathtt{T}$	*by* **3**, **4**, T-VAR, **1**, **2**

Case 2 (T-NULL)

by lemma 19

Case 3 (T-CAST)

1.	$e = (T)e'$	*by def* T-CAST
2.	$\Delta;\Gamma \vdash \mathtt{e}' : \mathtt{U}$	*by premises of* T-CAST
3.	$\Delta \vdash T <: U$	
4.	$\Delta \vdash \mathtt{T}$ OK	
5.	$\Delta'; [\overline{\mathtt{T/X}}]\Gamma \vdash [\overline{\mathtt{T/X}}]\mathtt{e}' : [\overline{\mathtt{T/X}}]\mathtt{U}$	*by* **2**, **b**–**i**, *ind hyp*
6.	$\Delta \vdash [\overline{\mathtt{T/X}}]\mathtt{T} <: [\overline{\mathtt{T/X}}]\mathtt{U}$	*by* **3**, **b**–**g**, *lemma 17*
7.	$\Delta' \vdash [\overline{\mathtt{T/X}}]\mathtt{T}$ OK	*by* **4**,**b**–**i**,*lemma 19*
8.	*done*	*by* **5**, **6**, **7**, T-CAST

Case 4 (T-NEW)

1.	$\mathtt{e} = \mathtt{new\ C}\texttt{<}\overline{\mathcal{T}},\ \mathcal{T},\ \star\ \texttt{>}$	*by def* T-NEW
2.	$\mathtt{T} = \exists \mathtt{O}\rightarrow[\bot\ \ \mathcal{T}].\mathtt{C}\texttt{<}\overline{\mathcal{T}},\ \mathcal{T},\ \mathtt{O}\texttt{>}$	
3.	$\Delta \vdash \overline{\mathcal{T}}, \mathcal{T}$ OK	*by premise of* T-NEW
4.	$\Delta \vdash \exists \mathtt{O}\rightarrow[\bot\ \ \mathcal{T}].\mathtt{C}\texttt{<}\overline{\mathcal{T}},\ \mathcal{T},\ \mathtt{O}\texttt{>}$ OK	

5.	$\Delta' \vdash [\overline{\texttt{T/X}}]\overline{\mathcal{T}}, [\overline{\texttt{T/X}}]\mathcal{T}$ OK	*by* **3**, **b**–**i**, *lemma 19*
6.	$\Delta' \vdash [\overline{\texttt{T/X}}]\exists\texttt{O}\rightarrow[\bot\ \mathcal{T}]\texttt{.C<}\overline{\mathcal{T}}\texttt{,}\ \mathcal{T}\texttt{,}\ \texttt{O>}$ OK	*by* **4**, **b**–**i**, *lemma 19*
7.	*done*	*by* **5**, **6**, T-New

Case 5 (T-Field)

1.	$\texttt{e} = \texttt{e}'\texttt{.f}$	*by def* T-Field
2.	$\texttt{T} = \Downarrow_{\Delta''} \texttt{U}$	*by def* T-Field
3.	$\Delta; \Gamma \vdash \texttt{e}' : \exists\Delta''\texttt{.N}$	*by premises of* T-Field
4.	$fType(\texttt{f}, \texttt{N}) = \texttt{U}$	*by premises of* T-Field
5.	$\Delta'; [\overline{\texttt{T/X}}]\Gamma \vdash [\overline{\texttt{T/X}}]\texttt{e}' : [\overline{\texttt{T/X}}]\exists\Delta''\texttt{.N}$	*by* **3**, **b**–**i**, *ind hyp*
6.	$\Delta'; [\overline{\texttt{T/X}}]\Gamma \vdash [\overline{\texttt{T/X}}]\texttt{e}' : \exists[\overline{\texttt{T/X}}]\Delta''\texttt{.}[\overline{\texttt{T/X}}]\texttt{N}$	*by* **5**, *Barendregt*
7.	$fType(\texttt{f}, [\overline{\texttt{T/X}}]\texttt{N}) = [\overline{\texttt{T/X}}]\texttt{U}$	*by* **4**, *lemma 5*
8.	$\Delta'; [\overline{\texttt{T/X}}]\Gamma \vdash [\overline{\texttt{T/X}}]\texttt{e} : \Downarrow_{[\overline{\texttt{T/X}}]\Delta''} [\overline{\texttt{T/X}}]\texttt{U}$	*by* **1**, **6**, **7**, T-Field
9.	$\Delta'; [\overline{\texttt{T/X}}]\Gamma \vdash [\overline{\texttt{T/X}}]\texttt{e} : [\overline{\texttt{T/X}}] \Downarrow_{\Delta''} \texttt{U}$	*by* **8**, *lemma 20*, **g**, *Barendregt*
10.	*done*	*by* **9**, **2**

Case 6 (T-Assign)

1.	$\texttt{e} = \texttt{e}'\texttt{.f = e}''$	*by def* T-Assign
2.	$\Delta; \Gamma \vdash \texttt{e}' : \exists\Delta''\texttt{.N}$	*by premises of* T-Assign
3.	$fType(\texttt{f}, \texttt{N}) = \texttt{U}$	*by premises of* T-Assign
4.	$\Delta; \Gamma \vdash \texttt{e}'' : \texttt{T}$	*by premises of* T-Assign
5.	$\Delta, \Delta'' \vdash \texttt{T} <: \texttt{U}$	*by premises of* T-Assign
6.	$\Delta'; [\overline{\texttt{T/X}}]\Gamma \vdash [\overline{\texttt{T/X}}]\texttt{e}' : [\overline{\texttt{T/X}}]\exists\Delta''\texttt{.N}$	*by* **2**, **b**–**i**, *ind hyp*
7.	$\Delta'; [\overline{\texttt{T/X}}]\Gamma \vdash [\overline{\texttt{T/X}}]\texttt{e}' : \exists[\overline{\texttt{T/X}}]\Delta''\texttt{.}[\overline{\texttt{T/X}}]\texttt{N}$	*by* **6**, *Barendregt*
8.	$fType(\texttt{f}, [\overline{\texttt{T/X}}]\texttt{N}) = [\overline{\texttt{T/X}}]\texttt{U}$	*by* **3**, *lemma 5*
9.	$\Delta'; [\overline{\texttt{T/X}}]\Gamma \vdash [\overline{\texttt{T/X}}]\texttt{e}'' : [\overline{\texttt{T/X}}]\texttt{T}$	*by* **4**, **b**–**i**, *ind hyp*
10.	$\Delta', [\overline{\texttt{T/X}}]\Delta''; [\overline{\texttt{T/X}}]\Gamma \vdash [\overline{\texttt{T/X}}]\texttt{T} <: [\overline{\texttt{T/X}}]\texttt{U}$	*by* **5**, **b**–**i**, *ind hyp*
11.	$\Delta'; [\overline{\texttt{T/X}}]\Gamma \vdash [\overline{\texttt{T/X}}]\texttt{e} : [\overline{\texttt{T/X}}]\texttt{T}$	*by* **1**, **7**, **8**, **9**, **10**, T-Assign

Case 7 (T-Subs)

1.	$\Delta; \Gamma \vdash \texttt{e} : \texttt{U}$	*by premises of* T-Subs
2.	$\Delta \vdash \texttt{U} <: \texttt{T}$	*by premises of* T-Subs
3.	$\Delta \vdash \texttt{T}$ OK	*by premises of* T-Subs
4.	$\Delta'; [\overline{\texttt{T/X}}]\Gamma \vdash [\overline{\texttt{T/X}}]\texttt{e} : [\overline{\texttt{T/X}}]\texttt{U}$	*by* **1**, **b**–**i**, *ind hyp*
5.	$\Delta \vdash [\overline{\texttt{T/X}}]\texttt{U} <: [\overline{\texttt{T/X}}]\texttt{T}$	*by* **2**, **b**–**g**, *lemma 17*
6.	$\Delta' \vdash [\overline{\texttt{T/X}}]\texttt{T}$ OK	*by* **3**,**b**–**i**,*lemma 19*
7.	*done*	*by* **4**, **5**, **6**, T-Subs

Case 8 (T-Invk)

1.	$e = e'.\texttt{<}\overline{P}\texttt{>}m(\overline{e})$	*by def* T-INVK
2.	$T = \Downarrow_{\Delta''',\overline{\Delta}} [\overline{T'/Y}]U$	
3.	$\Delta;\Gamma \vdash e' : \exists\Delta'''.N$	*by premises of* T-INVK
4.	$mType(m,N) = \texttt{<}\overline{Y \to [B_l\ B_u]}\texttt{>}\overline{U}\to U$	
5.	$\Delta;\Gamma\vdash \overline{e : \exists\Delta.R}$	
6.	$match(sift(\overline{R},\overline{U},\overline{Y}),\overline{P},\overline{Y},\overline{T'})$	
7.	$\Delta\vdash\overline{P}$ OK	
8.	$\Delta,\Delta''',\overline{\Delta}\vdash \overline{T' <: [\overline{T'/Y}]B}$	
9.	$\Delta,\Delta''',\overline{\Delta}\vdash \overline{R <: [\overline{T'/Y}]U}$	
10.	$\Delta'; [\overline{T/X}]\Gamma \vdash [\overline{T/X}]e' : [\overline{T/X}]\exists\Delta'''.N$	*by* **3**, **b**–**i**, *ind hyp*
11.	$\Delta'; [\overline{T/X}]\Gamma \vdash [\overline{T/X}]e' : \exists[\overline{T/X}]\Delta'''.[\overline{T/X}]N$	*by* **10**, *def subst, Barendregt,* **g**
12.	$mType(m,[\overline{T/X}]N) =$ $\texttt{<}\overline{Y\to[[\overline{T/X}]B_l\ [\overline{T/X}]B_u]}\texttt{>}\overline{[\overline{T/X}]U}\to[\overline{T/X}]U$	*by* **4**, *lemma 6*
13.	$\Delta; [\overline{T/X}]\Gamma \vdash \overline{[\overline{T/X}]e : [\overline{T/X}]\exists\Delta.R}$	*by* **5**, **b**–**i**, *ind hyp*
14.	$\Delta; [\overline{T/X}]\Gamma \vdash \overline{[\overline{T/X}]e : \exists[\overline{T/X}]\Delta.[\overline{T/X}]R}$	*by* **13**, *def subst, Barendregt,* **g**
15.	$(\overline{X}\cup fv(\overline{T}))\cap\overline{Y})=\emptyset$	*by* **4**, *disjointness of formal variables,* **g**
16.	$match(sift(\overline{[\overline{T/X}]R},\overline{[\overline{T/X}]U},\overline{Y}),\overline{[\overline{T/X}]P},\overline{Y},\overline{[\overline{T/X}]T'})$	*by* **6**, **15**, *lemma 2, lemma 3, lemma 4*
17.	$\Delta', [\overline{T/X}]\Delta_j \vdash [\overline{T/X}]T'_i$ OK	*by* **7**, **b**–**i**, *lemma 19*
18.	$\Delta', [\overline{T/X}]\Delta''', [\overline{T/X}]\overline{\Delta}\vdash [\overline{T/X}]T'_i <: [\overline{T/X}][\overline{T'/Y}]B_i$	*by* **8**, **b**–**g**, *lemma 17*
19.	$\overline{Y\to[B_l\ B_u]}\vdash\overline{B_u}$ OK	*by def mType, wf prog,* T-CLASS, T-METHOD
20.	$\Delta', [\overline{T/X}]\Delta''', [\overline{T/X}]\overline{\Delta}\vdash [\overline{T/X}]T'_i <: [\overline{[\overline{T/X}]T'/Y}]B_i$	*by* **18**, **19**
21.	$\Delta', [\overline{T/X}]\Delta''', [\overline{T/X}]\overline{\Delta}\vdash \overline{[\overline{T/X}]R <: [\overline{T/X}][\overline{T'/Y}]U}$	*by* **9**, **b**–**g**, *lemma 17*
22.	$\Delta'; [\overline{T/X}]\Gamma\vdash[\overline{T/X}]e'.\texttt{<}[\overline{T/X}]\overline{P}\texttt{>}m(\overline{[\overline{T/X}]e}) : \Downarrow_{[\overline{T/X}](\Delta''',\overline{\Delta})}[\overline{[\overline{T/X}]T'/Y}][\overline{T/X}]U$	*by* **11**, **12**, **14**, **16**, **17**, **20**, **21**, T-INVK
23.	$[\overline{T/X}][\overline{T'/Y}]U = [\overline{[\overline{T/X}]T'/Y}][\overline{T/X}]U$	*by* **g**
24.	$\Delta'; [\overline{T/X}]\Gamma\vdash[\overline{T/X}]e'.\texttt{<}[\overline{T/X}]\overline{P}\texttt{>}m(\overline{[\overline{T/X}]e}) : [\overline{T/X}]\Downarrow_{\Delta''',\overline{\Delta}}[\overline{T'/Y}]U$	*by* **22**, *lemma 20,* **g**, *Barendregt,* **23**
25.	*done*	*by* **24**, **1**, **2**

□

Lemma 22 (Superclasses are well-formed).

If:

a. $\vdash R \sqsubset: R'$

b. $\Delta\vdash R$ OK

c. $\emptyset\vdash\Delta$ OK

then:

$\Delta \vdash \mathtt{R}'$ OK

Proof *by structural induction on the derivation of* $\vdash \mathtt{R} \sqsubset: \mathtt{R}'$ *with a case analysis on the last step:*

Case 1 (SC-Reflex)

trivial

Case 2 (SC-Trans)

1. $\vdash \mathtt{R} \sqsubset: \mathtt{R}''$ — *by premises* SC-Trans
2. $\vdash \mathtt{R}'' \sqsubset: \mathtt{R}'$ — *by premises* SC-Trans

3. $\Delta \vdash \mathtt{R}''$ OK — *by* **1**, **b**, **c**, *ind hyp*
4. $\Delta \vdash \mathtt{R}'$ OK — *by* **2**, **3**, **c**, *ind hyp*

Case 3 (SC-Sub-Class)

1. $\mathtt{R} = \mathtt{C}\texttt{<}\overline{\mathcal{T}}\texttt{>}$ — *by def* SC-Sub-Class
2. $\mathtt{R}' = [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{N}$ — *by def* SC-Sub-Class

3. *let* $\overline{\mathcal{X}} = \overline{\mathtt{Y}}, \overline{\mathtt{O}}, \mathtt{O}_o, \mathtt{O}_t$
4. *let* $\overline{\mathcal{T}} = \overline{\mathtt{U}}, \overline{\tau}, \tau_o, \tau_t$
5. $\mathtt{class}\ \mathtt{C}\texttt{<}\overline{\mathcal{X} \lhd \mathtt{T}_u}\texttt{>} \lhd \mathtt{N}\ldots$ — *by premise* SC-Sub-Class
6. $\vdash \mathtt{class}\ \mathtt{C}\texttt{<}\overline{\mathcal{X} \lhd \mathtt{T}_u}\texttt{>} \lhd \mathtt{N}\ldots$ OK — *by* **5**, *wf-prog*
7. $\overline{\mathtt{Y}\to[\bot\ \mathtt{T}_u]}, \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \to [\bot\ \mathtt{T}_u], \mathtt{O}_t \to [\bot\ \mathtt{T}_u] \vdash \mathtt{N}$ OK — *by* **6**, *def* T-Class
8. $\mathtt{N} = \mathtt{D}\texttt{<}\overline{\mathcal{T}'}, \mathtt{O}_o, \mathtt{O}_t\texttt{>}$ — *by* **6**, *def* T-Class

9. $\Delta \vdash \overline{\mathcal{T} <: [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{T}_u}$ — *by* **1**, **b**, *def* F-Class, **5**
10. $\Delta \vdash \overline{\mathcal{T}}$ OK — *by* **1**, **b**, *def* F-Class, **5**
11. $\forall \tau \in \overline{\tau}.\ \Delta \vdash \tau_o <: \tau$ — *by* **1**, **b**, *def* F-Class, **5**

12. $[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o = \tau_o$ — *by* **3**, **4**
13. $\Delta \vdash \overline{[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o <: \tau}$ — *by* **11**, **12**
14. $\overline{\mathcal{X}} \cap dom(\Delta) = \emptyset$ — *by* **5**, *distinctness of formal variables*
15. $\Delta, \overline{\mathtt{Y}\to[\bot\ \mathtt{T}_u]}, \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \to [\bot\ \mathtt{T}_u], \mathtt{O}_t \to [\bot\ \mathtt{T}_u] \vdash \mathtt{N}$ OK — *by* **7**, **14**, *lemma 9*
16. $\Delta \vdash [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{N}$ OK — *by* **15**, **10**, **9**, XS-Bttm, **13**, **c**, **14**, *lemma 18*
17. $\Delta \vdash \mathtt{R}'$ OK — *by* **16**, **2**

□

Lemma 23 (Subclassing preserves field types).

If:

a. $\vdash \mathtt{N} \sqsubset: \mathtt{N}'$
b. $fType(\mathtt{f}, \mathtt{N}') = \mathtt{T}$

then:

$fType(\mathtt{f}, \mathtt{N}) = \mathtt{T}$

Proof *by structural induction on the derivation of* $\vdash \mathtt{N} \sqsubset: \mathtt{N}'$ *with a case analysis on the last step:*

Case 1 (SC-Reflex)

trivial

Case 2 (SC-Sub-Class)

1. $\mathtt{N} = \mathtt{C}\texttt{<}\overline{\mathtt{U}}\texttt{>}$ } *by def* SC-Sub-Class

2. $\mathtt{N}' = [\overline{\mathtt{U}/\mathtt{X}}]\mathtt{N}''$ } *by def* SC-Sub-Class

3. $\mathtt{class\ C}\texttt{<}\overline{\mathtt{X}\ldots}\texttt{>} \lhd \mathtt{N}'' \ \{\overline{\mathtt{T\ f;}}\ \overline{\mathtt{M}}\}$ — *by premise of* SC-Sub-Class

Case analysis on $\mathtt{f} \in \overline{\mathtt{f}}$**:**

Case 1 $\mathtt{f} = \mathtt{f}_i$

1.1. $\mathtt{f} \notin fields(\mathtt{N}'')$ — *by distinctness of field names*

1.2. $fType(\mathtt{f}, \mathtt{N}')$ *not defined* — *by* **1.1**, *def fType, def fields*

1.3. *contradiction* — *by* **1.2**, **b**

Case 2 $\mathtt{f} \notin \overline{\mathtt{f}}$

2.1. $\mathtt{f} \in fields(\mathtt{N}'')$ — *by distinctness of field names*

2.2. $fType(\mathtt{f}, \mathtt{N}) = fType(\mathtt{f}, [\overline{\mathtt{U}/\mathtt{X}}]\mathtt{N}'')$ — *by* **2.1**, *def fType, def fields*

2.3. $fType(\mathtt{f}, \mathtt{N}) = \mathtt{T}$ — *by* **2.2**, **2**, **b**

Case 3 (SC-Trans)

1. $\vdash \mathtt{N} \sqsubset: \mathtt{R}$ } *by premises of* SC-Trans

2. $\vdash \mathtt{R} \sqsubset: \mathtt{N}'$ } *by premises of* SC-Trans

3. $\mathtt{R} = \mathtt{N}''$ — *by* **2**, *lemma 15*

4. $fType(\mathtt{m}, \mathtt{N}'') = \mathtt{T}$ — *by* **b**, **2**, **3***m ind hyp*

5. $fType(\mathtt{m}, \mathtt{N}) = \mathtt{T}$ — *by* **4**, **1**, *ind hyp*

□

Lemma 24 (Subclassing preserves method return type).

If:

a. $\vdash \mathtt{N}_1 \sqsubset: \mathtt{N}_2$

b. $mType(\mathtt{m}, \mathtt{N}_2) = \texttt{<}\overline{\mathtt{Y} \lhd \mathtt{T}_u}\texttt{>}\overline{\mathtt{T}} \rightarrow \mathtt{T}$

then:

$mType(\mathtt{m}, \mathtt{N}_1) = \texttt{<}\overline{\mathtt{Y} \lhd \mathtt{T}_u}\texttt{>}\overline{\mathtt{T}} \rightarrow \mathtt{T}$

Proof *by structural induction on the derivation of* $\vdash \mathtt{N}_1 \sqsubset: \mathtt{N}_2$ *with a case analysis on the last step:*

Case 1 (SC-Reflex)

trivial

Case 2 (SC-Sub-Class)

1. $\mathtt{N}_1 = \mathtt{C}\texttt{<}\overline{\mathtt{U}}\texttt{>}$
2. $\mathtt{N}_2 = [\overline{\mathtt{U/X}}]\mathtt{N}$ — *by def* SC-Sub-Class

3. $\mathtt{class}\ \mathtt{C}\texttt{<}\overline{\mathtt{X}\ldots}\texttt{>} \lhd \mathtt{N}\ \{\overline{\mathtt{T}'\ \mathtt{f};}\ \overline{\mathtt{M}}\}$ — *by premise of* SC-Sub-Class

Case analysis on $\mathtt{m} \in \overline{\mathtt{M}}$:

Case 1 $\mathtt{m} \in \mathtt{M}$

1.1. $mType(\mathtt{m}, \mathtt{N}_1) = [\overline{\mathtt{U/X}}]\texttt{<}\overline{\mathtt{Y}' \lhd \mathtt{T}'_u}\texttt{>}\overline{\mathtt{T}'}\to\mathtt{T}'$ — *by def mType*
1.2. $\texttt{<}\overline{\mathtt{Y}' \lhd \mathtt{T}'_u}\texttt{>}\ \mathtt{T}'\ \mathtt{m}(\overline{\mathtt{T}'\ \mathtt{x}})\ \{\ldots\} \in \overline{\mathtt{M}}$ — *by premise of mType*
1.3. $\overline{\mathtt{X}\ldots} \vdash \texttt{<}\overline{\mathtt{Y}' \lhd \mathtt{T}'_u}\texttt{>}\ \mathtt{T}'\ \mathtt{m}(\overline{\mathtt{T}'\ \mathtt{x}})\ \{\ldots\}\ \text{OK}$ — *by wf prog,* **3***, premises of* T-Class
1.4. $overrride(\mathtt{m}, \mathtt{N}, \texttt{<}\overline{\mathtt{Y}' \lhd \mathtt{T}'_u}\texttt{>}\overline{\mathtt{T}'}\to\mathtt{T}')$ — *by premise* T-Method*,* **1.3**
1.5. $\texttt{<}\overline{\mathtt{Y}' \lhd \mathtt{T}'_u}\texttt{>}\overline{\mathtt{T}'}\to\mathtt{T}' = mType(\mathtt{m}, \mathtt{N})$ — *by* **1.4***,* **b***,* **2**
1.6. $[\overline{\mathtt{U/X}}]\texttt{<}\overline{\mathtt{Y}' \lhd \mathtt{T}'_u}\texttt{>}\overline{\mathtt{T}'}\to\mathtt{T}' = mType(\mathtt{m}, [\overline{\mathtt{U/X}}]\mathtt{N})$ — *by* **1.5***, lemma 6*
1.7. *done* — *by* **2***,* **1.6***,* **1.1**

Case 2 $\mathtt{m} \notin \overline{\mathtt{M}}$

2.1. $mType(\mathtt{m}, \mathtt{N}_1) = mType(\mathtt{m}, [\overline{\mathtt{U/X}}]\mathtt{N})$ — *by def mType*
2.2. *done* — *by* **2.1***,* **2**

Case 3 (SC-Trans)

1. $\vdash \mathtt{N}_1 \sqsubset: \mathtt{R}$
2. $\vdash \mathtt{R} \sqsubset: \mathtt{N}_2$ — *by premises of* SC-Trans

3. $\mathtt{R} = \mathtt{N}_3$ — *by* **2***, lemma 15*
4. $mType(\mathtt{m}, \mathtt{N}_3) = \texttt{<}\overline{\mathtt{Y} \lhd \mathtt{T}_u}\texttt{>}\overline{\mathtt{T}}\to\mathtt{T}$ — *by* **b***,* **2***, ind hyp*
5. $mType(\mathtt{m}, \mathtt{N}_1) = \texttt{<}\overline{\mathtt{Y} \lhd \mathtt{T}_u}\texttt{>}\overline{\mathtt{T}}\to\mathtt{T}$ — *by* **4***,* **1***, ind hyp*

□

Lemma 25 (Expression substitution preserves typing).

If:

a. $\Delta; \Gamma, \gamma : \mathtt{U} \vdash \mathtt{e} : \mathtt{T}$
b. $\Delta; \Gamma \vdash \mathtt{e}' : \mathtt{U}'$
c. $\Delta \vdash \mathtt{U}' <: \mathtt{U}$
d. $\Delta \vdash \mathtt{U}\ \text{OK}$

then:

$\Delta; \Gamma \vdash [\mathtt{e}'/\gamma]\mathtt{e} : \mathtt{T}$

Proof *by structural induction on the derivation of* $\Delta; \Gamma, \gamma : \mathtt{U} \vdash \mathtt{e} : \mathtt{T}$ *with a case analysis on the last step:*

Case 1 (T-Var)

1.	$\mathtt{e} = \gamma'$	*by def* T-Var
2.	$\mathtt{T} = \Gamma(\gamma')$	

Case analysis on γ'**:**

Case 1 $\gamma' = \gamma$

1.1.	$\mathtt{T} = \mathtt{U}$	*by* **a**, **2**
1.2.	$[\mathtt{e}'/\gamma]\mathtt{e} = \mathtt{e}'$	*by def subst*
1.3.	$\Delta; \Gamma \vdash [\mathtt{e}'/\gamma]\mathtt{e} : \mathtt{U}'$	*by* **1.2**, **b**
1.4.	$\Delta; \Gamma \vdash [\mathtt{e}'/\gamma]\mathtt{e} : \mathtt{U}$	*by* **1.3**, **c**, **d**, F-Env-Empty, T-Subs
1.5.	$\Delta; \Gamma \vdash [\mathtt{e}'/\gamma]\mathtt{e} : \mathtt{T}$	*by* **1.4**, **1.1**

Case 2 $\gamma' \neq \gamma$

2.1.	$[\mathtt{e}'/\gamma]\mathtt{e} = \mathtt{e}$	*by def subst*
2.2.	$\Delta; \Gamma \vdash [\mathtt{e}'/\gamma]\mathtt{e} : \mathtt{T}$	*by* **2.1**, **2**, **1**, **a**

Case 2 (T-Null)

trivial

Case 3 (T-Cast)

1.	$e = (T)\mathtt{e}'$	*by def* T-Cast
2.	$\Delta; \Gamma, \gamma : \mathtt{U} \vdash \mathtt{e}' : \mathtt{T}'$	*by premises* T-Cast
3.	$\Delta \vdash \mathtt{T} <: \mathtt{T}'$	
4.	$\Delta \vdash \mathtt{T}$ OK	
5.	$\Delta; \Gamma \vdash [\mathtt{e}'/\gamma]\mathtt{e}' : \mathtt{T}'$	*by* **2**, **b**, **c**, **d**, *ind hyp*
6.	$\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T}$	*by* **1**, **5**, **3**, **4**, T-Cast

Case 4 (T-New)

trivial

Case 5 (T-Field)

1.	$\mathtt{e} = \mathtt{e}.''\mathtt{f}$	*by def* T-Field
2.	$\mathtt{T} = \Downarrow_{\Delta'} \mathtt{T}'$	
3.	$\Delta; \Gamma, \gamma : \mathtt{U} \vdash \mathtt{e}'' : \exists\Delta'.\mathtt{N}$	*by premises* T-Field
4.	$fType(\mathtt{f}, \mathtt{N}) = \mathtt{T}'$	

5.	$\Delta; \Gamma \vdash [e'/\gamma]e'' : \exists\Delta'.N$	*by* **3**, **b**, **c**, **d**, *ind hyp*
6.	$\Delta; \Gamma \vdash [e'/\gamma]e''.f :\Downarrow_{\Delta'} T'$	*by* **5**, **4**, T-Field
7.	$[e'/\gamma]e = [e'/\gamma]e''.f$	*by def subst,* **1**
8.	$\Delta; \Gamma \vdash [e'/\gamma]e : T$	*by* **6**, **7**, **2**

Case 6 (T-Assign)

1.	$e = e.''f\ =\ e_3$	*by def* T-Assign
2.	$\Delta; \Gamma, \gamma{:}U \vdash e'' : \exists\Delta'.N$	} *by premises* T-Assign (2–5)
3.	$fType(f, N) = T'$	
4.	$\Delta; \Gamma, \gamma{:}U \vdash e_3 : T$	
5.	$\Delta, \Delta' \vdash T <: T'$	
6.	$\Delta; \Gamma \vdash [e'/\gamma]e'' : \exists\Delta'.N$	*by* **2**, **b**, **c**, **d**, *ind hyp*
7.	$\Delta; \Gamma \vdash [e'/\gamma]e_3 : T$	*by* **4**, **b**, **c**, **d**, *ind hyp*
8.	$\Delta; \Gamma \vdash [e'/\gamma]e''.f\ =\ [e'/\gamma]e_3 : T$	*by* **6**, **3**, **7**, **5**, T-Assign
9.	$[e'/\gamma]e = [e'/\gamma]e''.f\ =\ [e'/\gamma]e_3$	*by def subst,* **1**
10.	$\Delta; \Gamma \vdash [e'/\gamma]e : T$	*by* **8**, **9**

Case 7 (T-Invk)

1.	$e = e''.{<}\overline{P}{>}m(\overline{e})$	} *by def* T-Invk (1–2)
2.	$T =\Downarrow_{\Delta'',\overline{\Delta}} [\overline{T/Y}]U_m$	
3.	$\Delta; \Gamma, \gamma{:}U \vdash e'' : \exists\Delta''.N$	} *by premises* T-Invk (3–9)
4.	$mType(m,\ N) = {<}\overline{Y \lhd\ B_m}{>}\overline{U_m} \rightarrow U_m$	
5.	$\Delta; \Gamma, \gamma{:}U \vdash \overline{e : \exists\Delta.R}$	
6.	$match(sift(\overline{R}, \overline{U_m}, \overline{Y}), \overline{P}, \overline{Y}, \overline{T})$	
7.	$\Delta, \overline{\Delta} \vdash \overline{T}$ OK	
8.	$\Delta, \Delta'', \overline{\Delta} \vdash \overline{T <: [\overline{T/Y}]B_m}$	
9.	$\Delta, \Delta'', \overline{\Delta} \vdash \overline{\exists\emptyset.R <: [\overline{T/Y}]U_m}$	
10.	$\Delta; \Gamma \vdash [e'/\gamma]e'' : \exists\Delta''.N$	*by* **3**, **b**, **c**, **d**, *ind hyp*
11.	$\Delta; \Gamma \vdash \overline{[e'/\gamma]e : \exists\Delta.R}$	*by* **5**, **b**, **c**, **d**, *ind hyp*
12.	$\Delta; \Gamma \vdash [e'/\gamma]e''.{<}\overline{P}{>}m(\overline{[e'/\gamma]e}) :\Downarrow_{\Delta'',\overline{\Delta}} [\overline{T/Y}]U_m$	*by* **10**, **4**, **11**, **6**, **7**, **8**, **9**, T-Invk
13.	$[e'/\gamma]e = [e'/\gamma]e''.{<}\overline{P}{>}m(\overline{[e'/\gamma]e})$	*by def subst,* **1**
14.	$\Delta; \Gamma \vdash e : T$	*by* **12**, **13**, **2**

Case 8 (T-Subs)

1.	$\Delta; \Gamma, \gamma{:}U \vdash e : T'$	} *by premises* T-Subs (1–3)
2.	$\Delta \vdash T' <: T$	
3.	$\Delta \vdash T$ OK	
4.	$\Delta; \Gamma \vdash [e'/\gamma]e : T'$	*by* **1**, **b**, **c**, **d**, *ind hyp*
5.	$\Delta; \Gamma \vdash e : T$	*by* **4**, **2**, **3**, T-Subs

□

Lemma 26 (Corrolary to lemma 25).

a. $\Delta; \Gamma, \gamma : \mathtt{U} \vdash \mathtt{e} : \mathtt{T}$
b. $\Delta \vdash \mathtt{U'} <: \mathtt{U}$
c. $\Delta \vdash \mathtt{U}$ OK

then:

$\Delta; \Gamma, \gamma : \mathtt{U'} \vdash \mathtt{e} : \mathtt{T}$

Proof

1.	$\Delta; \Gamma, \gamma' : \mathtt{U'}, \gamma : \mathtt{U} \vdash \mathtt{e} : \mathtt{T}$	*by* **a**, $\mathtt{x'}$ *is fresh, lemma 10*
2.	$\Delta; \Gamma, \gamma' : \mathtt{U'} \vdash \gamma' : \mathtt{U'}$	*by* T-Var
3.	$\Delta; \Gamma, \gamma' : \mathtt{U'} \vdash [\gamma'/\gamma]\mathtt{e} : \mathtt{T}$	*by* **1**, **2**, **b**, **c**, *lemma 25*
4.	$\Delta; \Gamma, \gamma : \mathtt{U'} \vdash \mathtt{e} : \mathtt{T}$	*by renaming* **3**

□

Lemma 27 (*fType* gives well-formed types).

If:

a. $fType(\mathtt{f}, \mathtt{C}\texttt{<}\overline{\mathcal{T}}\texttt{>}) = \mathtt{T}$
b. $\emptyset \vdash \Delta$ OK
c. $\Delta \vdash \exists\Delta'.\mathtt{C}\texttt{<}\overline{\mathcal{T}}\texttt{>}$ OK

then:

$\Delta, \Delta' \vdash \mathtt{T}$ OK

Proof *by induction on the derivation of* $fType(\mathtt{f}, \mathtt{C}\texttt{<}\overline{\mathcal{T}}\texttt{>}) = \mathtt{T}$ *with a case analysis on the last step:*

Case 1 *base case*

1.	$\mathtt{class}\ \mathtt{C}\texttt{<}\overline{\mathcal{X} \triangleleft\ \mathtt{T}_u}\texttt{>}\ldots\overline{\mathtt{T}\ \mathtt{f};}\ldots$	*by premise fType*
2.	$\mathtt{f} = \mathtt{f}_i$	*by def fType*
3.	$\mathtt{T} = [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{T}_i$	*by def fType*
4.	$let\overline{\mathcal{X}} = \overline{\mathtt{Y}}, \overline{\mathtt{O}}, \mathtt{O}_o, \mathtt{O}_t$	
5.	$let\overline{\mathcal{T}} = \overline{\mathtt{U}}, \overline{\tau}, \tau_o, \tau_t$	
6.	$\vdash \mathtt{class}\ \mathtt{C}\texttt{<}\overline{\mathcal{X} \triangleleft\ \mathtt{T}_u}\texttt{>}\ldots\overline{\mathtt{T}\ \mathtt{f};}\ldots$ OK	*by* **1**, *wf-prog*
7.	$\overline{\mathtt{Y} \rightarrow [\bot\ \mathtt{T}_u]}, \overline{\mathtt{O} \rightarrow [\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \rightarrow [\bot\ \mathtt{T}_u], \mathtt{O}_t \rightarrow [\bot\ \mathtt{T}_u] \vdash \mathtt{T}_i$ OK	*by* **6**, *def* T-Class
8.	$\Delta, \Delta', \overline{\mathtt{Y} \rightarrow [\bot\ \mathtt{T}_u]}, \overline{\mathtt{O} \rightarrow [\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \rightarrow [\bot\ \mathtt{T}_u], \mathtt{O}_t \rightarrow [\bot\ \mathtt{T}_u] \vdash \mathtt{T}_i$ OK	*by* **7**, **1**, *distinctness of formal variables, lemma 9*
9.	$\Delta, \Delta' \vdash \mathtt{C}\texttt{<}\overline{\mathcal{T}}\texttt{>}$ OK	*by* **c**, *def* F-Exists
10.	$\Delta \vdash \Delta'$ OK	*by* **c**, *def* F-Exists

11.	$\Delta, \Delta' \vdash \overline{\mathcal{T}}$ OK	*by* **9**, *def* F-Class
12.	$\Delta, \Delta' \vdash \overline{\mathcal{T} <: [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{T}_u}$	
13.	$\forall \tau \in \overline{\tau}.\ \Delta \vdash \tau_o <: \tau$	
14.	$\emptyset \vdash \Delta, \Delta'$ OK	*by* **10**, **b**, *lemma 12*
15.	$[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o = \tau_o$	*by* **4**, **5**
16.	$\Delta \vdash \overline{[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o <: \tau}$	*by* **13**, **15**
17.	$\Delta, \Delta' \vdash [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{T}_i$ OK	*by* **8**, **11**, **12**, XS-Bttm, **16**, **14**, *lemma 18*
18.	$\Delta, \Delta' \vdash \mathtt{T}$ OK	*by* **17**, **3**

Case 2 *inductive case*

1.	`class` $\mathtt{C}$<$\overline{\mathcal{X} \lhd \mathtt{T}_u}$> $\lhd$ `N` ...$\overline{\mathtt{T\ f;}}$...	*by premises fType*
2.	$\mathtt{f} \notin \overline{\mathtt{f}}$	
3.	$\mathtt{T} = fType(\mathtt{f}, [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{N})$	*by def fType*
4.	$let \overline{\mathcal{X}} = \overline{\mathtt{Y}}, \overline{\mathtt{O}}, \mathtt{O}_o, \mathtt{O}_t$	
5.	$let \overline{\mathcal{T}} = \overline{\mathtt{U}}, \overline{\tau}, \tau_o, \tau_t$	
6.	$\vdash$ `class` $\mathtt{C}$<$\overline{\mathcal{X} \lhd \mathtt{T}_u}$> $\lhd$ `N` ...$\overline{\mathtt{T\ f;}}$... OK	*by* **1**, *wf-prog*
7.	$\overline{\mathtt{Y} \to [\bot\ \mathtt{T}_u]}, \overline{\mathtt{O} \to [\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \to [\bot\ \mathtt{T}_u], \mathtt{O}_t \to [\bot\ \mathtt{T}_u] \vdash \mathtt{N}$ OK	*by* **6**, *def* T-Class
8.	$\Delta, \Delta', \overline{\mathtt{Y} \to [\bot\ \mathtt{T}_u]}, \overline{\mathtt{O} \to [\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \to [\bot\ \mathtt{T}_u], \mathtt{O}_t \to [\bot\ \mathtt{T}_u] \vdash \mathtt{N}$ OK	*by* **7**, **1**, *distinctness of formal variables, lemma 9*
9.	$\Delta, \Delta' \vdash \mathtt{C}$<$\overline{\mathcal{T}}$> OK	*by* **c**, *def* F-Exists
10.	$\Delta \vdash \Delta'$ OK	
11.	$\Delta, \Delta' \vdash \overline{\mathcal{T}}$ OK	*by* **9**, *def* F-Class
12.	$\Delta, \Delta' \vdash \overline{\mathcal{T} <: [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{T}_u}$	
13.	$\forall \tau \in \overline{\tau}.\ \Delta \vdash \tau_o <: \tau$	
14.	$\emptyset \vdash \Delta, \Delta'$ OK	*by* **10**, **b**, *lemma 12*
15.	$[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o = \tau_o$	*by* **4**, **5**
16.	$\Delta \vdash \overline{[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o <: \tau}$	*by* **13**, **15**
17.	$\Delta, \Delta' \vdash [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{N}$ OK	*by* **8**, **11**, **12**, XS-Bttm, **16**, **14**, *lemma 18*
18.	$\Delta, \Delta' \vdash \mathtt{T}$ OK	*by* **3**, **17**, **14**, *ind hyp*

□

Lemma 28 **(*mType* gives well-formed types).**

If:

a.	$mType(\mathtt{m}, \mathtt{C}$<$\overline{\mathcal{T}}$>$) =$ <$\overline{\mathtt{Y} \lhd \mathtt{T}_u}$>$\overline{\mathtt{T}} \to \mathtt{T}$
b.	$\emptyset \vdash \Delta$ OK
c.	$\Delta \vdash \exists \Delta'.\mathtt{C}$<$\overline{\mathcal{T}}$> OK

then:

$\Delta, \Delta', \overline{\mathtt{Y} \to [\bot\ \mathtt{T}_u]} \vdash \overline{\mathtt{T}}$ OK

$\Delta, \Delta', \overline{\mathtt{Y} \to [\bot\ \mathtt{T}_u]} \vdash \mathtt{T}$ OK

$\Delta, \Delta', \overline{\mathtt{Y} \to [\bot\ \mathtt{T}_u]} \vdash \overline{\mathtt{T}_u}$ OK

Proof *by induction on the derivation of* $mType(\mathtt{m}, \mathtt{C<}\overline{\mathcal{T}}\mathtt{>}) = \mathtt{<}\overline{\mathtt{Y}\lhd\ \mathtt{T}_u}\mathtt{>}\overline{\mathtt{T}}\to\mathtt{T}$ *with a case analysis on the last step:*

Case 1 *base case*

1.	$\mathtt{class\ C<}\overline{\mathcal{X}\lhd\ \mathtt{U}_u}\mathtt{>}\ \lhd\ \mathtt{N}\ \ldots\overline{\mathtt{M}}\ldots$	*by premises mType*
2.	$\mathtt{<Y'}\lhd\ \mathtt{T}'_u\mathtt{>T'\ m(}\overline{\mathtt{T'\ x}}\mathtt{)}\ \ldots \in \overline{\mathtt{M}}$	
3.	$\mathtt{<}\overline{\mathtt{Y}\lhd\ \mathtt{T}_u}\mathtt{>}\overline{\mathtt{T}}\to\mathtt{T} = [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{<}\overline{\mathtt{Y'}\lhd\ \mathtt{T}'_u}\mathtt{>}\overline{\mathtt{T'}}\to\mathtt{T'}$	*by def mType*
4.	$let\overline{\mathcal{X}} = \overline{\mathtt{Y}}, \overline{\mathtt{O}}, \mathtt{O}_o, \mathtt{O}_t$	
5.	$let\overline{\mathcal{T}} = \overline{\mathtt{U}}, \overline{\tau}, \tau_o, \tau_t$	
6.	$\vdash \mathtt{class\ C<}\overline{\mathcal{X}\lhd\ \mathtt{U}_u}\mathtt{>}\ldots\overline{\mathtt{M}}\ldots$ OK	*by* **1**, *wf-prog*
7.	$\overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{U}_u]}, \mathtt{O}_o\to[\bot\ \mathtt{U}_u], \mathtt{O}_t\to[\bot\ \mathtt{U}_u] \vdash \mathtt{<}\overline{\mathtt{Y'}\lhd\ \mathtt{T}'_u}\mathtt{>T'\ m(}\overline{\mathtt{T'\ x}}\mathtt{)}$ OK	*by* **6**, *def* T-Class OK
8.	$\overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{U}_u]}, \mathtt{O}_o\to[\bot\ \mathtt{U}_u], \mathtt{O}_t\to[\bot\ \mathtt{U}_u], \overline{\mathtt{Y'}\to[\bot\ \mathtt{T}'_u]} \vdash \overline{\mathtt{T'}}, \mathtt{T'}$ OK	*by* **7**, *def* T-Method
9.	$\Delta, \Delta', \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{U}_u]}, \mathtt{O}_o\to[\bot\ \mathtt{U}_u], \mathtt{O}_t\to[\bot\ \mathtt{U}_u], \overline{\mathtt{Y'}\to[\bot\ \mathtt{T}'_u]} \vdash \overline{\mathtt{T'}}, \mathtt{T'}$ OK	*by* **8**, **12**, *distinctness of formal variables, lemma 9*
10.	$\overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{U}_u]}, \mathtt{O}_o\to[\bot\ \mathtt{U}_u], \mathtt{O}_t\to[\bot\ \mathtt{U}_u] \vdash \overline{\mathtt{Y'}\to[\bot\ \mathtt{T}'_u]}$ OK	*by* **7**, *def* T-Method
11.	$\Delta, \Delta', \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{U}_u]}, \mathtt{O}_o\to[\bot\ \mathtt{U}_u], \mathtt{O}_t\to[\bot\ \mathtt{U}_u] \vdash \overline{\mathtt{Y'}\to[\bot\ \mathtt{T}'_u]}$ OK	*by* **10**, **11**, *distinctness of formal variables, lemma 9*
12.	$\overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{U}_u]}, \mathtt{O}_o\to[\bot\ \mathtt{U}_u], \mathtt{O}_t\to[\bot\ \mathtt{U}_u], \overline{\mathtt{Y'}\to[\bot\ \mathtt{T}'_u]} \vdash \overline{\mathtt{T}'_u}$ OK	*by* **10**, *def* F-Env
13.	$\Delta, \Delta', \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{U}_u]}, \mathtt{O}_o\to[\bot\ \mathtt{U}_u], \mathtt{O}_t\to[\bot\ \mathtt{U}_u], \overline{\mathtt{Y'}\to[\bot\ \mathtt{T}'_u]} \vdash \overline{\mathtt{T}'_u}$ OK	*by* **12**, **12**, *distinctness of formal variables, lemma 9*
14.	$\Delta, \Delta' \vdash \mathtt{C<}\overline{\mathcal{T}}\mathtt{>}$ OK	*by* **c**, *def* F-Exists
15.	$\Delta \vdash \Delta'$ OK	
16.	$\Delta, \Delta' \vdash \overline{\mathcal{T}}$ OK	*by* **14**, *def* F-Class
17.	$\Delta, \Delta' \vdash \overline{\mathcal{T} <: [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{U}_u}$	
18.	$\forall\tau \in \overline{\tau}.\ \Delta \vdash \tau_o <: \tau$	
19.	$\emptyset \vdash \Delta, \Delta'$ OK	*by* **15**, **b**, *lemma 12*
20.	$[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o = \tau_o$	*by* **4**, **5**
21.	$\Delta \vdash \overline{[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o <: \tau}$	*by* **18**, **20**
22.	$\Delta, \Delta', \overline{\mathtt{Y'}\to[\bot\ [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{T}'_u]} \vdash \overline{[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{T}'_u}, \overline{[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{T'}}, [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{T}$ OK	*by* **9**, **13**, **16**, **17**, XS-Bttm, **21** **19**, **11**, *lemma 19*
23.	$\Delta, \Delta', \overline{\mathtt{Y}\to[\bot\ \mathtt{T}_u]} \vdash \overline{\mathtt{T}_u}, \overline{\mathtt{T}}, \mathtt{T}$ OK	*by* **22**, **3**

Case 2 *inductive case*

1.	$\mathtt{class\ C<}\overline{\mathcal{X}\lhd\ \mathtt{U}_u}\mathtt{>}\ \lhd\ \mathtt{N}\ \ldots\overline{\mathtt{M}}\ldots$	*by premises mType*
2.	$\mathtt{m} \notin \overline{\mathtt{M}}$	
3.	$\mathtt{<}\overline{\mathtt{Y}\lhd\ \mathtt{T}_u}\mathtt{>}\overline{\mathtt{T}}\to\mathtt{T} = mType(\mathtt{m}, [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{N})$	*by def mType*
4.	$let\overline{\mathcal{X}} = \overline{\mathtt{Y}}, \overline{\mathtt{O}}, \mathtt{O}_o, \mathtt{O}_t$	
5.	$let\overline{\mathcal{T}} = \overline{\mathtt{U}}, \overline{\tau}, \tau_o, \tau_t$	

6.	$\vdash \mathtt{class\ C}\texttt{<}\overline{\mathcal{X} \lhd\ \mathtt{U}_u}\texttt{>}\ \lhd\ \mathtt{N}\ \ldots\overline{\mathtt{M}}\ldots\ \text{OK}$	*by* **1**, *wf-prog*
7.	$\overline{\mathtt{Y}\rightarrow[\bot\ \mathtt{T}_u]}, \overline{\mathtt{O}\rightarrow[\mathtt{O}_o\ \mathtt{U}_u]}, \mathtt{O}_o \rightarrow[\bot\ \mathtt{U}_u], \mathtt{O}_t \rightarrow[\bot\ \mathtt{U}_u] \vdash \mathtt{N}\ \text{OK}$	*by* **6**, *def* T-CLASS
8.	$\Delta, \Delta', \overline{\mathtt{Y}\rightarrow[\bot\ \mathtt{T}_u]}, \overline{\mathtt{O}\rightarrow[\mathtt{O}_o\ \mathtt{U}_u]}, \mathtt{O}_o \rightarrow[\bot\ \mathtt{U}_u], \mathtt{O}_t \rightarrow[\bot\ \mathtt{U}_u] \vdash \mathtt{N}\ \text{OK}$	*by* **7**, *distinctness of formal variables, lemma 9*
9.	$\Delta, \Delta' \vdash \mathtt{C}\texttt{<}\overline{\mathcal{T}}\texttt{>}\ \text{OK}$	} *by* **c**, *def* F-EXISTS
10.	$\Delta \vdash \Delta'\ \text{OK}$	
11.	$\Delta, \Delta' \vdash \overline{\mathcal{T}}\ \text{OK}$	} *by* **9**, *def* F-CLASS
12.	$\Delta, \Delta' \vdash \overline{\mathcal{T} <: [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{U}_u}$	
13.	$\forall \tau \in \overline{\tau}.\ \Delta \vdash \tau_o <: \tau$	
14.	$\emptyset \vdash \Delta, \Delta'\ \text{OK}$	*by* **10**, **b**, *lemma 12*
15.	$[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o = \tau_o$	*by* **4**, **5**
16.	$\Delta \vdash \overline{[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o <: \tau}$	*by* **13**, **15**
17.	$\Delta, \Delta' \vdash [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{N}\ \text{OK}$	*by* **8**, **11**, **12**, XS-BTTM, **16**, **14**, *lemma 18*
18.	$\Delta, \Delta', \overline{\mathtt{Y}\rightarrow[\bot\ \mathtt{T}_u]} \vdash \overline{\mathtt{T}}\ \text{OK}$	} *by* **3**, **17**, **14**, *ind hyp*
19.	$\Delta, \Delta', \overline{\mathtt{Y}\rightarrow[\bot\ \mathtt{T}_u]} \vdash \mathtt{T}\ \text{OK}$	
20.	$\Delta, \Delta', \overline{\mathtt{Y}\rightarrow[\bot\ \mathtt{T}_u]} \vdash \overline{\mathtt{T}_u}\ \text{OK}$	

□

Lemma 29 (*match* gives well-formed types).

If:

a. $\Delta \vdash \overline{\mathtt{P}}\ \text{OK}$

b. $\Delta \vdash \overline{\exists\Delta.\mathtt{R}}\ \text{OK}$

c. $\emptyset \vdash \Delta\ \text{OK}$

d. $match(\langle\overline{\mathtt{R}}, \overline{\exists\Delta'.\mathtt{R}'}\rangle, \overline{\mathtt{P}}, \overline{\mathtt{Y}}, \overline{\mathtt{T}})$

then:

$\Delta, \overline{\Delta} \vdash \overline{\mathtt{T}}\ \text{OK}$

Proof

1.	$\forall i\ where\ \mathtt{P}_i \neq \star : \mathtt{T}_i = \mathtt{P}_i$	} *by* **d**, *def match*
2.	$\forall j\ \underline{where}\ \mathtt{P}_j = \star : \mathtt{Y}_j \in fv(\overline{\mathtt{R}'})$	
3.	$\vdash \mathtt{R} \sqsubset: [\overline{\mathtt{T}/\mathtt{Y}}, \overline{\mathtt{T}'/\mathtt{X}}]\mathtt{R}'$	
4.	$dom(\overline{\Delta}) = \overline{\mathtt{X}}$	
5.	$fv(\overline{\mathtt{T}}, \overline{\mathtt{T}'}) \cap \overline{\mathtt{Y}}, \overline{\mathtt{X}} = \emptyset$	
6.	$\Delta, \overline{\Delta} \vdash \overline{\mathtt{R}}\ \text{OK}$	} *by* **b**, *def* F-EXIST
7.	$\Delta \vdash \overline{\Delta}\ \text{OK}$	
8.	$\emptyset \vdash \Delta, \overline{\Delta}\ \text{OK}$	*by* **7**, **c**, *lemma 12*
9.	$\Delta, \overline{\Delta} \vdash [\overline{\mathtt{T}/\mathtt{Y}}, \overline{\mathtt{T}'/\mathtt{X}}]\mathtt{R}'\ \text{OK}$	*by* **3**, **6**, **8**, *lemma 22*

Case analysis on *each* $\mathtt{T}_i \in \overline{\mathtt{T}}$**:**

Case 1 $\mathtt{P}_i \neq \star$

1.1.	$\Delta \vdash \mathtt{P}_i$ OK	*by* **a**
1.2.	$\Delta, \overline{\Delta} \vdash \mathtt{P}_i$ OK	*by* **1.1**, **8**, *lemma 9*
1.3.	$\Delta, \overline{\Delta} \vdash \mathtt{T}_i$ OK	*by* **1.2**, **1**

Case 2 $\mathtt{P}_i = \star$

2.1.	$\mathtt{Y}_i \in fv(\overline{\mathtt{N}'})$	*by* **2**
2.2.	*let* $\overline{\mathtt{N}'} = \overline{\mathtt{C<\overline{U}>}}$	
2.3.	$\mathtt{[\overline{T/Y},\overline{T'/X}]N'} = \overline{\mathtt{C<[\overline{T/Y},\overline{T'/X}]\overline{U}>}}$	*by* **2.2**, *def subst*
2.4.	$\exists \mathtt{N}'_j$ *such that* $\mathtt{[\overline{T/Y},\overline{T'/X}]N}'_j = \mathtt{C}_j\mathtt{<\ldots,T}_i\mathtt{,\ldots>}$	*by* **2.1**, **2.3**
2.5.	$\Delta, \overline{\Delta} \vdash \mathtt{T}_i$ OK	*by* **2.4**, **9**, *def* F-CLASS

□

Lemma 30 (*Close* gives well-formed types).

If:

a. $\Delta, \Delta' \vdash \mathtt{T}$ OK

b. $\Delta \vdash \Delta'$ OK

then:

$\Delta \vdash \Downarrow_{\Delta'} \mathtt{T}$ OK

Proof *by structural induction on the derivation of* $\Downarrow_{\Delta} \mathtt{T}$ *with a case analysis on the last step:*

Case 1 *1*

1.	$\mathtt{T} = \exists\emptyset.\mathtt{X}$	} *by def* close
2.	$\Downarrow_{\Delta'} \mathtt{T} = \exists\emptyset.\mathtt{X}$	
3.	$\mathtt{X} \notin dom(\Delta')$	*by premise of* close
4.	$\Delta \vdash \exists\emptyset.\mathtt{X}$ OK	*by* **3**, **a**, *def* F-VAR
5.	*done*	*by* **4**, **1**, **2**

Case 2 *2*

1.	$\mathtt{T} = \exists\emptyset.\mathtt{X}$	} *by def* close
2.	$\Downarrow_{\Delta'} \mathtt{T} = \Downarrow_{\Delta} \mathtt{B}_u$	
3.	$\Delta'(\mathtt{X}) = [\mathtt{B}_l\ \ \mathtt{B}_u]$	*by premise of* close
4.	$\Delta, \Delta' \vdash \mathtt{B}_u$ OK	*by* **b**, *def* F-ENV
5.	$\Delta \vdash \Downarrow_{\Delta'} \mathtt{B}_u$ OK	*by* **4**, **b**, *ind hyp*
6.	*done*	*by* **5**, **2**

Case 3 *3*

1.	$\mathtt{T} = \exists\Delta''.\mathtt{N}$	*by def* close
2.	$\Downarrow_{\Delta'} \mathtt{T} = \exists\Delta', \Delta''.\mathtt{N}$	
3.	$\Delta, \Delta' \vdash \Delta''$ OK	*by* **a**, **1**, *def* F-Exists
4.	$\Delta, \Delta', \Delta'' \vdash \mathtt{N}$ OK	
5.	$\Delta \vdash \Delta'', \Delta'$ OK	*by* **3**, **a**, *lemma 12*
6.	$\Delta \vdash \exists\Delta', \Delta''.\mathtt{N}$ OK	*by* **5**, **4**, F-Exists
7.	*done*	*by* **6**, **2**

□

Lemma 31 (Typing gives well-formed types).

If:

a. $\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T}$

b. $\emptyset \vdash \Delta$ OK

c. $\forall \mathtt{x} \in dom(\Gamma) : \ \Delta \vdash \Gamma(\mathtt{x})$ OK

then:

$\Delta \vdash \mathtt{T}$ OK

Proof *by structural induction on the derivation of* $\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T}$ *with a case analysis on the last step:*

Case 1 (T-Var)

1.	$\Delta' = \emptyset$	*by def* T-Var
2.	$\mathtt{T} = \Gamma(\mathtt{x})$	*by def* T-Var
3.	*done*	*by* **2**, **1**, **c**

Case 2 (T-Subs, T-Cast, T-Null, T-New)

trivial

Case 3 (T-Field)

1.	$\mathtt{e} = \mathtt{e}'.\mathtt{f}$	*by def* T-Field
2.	$\mathtt{T} = \Downarrow_{\Delta'} \mathtt{U}$	
3.	$\Delta; \Gamma \vdash \mathtt{e}' : \exists\Delta'\mathtt{N}$	*by premises* T-Field
4.	$fType(\mathtt{f}, \mathtt{N}) = \mathtt{U}$	
5.	$\Delta \vdash \exists\Delta'.\mathtt{N}$ OK	*by* **3**, **b**, **c**, *ind hyp*
6.	$\Delta, \Delta' \vdash \mathtt{U}$ OK	*by* **4**, **5**, **b**, *lemma 27*
7.	$\Delta \vdash \Delta'$ OK	*by* **5**, *def* F-Exists
8.	$\Delta \vdash \Downarrow_{\Delta'} \mathtt{U}$ OK	*by* **6**, **7**, *lemma 30*
9.	*done*	*by* **8**, **2**

Case 4 (T-Assign)

1.	$\mathtt{e} = \mathtt{e}'\mathtt{.f\ =\ e}''$	*by def* T-Assign
2.	$\Delta; \Gamma \vdash \mathtt{e}'' : \mathtt{T}$	*by premise* T-Assign
3.	$\Delta \vdash \mathtt{T}$ OK	*by* **2**, **b**, **c**, *ind hyp*

Case 5 (T-Invk)

1.	$\mathtt{e} = \mathtt{e}'\mathtt{<}\overline{\mathtt{P}}\mathtt{>m(}\overline{\mathtt{e}}\mathtt{)}$	} *by def* T-Invk
2.	$\mathtt{T} = \Downarrow_{\Delta'',\overline{\Delta}} \overline{\mathtt{[T/Y]U}}$	
3.	$\Delta; \Gamma \vdash \mathtt{e}' : \exists\Delta''\mathtt{.N}$	} *by premises* T-Invk
4.	$mType(\mathtt{m}, \mathtt{N}) = \mathtt{<}\overline{\mathtt{Y}\lhd\ \mathtt{T}_u}\mathtt{>}\overline{\mathtt{U}}\mathtt{\rightarrow U}$	
5.	$\Delta; \Gamma \vdash \overline{\mathtt{e} : \exists\Delta\mathtt{.R}}$	
6.	$\Delta \vdash \overline{\mathtt{P}}$ OK	
7.	$\Delta, \Delta'', \overline{\Delta} \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T}/\mathtt{Y}}]\mathtt{T}_u}$	
8.	$match(sift(\overline{\mathtt{R}}, \overline{\mathtt{U}}, \overline{\mathtt{Y}}), \overline{\mathtt{P}}, \overline{\mathtt{Y}}, \overline{\mathtt{T}})$	
9.	$\Delta \vdash \exists\Delta''\mathtt{.N}$ OK	*by* **3**, **b**, **c**, *ind hyp*
10.	$\Delta, \Delta'', \overline{\mathtt{Y}\rightarrow[\bot\ \mathtt{T}_u]} \vdash \mathtt{U}$ OK	*by* **4**, **9**, **b**, *lemma 28*
11.	$\Delta \vdash \overline{\exists\Delta\mathtt{.R}}$ OK	*by* **5**, **b**, **c**, *ind hyp*
12.	$\Delta, \overline{\Delta} \vdash \overline{\mathtt{T}}$ OK	*by* **6**, **11**, **b**, **8**, *def sift*, *lemma 29*
13.	$\Delta, \Delta'', \overline{\Delta} \vdash \overline{\mathtt{T}}$ OK	*by* **12**, **7**, *lemma 9*
14.	$\Delta, \Delta'', \overline{\Delta}, \overline{\mathtt{Y}\rightarrow[\bot\ \mathtt{T}_u]} \vdash \mathtt{U}$ OK	*by* **10**, **7**, *lemma 9*
15.	$\Delta \vdash \Delta''$ OK	*by* **9**, *def* F-Exists
16.	$\Delta \vdash \overline{\Delta}$ OK	*by* **11**, *def* F-Exists
17.	$\Delta \vdash \Delta'', \overline{\Delta}$ OK	*by* **15**, **16**, *Barendregt*, *lemma 13*
18.	$\emptyset \vdash \Delta, \Delta'', \overline{\Delta}$ OK	*by* **b**, **17**, *lemma 12*
19.	$\Delta, \Delta'', \overline{\Delta} \vdash [\overline{\mathtt{T}/\mathtt{Y}}]\mathtt{U}$ OK	*by* **14**, **13**, **7**, XS-Bttm, **18**, F-Env-Empty, *lemma 19*
20.	$\Delta \vdash \Downarrow_{\Delta'',\overline{\Delta}} [\overline{\mathtt{T}/\mathtt{Y}}]\mathtt{U}$ OK	*by* **19**, **17**, *lemma 30*
21.	$\Delta \vdash \mathtt{T}$ OK	*by* **20**, **2**

□

Lemma 32 (Inversion Lemma (object creation)).

If:

a. $\Delta; \Gamma \vdash \mathtt{new\ C<}\overline{\mathcal{T}}\mathtt{,\ }\mathcal{T}\mathtt{,\ }\star\mathtt{\ >} : \mathtt{T}$

then:

$\Delta \vdash \overline{\mathcal{T}}, \mathcal{T}$ OK

$\Delta \vdash \exists\mathtt{O}\rightarrow[\bot\ \mathcal{T}]\mathtt{.C<}\overline{\mathcal{T}}\mathtt{,\ }\mathcal{T}\mathtt{,\ O>}$ OK

$\Delta \vdash \exists\mathtt{O}\rightarrow[\bot\ \mathcal{T}]\mathtt{.C<}\overline{\mathcal{T}}\mathtt{,\ }\mathcal{T}\mathtt{,\ O>} <: \mathtt{T}$

Proof *by structural induction on the derivation of* $\Delta; \Gamma \vdash \mathtt{new\ C<}\overline{\mathcal{T}}\mathtt{,\ }\mathcal{T}\mathtt{,\ }\star\mathtt{\ >} : \mathtt{T}$ *with a case analysis on the last step:*

Case 1 (T-New)

1.	$\texttt{T} = \exists\texttt{O}\rightarrow\texttt{[}\bot\ \mathcal{T}\texttt{].C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, O>}$	*by def* T-New
2.	$\Delta \vdash \overline{\mathcal{T}}, \mathcal{T}$ OK	*by premises* T-New
3.	$\Delta \vdash \exists\texttt{O}\rightarrow\texttt{[}\bot\ \mathcal{T}\texttt{].C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, O>}$ OK	*by premises* T-New
4.	*done*	*by* **3**, **2**, **1**, *reflexivity*

Case 2 (T-Subs)

1.	$\Delta; \Gamma \vdash \texttt{new C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, }\star\texttt{ >} : \texttt{U}$	*by premises* T-Subs
2.	$\Delta \vdash \texttt{U} <: \texttt{T}$	*by premises* T-Subs
3.	$\Delta \vdash \texttt{T}$ OK	*by premises* T-Subs
4.	$\Delta \vdash \overline{\mathcal{T}}, \mathcal{T}$ OK	*by* **1**, *ind hyp*
5.	$\Delta \vdash \exists\texttt{O}\rightarrow\texttt{[}\bot\ \mathcal{T}\texttt{].C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, O>}$ OK	*by* **1**, *ind hyp*
6.	$\Delta \vdash \exists\texttt{O}\rightarrow\texttt{[}\bot\ \mathcal{T}\texttt{].C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, O>} <: \texttt{U}$	*by* **1**, *ind hyp*
7.	$\Delta \vdash \exists\texttt{O}\rightarrow\texttt{[}\bot\ \mathcal{T}\texttt{].C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, O>} <: \texttt{T}$	*by* **6**, **2**, S-Trans
8.	*done*	*by* **4**, **5**, **7**

□

Lemma 33 (Inversion Lemma (field access)).

If:

a. $\Delta; \Gamma \vdash \texttt{e.f = e}' : \texttt{T}$

then:

$\Delta; \Gamma \vdash \texttt{e} : \exists\Delta'\texttt{.N}$

$\Delta \vdash \Downarrow_{\Delta'} fType(\texttt{f}, \texttt{N}) <: \texttt{T}$

Proof *by structural induction on the derivation of* $\Delta; \Gamma \vdash \texttt{e.f} : \texttt{T}$ *with a case analysis on the last step:*

Case 1 (T-Field)

1.	$\texttt{T} = \Downarrow_{\Delta'} \texttt{U}$	*by def* T-Field
2.	$\Delta; \Gamma \vdash \texttt{e} : \exists\Delta'\texttt{.N}$	*by premises* T-Field
3.	$fType(\texttt{f}, \texttt{N}) = \texttt{U}$	*by premises* T-Field
4.	*done*	*by* **2**, **3**, **1**, *reflexivity*

Case 2 (T-Subs)

1.	$\Delta; \Gamma \vdash \texttt{e.f} : \texttt{U}$	*by premises* T-Subs
2.	$\Delta \vdash \texttt{U} <: \texttt{T}$	*by premises* T-Subs
3.	$\Delta \vdash \texttt{T}$ OK	*by premises* T-Subs
4.	$\Delta; \Gamma \vdash \texttt{e} : \exists\Delta'\texttt{.N}$	*by* **1**, *ind hyp*
5.	$\Delta \vdash \Downarrow_{\Delta'} fType(\texttt{f}, \texttt{N}) <: \texttt{U}$	*by* **1**, *ind hyp*

6.	$\Delta \vdash \Downarrow_{\Delta'} fType(\mathtt{f}, \mathtt{N}) <: \mathtt{T}$	*by* **5**, **2**, S-Trans
7.	*done*	*by* **4**, **6**

□

Lemma 34 (Inversion Lemma (field assignment)).

If:

a. $\Delta; \Gamma \vdash \mathtt{e.f = e'} : \mathtt{T}$

then:

$\Delta; \Gamma \vdash \mathtt{e} : \exists\Delta'.\mathtt{N}$
$\mathtt{U} = fType(\mathtt{f}, \mathtt{N})$
$\Delta; \Gamma \vdash \mathtt{e'} : \mathtt{U'}$
$\Delta, \Delta' \vdash \mathtt{U'} <: \mathtt{U}$
$\Delta \vdash \mathtt{U'} <: \mathtt{T}$

Proof *by structural induction on the derivation of* $\Delta; \Gamma \vdash \mathtt{e.f = e'} : \mathtt{T}$ *with a case analysis on the last step:*

Case 1 (T-Assign)

1.	$\mathtt{T} = \mathtt{U'}$	*by def* T-Assign
2.	$\Delta; \Gamma \vdash \mathtt{e} : \exists\Delta'.\mathtt{N}$	*by premises* T-Assign
3.	$fType(\mathtt{f}, \mathtt{N}) = \mathtt{U}$	
4.	$\Delta; \Gamma \vdash \mathtt{e'} : \mathtt{U'}$	
5.	$\Delta, \Delta' \vdash \mathtt{U'} <: \mathtt{U}$	
6.	*done*	*by* **2**, **3**, **4**, **5**, **1**, *reflexivity*

Case 2 (T-Subs)

1.	$\Delta; \Gamma \vdash \mathtt{e.f = e'} : \mathtt{U''}$	*by premises* T-Subs
2.	$\Delta \vdash \mathtt{U''} <: \mathtt{T}$	
3.	$\Delta \vdash \mathtt{T}$ OK	
4.	$\Delta; \Gamma \vdash \mathtt{e} : \exists\Delta'.\mathtt{N}$	*by* **1**, *ind hyp*
5.	$fType(\mathtt{f}, \mathtt{N}) = \mathtt{U}$	
6.	$\Delta; \Gamma \vdash \mathtt{e'} : \mathtt{U'}$	
7.	$\Delta, \Delta' \vdash \mathtt{U'} <: \mathtt{U}$	
8.	$\Delta \vdash \mathtt{U'} <: \mathtt{U''}$	
9.	$\Delta \vdash \mathtt{U'} <: \mathtt{T}$	*by* **8**, **2**, S-Trans
10.	*done*	*by* **4**, **5**, **6**, **7**, **9**

□

Lemma 35 (Inversion Lemma (method invocation)).

If:

a. $\Delta; \Gamma \vdash \texttt{e.<}\overline{\texttt{P}}\texttt{>m(}\overline{\texttt{e}}\texttt{)} : \texttt{T}$

then:

where:

$\Delta; \Gamma \vdash \texttt{e} : \exists\Delta''\texttt{.N}$

$mType(\texttt{m, N}) = \texttt{<}\overline{\texttt{Y}\lhd\texttt{ B}}\texttt{>}\overline{\texttt{U}} \to \texttt{U}$

$\Delta; \Gamma \vdash \overline{\texttt{e} : \exists\Delta\texttt{.R}}$

$match(sift(\overline{\texttt{R}}, \overline{\texttt{U}}, \overline{\texttt{Y}}), \overline{\texttt{P}}, \overline{\texttt{Y}}, \overline{\texttt{T}})$

$\Delta \vdash \overline{\texttt{P}}$ OK

$\Delta, \Delta'', \overline{\Delta} \vdash \overline{\texttt{T} <: [\overline{\texttt{T/Y}}]\texttt{B}}$

$\Delta, \Delta'', \overline{\Delta} \vdash \overline{\exists\emptyset\texttt{.R} <: [\overline{\texttt{T/Y}}]\texttt{U}}$

$\Delta \vdash \Downarrow_{\Delta'',\overline{\Delta}} [\overline{\texttt{T/Y}}]\texttt{U} <: \texttt{T}$

Proof *by structural induction on the derivation of* $\Delta; \Gamma \vdash \texttt{e.<}\overline{\texttt{P}}\texttt{>m(}\overline{\texttt{e}}\texttt{)} : \texttt{T}$ *with a case analysis on the last step:*

Case 1 (T-Invk)

1. $\texttt{T} = \Downarrow_{\Delta'',\overline{\Delta}} [\overline{\texttt{T/Y}}]\texttt{U}$ — *by def* T-Invk

2. $\Delta; \Gamma \vdash \texttt{e} : \exists\Delta''\texttt{.N}$
3. $mType(\texttt{m, N}) = \texttt{<}\overline{\texttt{Y}\lhd\texttt{ B}}\texttt{>}\overline{\texttt{U}} \to \texttt{U}$
4. $\Delta; \Gamma \vdash \overline{\texttt{e} : \exists\Delta\texttt{.R}}$
5. $match(sift(\overline{\texttt{R}}, \overline{\texttt{U}}, \overline{\texttt{Y}}), \overline{\texttt{P}}, \overline{\texttt{Y}}, \overline{\texttt{T}})$
6. $\Delta \vdash \overline{\texttt{P}}$ OK
7. $\Delta, \Delta'', \overline{\Delta} \vdash \overline{\texttt{T} <: [\overline{\texttt{T/Y}}]\texttt{B}}$
8. $\Delta, \Delta'', \overline{\Delta} \vdash \overline{\exists\emptyset\texttt{.R} <: [\overline{\texttt{T/Y}}]\texttt{U}}$
— *by premises* T-Invk

9. *done* — *by* **2**, **3**, **4**, **5**, **6**, **7**, **8**, **1**, *reflexivity*

Case 2 (T-Subs)

1. $\Delta; \Gamma \vdash \texttt{e.<}\overline{\texttt{P}}\texttt{>m(}\overline{\texttt{e}}\texttt{)} : \texttt{U}$
2. $\Delta \vdash \texttt{U} <: \texttt{T}$
3. $\Delta \vdash \texttt{T}$ OK
— *by premises* T-Subs

4. $\Delta; \Gamma \vdash \texttt{e} : \exists\Delta''\texttt{.N}$
5. $mType(\texttt{m, N}) = \texttt{<}\overline{\texttt{Y}\lhd\texttt{ B}}\texttt{>}\overline{\texttt{U}} \to \texttt{U}$
6. $\Delta; \Gamma \vdash \overline{\texttt{e} : \exists\Delta\texttt{.R}}$
7. $match(sift(\overline{\texttt{R}}, \overline{\texttt{U}}, \overline{\texttt{Y}}), \overline{\texttt{P}}, \overline{\texttt{Y}}, \overline{\texttt{T}})$
8. $\Delta \vdash \overline{\texttt{P}}$ OK
9. $\Delta, \Delta'', \overline{\Delta} \vdash \overline{\texttt{T} <: [\overline{\texttt{T/Y}}]\texttt{B}}$
10. $\Delta, \Delta'', \overline{\Delta} \vdash \overline{\exists\emptyset\texttt{.R} <: [\overline{\texttt{T/Y}}]\texttt{U}}$
11. $\Delta \vdash \Downarrow_{\Delta'',\overline{\Delta}} [\overline{\texttt{T/Y}}]\texttt{U} <: \texttt{U}$
— *by* **1**, *ind hyp*

12.	$\Delta \vdash \Downarrow_{\Delta'',\overline{\Delta}} \texttt{[}\overline{\texttt{T/Y}}\texttt{]U} <: \texttt{T}$	*by* **11**, **2**, S-Trans
13.	*done*	*by* **4**, **5**, **6**, **7**, **8**, **9**, **10**, **12**

□

Lemma 36 (Inversion Lemma (null)).

If:

a. $\Delta; \Gamma \vdash \texttt{null} : T$

then:

$\Delta \vdash U$ OK

$\Delta \vdash U <: T$

Proof *by structural induction on the derivation of* $\Delta; \Gamma \vdash \texttt{null} : T$ *with a case analysis on the last step:*

Case 1 (T-Null)

1.	$\Delta \vdash U$ OK	*by premise* T-Null
2.	*done*	*by* **1**, *reflexivity*

Case 2 (T-Subs)

1.	$\Delta; \Gamma \vdash \texttt{null} : U'$	*by premises* T-Subs (1–3)
2.	$\Delta \vdash U' <: T$	
3.	$\Delta \vdash T$ OK	
4.	$\Delta \vdash U$ OK	*by* **1**, *ind hyp* (4–5)
5.	$\Delta \vdash U <: U'$	
6.	$\Delta \vdash U <: T$	*by* **5**, **2**, S-Trans
7.	*done*	*by* **4**, **6**

□

Lemma 37 (Inversion Lemma (cast)).

If:

a. $\Delta; \Gamma \vdash (T)e : T'$

then:

$\Delta; \Gamma \vdash e : U$

$\Delta \vdash T <: U$

$\Delta \vdash T$ OK

$\Delta \vdash T <: T'$

Proof *by structural induction on the derivation of* $\Delta; \Gamma \vdash (T)e : T'$ *with a case analysis on the last step:*

Case 1 (T-Cast)

1.	$T' = T$	*by def* T-Cast
2.	$\Delta; \Gamma \vdash e : U$	
3.	$\Delta \vdash T <: U$	*by premises* T-Cast
4.	$\Delta \vdash T$ ok	
5.	*done*	*by* **1**, **2**, **3**, **4**, *reflexivity*

Case 2 (T-Subs)

1.	$\Delta; \Gamma \vdash (T)e : U'$	
2.	$\Delta \vdash U' <: T'$	*by premises* T-Subs
3.	$\Delta \vdash T'$ ok	
4.	$\Delta; \Gamma \vdash e : U$	
5.	$\Delta \vdash T <: U$	*by* **1**, *ind hyp*
6.	$\Delta \vdash T$ ok	
7.	$\Delta \vdash T <: U'$	
8.	$\Delta \vdash T <: T'$	*by* **7**, **2**, S-Trans
9.	*done*	*by* **4**, **5**, **6**, **8**

□

Lemma 38 (Subclassing gives extended subclassing).

If:

a. $\vdash \texttt{R}' \sqsubset\!\!: \texttt{R}$

then:

$\Delta \vdash \exists\Delta'.\texttt{R}' \sqsubset: \exists\Delta'.\texttt{R}$

Proof *by structural induction on the derivation of* $\vdash \texttt{R}' \sqsubset\!\!: \texttt{R}$ *with a case analysis on the last step:*

Case 1 (SC-Reflex)

1.	$\texttt{R}' = \texttt{R}$	*by def* SC-Reflex
2.	$\Delta \vdash \exists\Delta'.\texttt{R}' \sqsubset: \exists\Delta'.\texttt{R}$	*by* **1**, XS-Reflex

Case 2 (SC-Sub-Class)

1.	$\texttt{R}' = \texttt{C<}\overline{\texttt{T}}\texttt{>}$	*by def* SC-Sub-Class
2.	$\texttt{R} = [\overline{\texttt{T/X}}]\texttt{N}$	
3.	$\texttt{class C<}\overline{\texttt{X...}}\texttt{>} \lhd \texttt{N ...}$	*by premise* SC-Sub-Class
4.	$\Delta \vdash \exists\Delta'.\texttt{R}' \sqsubset: \exists\Delta'.\texttt{R}$	*by* **1**, **2**, **3**, XS-Sub-Class

Case 3 (SC-Trans)

1.	$\vdash \mathtt{R}' \sqsubset\!\!\sqsubset: \mathtt{R}''$	} *by premises* SC-TRANS
2.	$\vdash \mathtt{R}'' \sqsubset\!\!\sqsubset: \mathtt{R}$	
3.	$\Delta \vdash \exists\Delta'.\mathtt{R}' \sqsubset: \exists\Delta'.\mathtt{R}''$	*by* **1**, *ind hyp*
4.	$\Delta \vdash \exists\Delta'.\mathtt{R}'' \sqsubset: \exists\Delta'.\mathtt{R}$	*by* **2**, *ind hyp*
5.	$\Delta \vdash \exists\Delta'.\mathtt{R}' \sqsubset: \exists\Delta'.\mathtt{R}$	*by* **3**, **4**, XS-TRANS

□

Lemma 39 (Extended subclassing gives subclassing).

If:

a. $\Delta \vdash \exists\Delta'.\mathtt{R}' \sqsubset: \exists\overline{\mathtt{X}\to[\mathtt{B}_l\ \mathtt{B}_u]}.\mathtt{R}$

b. $\Delta \vdash$ OK

then:

there exists $\overline{\mathtt{T}}$

where:

$\vdash \mathtt{R}' \sqsubset\!\!\sqsubset: [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{R}$

$\Delta, \Delta' \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T}/\mathtt{X}}]\mathtt{B}_u}$

$\Delta, \Delta' \vdash \overline{[\overline{\mathtt{T}/\mathtt{X}}]\mathtt{B}_l <: \mathtt{T}}$

$fv(\overline{\mathtt{T}}) \subseteq dom(\Delta, \Delta')$

Proof *by structural induction on the derivation of* $\Delta \vdash \exists\Delta'.\mathtt{R}' \sqsubset: \exists\overline{\mathtt{X}\to[\mathtt{B}_l\ \mathtt{B}_u]}.\mathtt{R}$ *with a case analysis on the last step:*

Case 1 (XS-REFLEX)

Easy, using SC-REFLEX, $\overline{\mathtt{T}} = \overline{\mathtt{X}}$ *and* S-BOUND.

Case 2 (XS-TRANS)

1.	$\Delta \vdash \exists\Delta'.\mathtt{R}' \sqsubset: \mathtt{B}$	} *by premises* XS-TRANS
2.	$\Delta \vdash \mathtt{B} \sqsubset: \exists\overline{\mathtt{X}\to[\mathtt{B}_l\ \mathtt{B}_u]}.\mathtt{R}$	
3.	$\mathtt{B} = \exists\overline{\mathtt{X}'\to[\mathtt{B}'_l\ \mathtt{B}'_u]}.\mathtt{R}''$	*by* **1** *gives* $\mathtt{B} \neq \bot$
4.	*wlog assume* $\overline{\mathtt{X}'}$ *are fresh*	*by* **3**, *Barendregt*
5.	*there exists* $\overline{\mathtt{U}'}$	} *by* **1**, **3**, **b**, *ind hyp*
6.	$\vdash \mathtt{R}' \sqsubset\!\!\sqsubset: [\overline{\mathtt{U}'/\mathtt{X}'}]\mathtt{R}''$	
7.	$\Delta, \Delta' \vdash \overline{\mathtt{U}' <: [\overline{\mathtt{U}'/\mathtt{X}'}]\mathtt{B}_u}$	
8.	$\Delta, \Delta' \vdash \overline{[\overline{\mathtt{U}'/\mathtt{X}'}]\mathtt{B}_l <: \mathtt{U}'}$	
9.	$fv(\overline{\mathtt{U}'}) \subseteq dom(\Delta, \Delta')$	
10.	*there exists* $\overline{\mathtt{U}}$	} *by* **2**, **3**, **b**, *ind hyp*
11.	$\vdash \mathtt{R}'' \sqsubset\!\!\sqsubset: [\overline{\mathtt{U}/\mathtt{X}}]\mathtt{R}$	
12.	$\Delta, \overline{\mathtt{X}'\to[\mathtt{B}'_l\ \mathtt{B}'_u]} \vdash \overline{\mathtt{U} <: [\overline{\mathtt{U}/\mathtt{X}}]\mathtt{B}'_u}$	
13.	$\Delta, \overline{\mathtt{X}'\to[\mathtt{B}'_l\ \mathtt{B}'_u]} \vdash \overline{[\overline{\mathtt{U}/\mathtt{X}}]\mathtt{B}'_l <: \mathtt{U}}$	
14.	$fv(\overline{\mathtt{U}}) \subseteq dom(\Delta), \overline{\mathtt{X}'}$	

15.	$\vdash [\overline{\texttt{U}'/\texttt{X}'}]\texttt{R}'' \sqsubset: [\overline{\texttt{U}'/\texttt{X}'}][\overline{\texttt{U}/\texttt{X}}]\texttt{R}$	*by* **11**, *lemma 1*
16.	$\vdash \texttt{R}' \sqsubset: [\overline{\texttt{U}'/\texttt{X}'}][\overline{\texttt{U}/\texttt{X}}]\texttt{R}$	*by* **6**, **15**, SC-Trans
17.	$\vdash \texttt{R}' \sqsubset: [\overline{[\overline{\texttt{U}'/\texttt{X}'}]\texttt{U}/\texttt{X}}]\texttt{R}$	*by* **16**, **4**
18.	$\Delta, \Delta', \overline{\texttt{X}' \to [\texttt{B}'_l\ \texttt{B}'_u]} \vdash \overline{\texttt{U} <: [\overline{\texttt{U}/\texttt{X}}]\texttt{B}_u}$	*by* **12**, **4**, *lemma 8*
19.	$\Delta, \Delta', \overline{\texttt{X}' \to [\texttt{B}'_l\ \texttt{B}'_u]} \vdash \overline{[\overline{\texttt{U}/\texttt{X}}]\texttt{B}_l <: \texttt{U}}$	*by* **13**, **4**, *lemma 8*
20.	$\Delta, \Delta' \vdash \overline{[\overline{\texttt{U}'/\texttt{X}'}]\texttt{U} <: [\overline{\texttt{U}'/\texttt{X}'}][\overline{\texttt{U}/\texttt{X}}]\texttt{B}_u}$	*by* **18**, **7**, **8**, **b**, *lemma 17*
21.	$\Delta, \Delta' \vdash \overline{[\overline{\texttt{U}'/\texttt{X}'}]\texttt{U} <: [\overline{[\overline{\texttt{U}'/\texttt{X}'}]\texttt{U}/\texttt{X}}]\texttt{B}_u}$	*by* **20**, **4**
22.	$\Delta, \Delta' \vdash \overline{[\overline{\texttt{U}'/\texttt{X}'}][\overline{\texttt{U}/\texttt{X}}]\texttt{B}_l <: [\overline{\texttt{U}'/\texttt{X}'}]\texttt{U}}$	*by* **19**, **7**, **8**, **b**, *lemma 17*
23.	$\Delta, \Delta' \vdash \overline{[\overline{[\overline{\texttt{U}'/\texttt{X}'}]\texttt{U}/\texttt{X}}]\texttt{B}_l <: [\overline{\texttt{U}'/\texttt{X}'}]\texttt{U}}$	*by* **22**, **4**
24.	$fv(\overline{[\overline{\texttt{U}'/\texttt{X}'}]\texttt{U}}) \subseteq dom(\Delta, \Delta')$	*by* **9**, **14**
25.	*let* $\overline{\texttt{T}} = \overline{[\overline{\texttt{U}'/\texttt{X}'}]\texttt{U}}$	
26.	*done*	*by* **25**, **17**, **21**, **23**, **24**

Case 3 (XS-Env)

1.	$\texttt{R} = \texttt{N}$	*by def* XS-Env
2.	$\texttt{R}' = [\overline{\texttt{U}/\texttt{X}}]\texttt{N}$	
3.	$\Delta, \Delta' \vdash \overline{\texttt{U} <: [\overline{\texttt{U}/\texttt{X}}]\texttt{B}_u}$	*by premises* XS-Env
4.	$\Delta, \Delta' \vdash \overline{[\overline{\texttt{U}/\texttt{X}}]\texttt{B}_l <: \texttt{U}}$	
5.	$dom(\Delta') \cap fv(\exists\overline{\texttt{X}\to[\texttt{B}_l\ \texttt{B}_u]}.\texttt{N}) = \emptyset$	
6.	$fv(\overline{\texttt{U}}) \subseteq dom(\Delta, \Delta')$	
7.	$\vdash \texttt{N} \sqsubset: \texttt{N}$	*by* SC-Reflex
8.	$\vdash [\overline{\texttt{U}/\texttt{X}}]\texttt{N} \sqsubset: [\overline{\texttt{U}/\texttt{X}}]\texttt{N}$	*by* **7**, *lemma 1*
9.	$\vdash \texttt{N}' \sqsubset: [\overline{\texttt{U}/\texttt{X}}]\texttt{N}$	*by* **8**, **2**
10.	*let* $\overline{\texttt{T}} = \overline{\texttt{U}}$	
11.	*done*	*by* **10**, **9**, **3**, **4**, **6**

Case 4 (XS-Sub-Class)

1.	$\Delta' = \overline{\texttt{X}\to[\texttt{B}_l\ \texttt{B}_u]}$	*by def* XS-Sub-Class
2.	$\texttt{R}' = \texttt{C<}\overline{\texttt{U}}\texttt{>}$	
3.	$\texttt{R} = [\overline{\texttt{U}/\texttt{Y}}]\texttt{N}''$	
4.	$\texttt{class C<}\overline{\texttt{Y...}}\texttt{>} \lhd \texttt{N}''\texttt{...}$	*by premise* XS-Sub-Class
5.	$\vdash \texttt{C<}\overline{\texttt{U}}\texttt{>} \sqsubset: [\overline{\texttt{U}/\texttt{Y}}]\texttt{N}''$	*by* **4**, SC-Sub-Class
6.	$\vdash \texttt{R}' \sqsubset: \texttt{R}$	*by* **5**, **2**, **3**
7.	*let* $\overline{\texttt{T}} = \overline{\texttt{X}}$	
8.	*done*	*by* **6**, **7**, S-Bound, **1**

Case 5 (XS-Bottom)

N/A

$\square$

Lemma 40 (Subclassing preserves *match*ing (receiver)).

If:

a.	$\Delta \vdash \exists\Delta_1 . N_1 \sqsubset: \exists\Delta_2 . N_2$
b.	$mType(m, N_2) = <\overline{Y_2 \rightarrow [B_{2l}\ B_{2u}]}>\overline{U_2}\rightarrow U_2$
c.	$mType(m, N_1) = <\overline{Y_1 \rightarrow [B_{1l}\ B_{1u}]}>\overline{U_1}\rightarrow U_1$
d.	$match(sift(\overline{R}, \overline{U_2}, \overline{Y_2}), \overline{P}, \overline{Y_2}, \overline{T})$
e.	$\emptyset \vdash \Delta$ OK
f.	$\Delta, \Delta' \vdash \overline{T}$ OK

then:

$match(sift(\overline{R}, \overline{U_1}, \overline{Y_1}), \overline{P}, \overline{Y_1}, \overline{T})$

Proof

1.	$\vdash N_1 \sqsubset: [\overline{T'/X}]N_2$	*by* **a**, **e**, *lemma 39*
2.	$\Delta_2 = \overline{X \rightarrow [B_l\ B_u]}$	*by* **a**, **e**, *lemma 39*
3.	$\Delta, \Delta_1 \vdash \overline{T' <: [\overline{T'/X}]B_u}$	*by* **a**, **e**, *lemma 39*
4.	$\Delta, \Delta_1 \vdash \overline{[\overline{T'/X}]B_l <: T'}$	*by* **a**, **e**, *lemma 39*
5.	*assume wlog* $\overline{X} \cap \overline{Y_2} = \emptyset$	*by* **b**
6.	*assume wlog* $fv(\overline{T'}) \cap \overline{Y_2} = \emptyset$	*by* **b**
7.	$mType(m, [\overline{T'/X}]N_2) =$ $[\overline{T'/X}]<\overline{Y_2 \rightarrow [B_{2l}\ B_{2u}]}>\overline{U_2}\rightarrow U_2$	*by* **b**, *lemma 6*
8.	$mType(m, N_1) = [\overline{T'/X}]<\overline{Y_2 \rightarrow [B_{2l}\ B_{2u}]}>\overline{U_2}\rightarrow U_2$	*by* **1**, **7**, *lemma 24*
9.	$\overline{Y_1} = \overline{Y_2}$	*by* **8**
10.	$\overline{U_1} = [\overline{T'/X}]\overline{U_2}$	*by* **8**
11.	*let* $sift(\overline{R}, \overline{U_2}, \overline{Y_2}) = \langle\overline{R''}, \overline{\exists\Delta . R'}\rangle$	*by* **d**, *def* sift
12.	$\forall i$ *where* $P_i \neq \star : T_i = P_i$	*by premises of* match, **d**, **11**
13.	$\forall j$ *where* $P_j = \star : Y_{2j} \in fv(\overline{R'})$	*by premises of* match, **d**, **11**
14.	$\vdash \overline{R'' \sqsubset: [\overline{T/Y_2}, \overline{T''/Z}]R'}$	*by premises of* match, **d**, **11**
15.	$dom(\overline{\Delta}) = \overline{Z}$	*by premises of* match, **d**, **11**
16.	$fv(\overline{T_2}, \overline{T''}) \cap \overline{Y_2}, \overline{Z} = \emptyset$	*by premises of* match, **d**, **11**
17.	$\vdash \overline{[\overline{T'/X}]R'' \sqsubset: [\overline{T'/X}][\overline{T/Y_2}, \overline{T''/Z}]R'}$	*by* **14**, *lemma 1*
18.	$\overline{X} \cap fv(\overline{R''}) = \emptyset$	*by Barendregt*
19.	$\vdash \overline{R'' \sqsubset: [\overline{T'/X}][\overline{T/Y_2}, \overline{T''/Z}]R'}$	*by* **17**, **18**
20.	$\overline{Z} \cap fv(\overline{T'}) = \emptyset$	*by* **15**, **11**, *Barendregt*
21.	$\vdash \overline{R'' \sqsubset: [\overline{[\overline{T'/X}]T/Y_2}, \overline{[\overline{T'/X}]T''/Z}][\overline{T'/X}]R'}$	*by* **19**, **6**, **20**
22.	$\forall j$ *where* $P_j = \star : Y_{2j} \in fv(\overline{[\overline{T'/X}]R'})$	*by* **13**, **5**
23.	$fv(\overline{[\overline{T'/X}]T}, \overline{[\overline{T'/X}]T''}) \cap \overline{Y_2}, \overline{Z} = \emptyset$	*by* **16**,**6**,**20**
24.	$match(\langle\overline{R''}, \overline{[\overline{T'/X}]\exists\Delta . R'}\rangle, \overline{P}, \overline{Y_2}, \overline{[\overline{T'/X}]T})$	*by* **12**, **22**, **21**, **15**, **23**, *def* match

25. $sift(\overline{\mathtt{R}}, \overline{[\overline{\mathtt{T'/X}}]\mathtt{U}_2}, \overline{\mathtt{Y}_2}) = \langle \overline{\mathtt{R''}}, \overline{[\overline{\mathtt{T'/X}}]\exists\Delta\,.\mathtt{R'}} \rangle$ — *by* **11**, **5**, **6**, *lemma 3*

26. $sift(\overline{\mathtt{R}}, \overline{\mathtt{U}_1}, \overline{\mathtt{Y}_1}) = \langle \overline{\mathtt{R''}}, \overline{[\overline{\mathtt{T'/X}}]\exists\Delta\,.\mathtt{R'}} \rangle$ — *by* **25**, **9**, **10**

27. $match(sift(\overline{\mathtt{R}}, \overline{\mathtt{U}_1}, \overline{\mathtt{Y}_1}), \overline{\mathtt{P}}, \overline{\mathtt{Y}_1}, \overline{[\overline{\mathtt{T'/X}}]\mathtt{T}})$ — *by* **24**, **26**, **9**

28. $match(sift(\overline{\mathtt{R}}, \overline{\mathtt{U}_1}, \overline{\mathtt{Y}_1}), \overline{\mathtt{P}}, \overline{\mathtt{Y}_1}, \overline{\mathtt{T}})$ — *by* **27**, **f**, **2**, *Barendregt*

□

Lemma 41 (Subclassing preserves *match*ing (arguments)).

If:

a. $\Delta \vdash \overline{\exists\Delta_1\,.\mathtt{R}_1 \sqsubset: \exists\Delta_2\,.\mathtt{R}_2}$

b. $match(sift(\overline{\mathtt{R}_2}, \overline{\mathtt{U}}, \overline{\mathtt{Y}}), \overline{\mathtt{P}}, \overline{\mathtt{Y}}, \overline{\mathtt{T}})$

c. $fv(\overline{\mathtt{U}}) \cap \overline{\mathtt{Z}} = \emptyset$

d. $\overline{\Delta_2} = \overline{\mathtt{Z} \to [\mathtt{B}_l\ \ \mathtt{B}_u]}$

e. $\emptyset \vdash \Delta$ OK

f. $\Delta \vdash \overline{\exists\Delta_1\,.\mathtt{R}_1}$ OK

g. $\Delta \vdash \overline{\mathtt{P}}$ OK

then:

there exists $\overline{\mathtt{U'}}$

where:

$match(sift(\overline{\mathtt{R}_1}, \overline{\mathtt{U}}, \overline{\mathtt{Y}}), \overline{\mathtt{P}}, \overline{\mathtt{Y}}, \overline{[\overline{\mathtt{U'/Z}}]\mathtt{T}})$

$\Delta, \overline{\Delta_1} \vdash \overline{\mathtt{U'} <: [\overline{\mathtt{U'/Z}}]\mathtt{B}_u}$

$\Delta, \overline{\Delta_1} \vdash \overline{[\overline{\mathtt{U'/Z}}]\mathtt{B}_l <: \mathtt{U'}}$

$\vdash \overline{\mathtt{R}_1} \sqsubset: \overline{[\overline{\mathtt{U'/Z}}]\mathtt{R}_2}$

$fv(\overline{\mathtt{U'}}) \subseteq \Delta, \overline{\Delta_1}$

Proof

1. *let* $sift(\overline{\mathtt{R}_2}, \overline{\mathtt{U}}, \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}'_2}, \overline{\exists\Delta_3\,.\mathtt{R}_3} \rangle$

2. $\overline{\mathtt{R}'_1}$ *and* $\overline{\mathtt{R}'_2}$ *are subsequences of* $\overline{\mathtt{R}_1}$ *and* $\overline{\mathtt{R}_2}$ *respectively*

3. *Take* Δ'_1 *and* Δ'_2 *to be the corresponding environments of* $\overline{\mathtt{R}'_1}$ *and* $\overline{\mathtt{R}'_2}$

4. $sift(\overline{\mathtt{R}_1}, \overline{\mathtt{U}}, \overline{\mathtt{Y}}) = \langle \overline{\mathtt{R}'_1}, \overline{\exists\Delta_3\,.\mathtt{R}_3} \rangle$

5. $\Delta \vdash \overline{\exists\Delta'_1\,.\mathtt{R}'_1 \sqsubset: \exists\Delta'_2\,.\mathtt{R}'_2}$

(4–5) *by* **1**, **a**, **2**, **3**, *lemma 4*

6. *there exists* $\overline{\mathtt{U'}}$

7. $\vdash \overline{\mathtt{R}_1} \sqsubset: \overline{[\overline{\mathtt{U'/Z}}]\mathtt{R}_2}$

8. $\Delta, \overline{\Delta_1} \vdash \overline{\mathtt{U'} <: [\overline{\mathtt{U'/Z}}]\mathtt{B}_u}$

9. $\Delta, \overline{\Delta_1} \vdash \overline{[\overline{\mathtt{U'/Z}}]\mathtt{B}_l <: \mathtt{U'}}$

10. $fv(\overline{\mathtt{U'}}) \subseteq dom(\Delta, \overline{\Delta_1})$

(6–10) *by* **a**, **e**, *lemma 39*

11. $fv(\overline{\exists\Delta_3\,.\mathtt{R}_3}) \cap \overline{\mathtt{Z}} = \emptyset$ — *by* **c**, *def* sift

12. $fv(\overline{\mathtt{R}_3}) \cap \overline{\mathtt{Z}} = \emptyset$ — *by* **11**, *Barendregt*

13.	$\forall i$ *where* $P_i \neq \star : T_i = P_i$	*by* **b**, **1**, *def* match
14.	$\forall j$ *where* $P_j = \star : Y_j \in fv(\overline{R_3})$	
15.	$\vdash \overline{R'_2 \sqsubset: [\overline{T/Y},\overline{T'/X}]R_3}$	
16.	$dom(\overline{\Delta_3}) = \overline{X}$	
17.	$fv(\overline{T},\overline{T'}) \cap \overline{Y},\overline{X} = \emptyset$	
18.	$\vdash \overline{[\overline{U'/Z}]R'_2 \sqsubset: [\overline{U'/Z}][\overline{T/Y},\overline{T'/X}]R_3}$	*by* **15**, *lemma 1*
19.	$\vdash \overline{R'_1 \sqsubset: [\overline{U'/Z}][\overline{T/Y},\overline{T'/X}]R_3}$	*by* **7**, **18**, SC-TRANS
20.	$\vdash \overline{R'_1 \sqsubset: [\overline{[\overline{U'/Z}]T/Y},\overline{[\overline{U'/Z}]T'/X}]R_3}$	*by* **12**, **19**
21.	$\Delta,\overline{\Delta_1} \vdash \overline{R_1}$ OK	*by* **f**, *def* F-EXIST
22.	$\Delta \vdash \overline{\Delta_1}$ OK	
23.	$\emptyset \vdash \Delta,\overline{\Delta_1}$ OK	*by* **22**, **e**, *lemma 12*
24.	$\Delta,\overline{\Delta_1} \vdash \overline{[\overline{U'/Z}]R_2}$ OK	*by* **7**, **f**, **e**, *lemma 22*
25.	$fv(\overline{U'}) \cap \overline{X} = \emptyset$	*by* **16**, *Barendregt*
26.	$fv(\overline{U'}) \cap \overline{Y} = \emptyset$	*by* **10**
27.	$fv(\overline{[\overline{U'/Z}]T}, \overline{[\overline{U'/Z}]T'}) \cap \overline{Y},\overline{X} = \emptyset$	*by* **17**, **25**, **26**
28.	$\forall i$ *where* $P_i \neq \star : [\overline{U'/Z}]T_i = [\overline{U'/Z}]P_i = P_i$	*by* **14**, **g**, **d**, *Barendregt*
29.	$match(\langle\overline{R'_1}, \overline{\exists\Delta_3.R_3}\rangle, \overline{P}, \overline{Y}, \overline{[\overline{U'/Z}]T})$	*by* **28**, **14**, **20**, **16**, **27**, *def* match
30.	$match(sift(\overline{R_1},\overline{U},\overline{Y}), \overline{P}, \overline{Y}, \overline{[\overline{U'/Z}]T})$	*by* **29**, **4**
31.	*done*	*by* **30**, **8**, **9**, **7**, **10**

□

Lemma 42 (Method body is well typed).

If:

a. $\emptyset \vdash \Delta$ OK

b. $\Delta \vdash C<\overline{\mathcal{T}}>$ OK

c. $mType(m,\ C<\overline{\mathcal{T}}>) = <\overline{Y \lhd U_u}>\overline{U} \to U$

d. $mBody(m,\ C<\overline{\mathcal{T}}>) = (\overline{x}; e)$

then:

$\Delta, \overline{Y \to [\bot\ U_u]}; \overline{x:U}, this:\exists\emptyset.C<\overline{\mathcal{T}}> \vdash e : U$

Proof *by induction on the derivation of* $mBody(m,\ C<\overline{\mathcal{T}}>) = (\overline{x}; e)$ *with a case analysis on the last step:*

Case 1 *Base case*

1.	class $C<\overline{\mathcal{X} \lhd T_u}> \lhd N \ \ldots\overline{M}\ldots$	*by premises mBody*
2.	$<\overline{Y' \lhd U'_u}>U'\ m(\overline{U'\ x})\ \{return\ e_0\} \in \overline{M}$	
3.	$e = [\overline{\mathcal{T}/\mathcal{X}}]e_0$	*by def mBody*
4.	$<\overline{Y \lhd U_u}>\overline{U} \to U = [\overline{\mathcal{T}/\mathcal{X}}]<\overline{Y' \lhd U'_u}>\overline{U'} \to U'$	*by* **1**, **2**, *mType*
5.	$let \overline{\mathcal{X}} = \overline{X}, \overline{O}, O_o, O_t$	

6. $let\overline{\mathcal{T}} = \overline{\mathtt{U}}, \overline{\tau}, \tau_o, \tau_t$

7. $\vdash \mathtt{class\ C<}\overline{\mathcal{X} \lhd \mathtt{U}_u}\mathtt{>} \lhd \mathtt{N}\ \ldots \overline{\mathtt{M}} \ldots\ \textsc{ok}$ — *by* **1**, *wf-prog*

8. $\overline{\mathtt{X}\to[\bot\ \mathtt{T}_u]}, \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \to [\bot\ \mathtt{T}_u], \mathtt{O}_t \to [\bot\ \mathtt{T}_u] \vdash \mathtt{<}\overline{\mathtt{Y}' \lhd \mathtt{U}'_u}\mathtt{>T}'\ \mathtt{m(}\overline{\mathtt{U}'\ \mathtt{x}}\mathtt{)}\ \{\mathtt{return\ e_0}\}\ \textsc{ok}$ — *by* **7**, *def* T-Class

9. $\overline{\mathtt{X}\to[\bot\ \mathtt{T}_u]}, \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \to [\bot\ \mathtt{T}_u], \mathtt{O}_t \to [\bot\ \mathtt{T}_u], \overline{\mathtt{Y}' \to [\bot\ \mathtt{U}'_u]}; \overline{\mathtt{x:U}'}, \mathtt{this:C<}\overline{\mathcal{X}}\mathtt{>} \vdash \mathtt{e}_0 : \mathtt{U}'$ — *by* **8**, *def* T-Method

10. $\Delta, \overline{\mathtt{X}\to[\bot\ \mathtt{T}_u]}, \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \to [\bot\ \mathtt{T}_u], \mathtt{O}_t \to [\bot\ \mathtt{T}_u], \overline{\mathtt{Y}' \to [\bot\ \mathtt{U}'_u]}; \overline{\mathtt{x:U}'}, \mathtt{this:C<}\overline{\mathcal{X}}\mathtt{>} \vdash \mathtt{e}_0 : \mathtt{U}'$ — *by* **9**, **1**, **2**, *distinctness of formal variables, lemma 10*

11. $\overline{\mathtt{X}\to[\bot\ \mathtt{T}_u]}, \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \to [\bot\ \mathtt{T}_u], \mathtt{O}_t \to [\bot\ \mathtt{T}_u] \vdash \overline{\mathtt{Y}' \to [\bot\ \mathtt{U}'_u]}\ \textsc{ok}$ — *by* **8**, *def* T-Method

12. $\Delta, \overline{\mathtt{X}\to[\bot\ \mathtt{T}_u]}, \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \to [\bot\ \mathtt{T}_u], \mathtt{O}_t \to [\bot\ \mathtt{T}_u] \vdash \overline{\mathtt{Y}' \to [\bot\ \mathtt{U}'_u]}\ \textsc{ok}$ — *by* **11**, **1**, *distinctness of formal variables, lemma 9*

13. $\Delta \vdash \overline{\mathcal{T}}\ \textsc{ok}$

14. $\Delta \vdash \overline{\mathcal{T} <: [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{T}_u}$

15. $\forall \tau \in \overline{\tau}.\ \Delta \vdash \tau_o <: \tau$

(13–15) *by* **b**, *def* F-Class

16. $[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o = \tau_o$ — *by* **5**, **6**

17. $\Delta \vdash \overline{[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o <: \tau}$ — *by* **15**, **16**

18. $\Delta, \overline{\mathtt{Y}' \to [\bot\ [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{U}'_u]}; \overline{\mathtt{x}:[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{U}'}, \mathtt{this}:\exists\emptyset.[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{C<}\overline{\mathcal{X}}\mathtt{>} \vdash [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{e}_0 : [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{U}'$ — *by* **10**, **13**, **14**, XS-Bttm, **17**, **a**, **12**, **1**, *distinctness of formal variables, lemma 21*

19. $\Delta, \overline{\mathtt{Y}\to[\bot\ \mathtt{U}_u]}; \overline{\mathtt{x:U}}, \mathtt{this}:\exists\emptyset.\mathtt{C<}\overline{\mathcal{T}}\mathtt{>} \vdash \mathtt{e} : \mathtt{U}$ — *by* **18**, **4**

***Case** 2 Inductive case*

1. $\mathtt{class\ C<}\overline{\mathcal{X} \lhd \mathtt{T}_u}\mathtt{>} \lhd \mathtt{N}\ \ldots \overline{\mathtt{M}} \ldots$

2. $\mathtt{m} \notin \overline{\mathtt{M}}$

(1–2) *by premises mBody*

3. $(\overline{\mathtt{x}}, \mathtt{e}) = mBody(\mathtt{m}, [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{N})$ — *by def mBody*

4. $\mathtt{<}\overline{\mathtt{Y} \lhd \mathtt{U}_u}\mathtt{>}\overline{\mathtt{U}}\to\mathtt{U} = mType(\mathtt{m}, [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{N})$ — *by* **1**, **2**, *mType*

5. $let\overline{\mathcal{X}} = \overline{\mathtt{X}}, \overline{\mathtt{O}}, \mathtt{O}_o, \mathtt{O}_t$

6. $let\overline{\mathcal{T}} = \overline{\mathtt{U}}, \overline{\tau}, \tau_o, \tau_t$

7. $\vdash \mathtt{class\ C<}\overline{\mathcal{X} \lhd \mathtt{T}_u}\mathtt{>} \lhd \mathtt{N}\ \ldots \overline{\mathtt{M}} \ldots\ \textsc{ok}$ — *by* **1**, *wf-prog*

8. $\overline{\mathtt{X}\to[\bot\ \mathtt{T}_u]}, \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \to [\bot\ \mathtt{T}_u], \mathtt{O}_t \to [\bot\ \mathtt{T}_u] \vdash \mathtt{N}\ \textsc{ok}$ — *by* **7**, *def* T-Class

9. $\Delta, \overline{\mathtt{X}\to[\bot\ \mathtt{T}_u]}, \overline{\mathtt{O}\to[\mathtt{O}_o\ \mathtt{T}_u]}, \mathtt{O}_o \to [\bot\ \mathtt{T}_u], \mathtt{O}_t \to [\bot\ \mathtt{T}_u] \vdash \mathtt{N}\ \textsc{ok}$ — *by* **8**, **1**, *distinctness of formal variables, lemma 9*

10. $\Delta \vdash \overline{\mathcal{T}}\ \textsc{ok}$

11. $\Delta \vdash \overline{\mathcal{T} <: [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{T}_u}$

12. $\forall \tau \in \overline{\tau}.\ \Delta \vdash \tau_o <: \tau$

(10–12) *by* **b**, *def* F-Class

13. $[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o = \tau_o$ — *by* **5**, **6**

14. $\Delta \vdash \overline{[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{O}_o <: \tau}$ — *by* **12**, **13**

15. $\Delta \vdash [\overline{\mathcal{T}/\mathcal{X}}]\mathtt{N}\ \textsc{ok}$ — *by* **9**, **a**, **10**, **11**, XS-Bttm, **14**, *lemma 18*

16. $\Delta, \overline{\mathtt{Y}\to[\bot\ \mathtt{U}_u]}; \overline{\mathtt{x:U}}, \mathtt{this}:\exists\emptyset.[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{N} \vdash \mathtt{e} : \mathtt{U}\emptyset$ — *by* **3**, **4**, **a**, **15**, *ind hyp*

17. $\Delta, \overline{\mathtt{Y}\to[\bot\ \mathtt{U}_u]} \vdash \exists\emptyset.\mathtt{C<}\overline{\mathcal{T}}\mathtt{>} <: \exists\emptyset.[\overline{\mathcal{T}/\mathcal{X}}]\mathtt{N}$ — *by* **1**, SC-Sub-Class

18. $\Delta, \overline{\mathtt{Y}\to[\bot\ \mathtt{U}_u]}; \overline{\mathtt{x:U}}, \mathtt{this}:\exists\emptyset.\mathtt{C<}\overline{\mathcal{T}}\mathtt{>} \vdash \mathtt{e} : \mathtt{U}\emptyset$ — *by* **16**, **17**, **15**, *lemma 26*

$\square$

Lemma 43 **($mType$ defined gives $mBody$ defined).**

If:

a. $mType(\mathtt{m}, \mathtt{C<}\overline{\mathtt{T}}\mathtt{>})\ defined$

then:

$mBody(\mathtt{m}, \mathtt{C<}\overline{\mathtt{T}}\mathtt{>})\ defined$

Proof *by case analaysis on the defintion of* $mType(\mathtt{m}, \mathtt{C<}\overline{\mathtt{T}}\mathtt{>})$

Case 1 *Base case*

1.	`class C...{...`$\overline{\mathtt{M}}$`}`	*by premises mType*
2.	$\mathtt{m} \in \overline{\mathtt{M}}$	*by premises mType*
3.	$mBody(m, \mathtt{C<}\overline{\mathtt{T}}\mathtt{>})\ defined$	*by* **1**, **2**, *base case of def mBody*

Case 2 *Inductive case*

1.	`class C...{...`$\overline{\mathtt{M}}$`}`	*by premises mType*
2.	$\mathtt{m} \notin \overline{\mathtt{M}}$	*by premises mType*
3.	$mBody(m, \mathtt{C<}\overline{\mathtt{T}}\mathtt{>})\ defined$	*by* **1**, **2**, *ind case of def mBody*

$\square$

Lemma 44 **($fType$ and $fields$ related).**

a. $fType(\mathtt{f}, \mathtt{C<}\overline{\mathtt{T}}\mathtt{>})\ defined$

b. $fields(\mathtt{C}) = \overline{\mathtt{f}}$

then:

$\mathtt{f} \in \overline{\mathtt{f}}$

Proof *by induction on the derivation of* $fType(\mathtt{f}, \mathtt{N})$ *with a case analysis on the last step:*

Case 1 (BASE CASE)

1.	`class C...`$\lhd$ `D<`$\overline{\mathtt{U}}$`>` $\{\overline{\mathtt{T\ f'}};\ldots\}$	*by premises fType*
2.	$\mathtt{f} \in \overline{\mathtt{f'}}$	*by premises fType*
3.	$fields(\mathtt{C}) = \overline{\mathtt{f'}}, fields(\mathtt{D})$	*by def fType*
4.	$\mathtt{f} \in \overline{\mathtt{f}}$	*by* **2**, **3**

Case 2 (INDUCTIVE CASE)

1.	`class` $\mathtt{C<\overline{X...}>} \lhd \mathtt{D<\overline{U}>}\ \{\overline{\mathtt{T\ f';...}}\}$	*by premises fType*
2.	$\mathtt{f} \notin \overline{\mathtt{f'}}$	
3.	$fType(\mathtt{f},\mathtt{N}) = fType(\mathtt{f},\mathtt{D<[\overline{T/X}]\overline{U}>})$	*by def fType*
4.	$fields(\mathtt{C}) = \overline{\mathtt{f'}}, fields(\mathtt{D})$	*by def fType*
5.	$\mathtt{f} \in fields(\mathtt{D}$	*by* **3**, **4**, *ind hyp*
6.	$\mathtt{f} \in \overline{\mathtt{f}}$	*by* **5**, **4**

□

Lemma 45 (Inversion lemma: locations).

If:

a. $\Delta; \mathcal{H} \vdash \iota : \mathtt{T}$

then:

$\mathcal{H}(\iota) = \{\mathtt{N}; ...\}$
$\Delta \vdash \exists\emptyset.\mathtt{N} <: \mathtt{T}$

Proof *by structural induction on the derivation of* $\Delta; \mathcal{H} \vdash \iota : \mathtt{T}$ *with a case analysis on the last step:*

Case 1 (T-Var)

1.	$\mathtt{T} = \exists\emptyset.\mathtt{N}$	*by def* T-Var, H-T, **b**
2.	*done*	*by* **1**, *reflexivity*

Case 2 (T-Subs)

1.	$\Delta; \mathcal{H} \vdash \iota : \mathtt{U}$	*by premises of* T-Subs
2.	$\Delta \vdash \mathtt{U} <: \mathtt{T}$	
3.	$\mathcal{H}(\iota) = \{\mathtt{N}; ...\}$	*by* **1**, *ind hyp*
4.	$\Delta \vdash \exists\emptyset.\mathtt{N} <: \mathtt{U}$	
5.	$\Delta \vdash \exists\emptyset.\mathtt{N} <: \mathtt{T}$	*by* **2**, **4**, S-Trans

□

Lemma 46 (Generalisation of XS-Env).

If:

a. $\Delta, \Delta' \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T/Z}}]\mathtt{B}_u}$

b. $\Delta, \Delta' \vdash \overline{[\overline{\mathtt{T/Z}}]\mathtt{B}_l <: \mathtt{T}}$

c. $fv(\overline{\mathtt{T}}) \subseteq dom(\Delta, \Delta', \Delta'')$

d. $dom(\Delta') \cap fv(\exists\Delta'', \overline{\mathtt{Z}\to[\mathtt{B}_l\ \mathtt{B}_u]}, \Delta'''.\mathtt{N}) = \emptyset$

e. $\Delta \vdash \exists\Delta'', \Delta'''.\mathtt{N}$ OK

then:

$\Delta \vdash \exists\Delta'', \Delta', [\overline{\mathtt{T/Z}}]\Delta'''.[\overline{\mathtt{T/Z}}]\mathtt{N} \sqsubset: \exists\Delta'', \overline{\mathtt{Z}\to[\mathtt{B}_l\ \mathtt{B}_u]}, \Delta'''.\mathtt{N}$

Proof *by deduction*

1.	*let* $\Delta_0 = \Delta'', \Delta', \Delta'''$	
2.	*let* $\Delta_1 = \Delta'', \overline{\mathtt{Z}\to\mathtt{[B}_l\ \mathtt{B}_u\mathtt{]}}, \Delta'''$	
3.	*let* $\Delta_1 = \overline{\mathtt{Y}\to\mathtt{[B}'_l\ \mathtt{B}'_u\mathtt{]}}$	
4.	*let* $\overline{\mathtt{X}\to\mathtt{[B}''_l\ \mathtt{B}''_u\mathtt{]}} = \Delta'''$	
5.	*let* $\mathtt{X}' \to\mathtt{[B}'''_l\ \mathtt{B}'''_u\mathtt{]} = \Delta''$	
6.	$\overline{\mathtt{X}} \subseteq \overline{\mathtt{Y}}$	*by* **2**, **3**, **4**
7.	$\overline{\mathtt{X}'} \subseteq \overline{\mathtt{Y}}$	*by* **2**, **3**, **5**
8.	$\overline{\mathtt{X}} \subseteq dom(\Delta_0)$	*by* **1**, **4**
9.	$\overline{\mathtt{X}'} \subseteq dom(\Delta_0)$	*by* **1**, **5**
10.	$fv(\overline{\mathtt{T}}), \overline{\mathtt{X}}, \overline{\mathtt{X}'} \subseteq dom(\Delta, \Delta_0)$	*by* **c**, **8**, **9**
11.	$dom(\Delta_0) \cap fv(\exists\Delta_1.\mathtt{N}) = \emptyset$	*by* **d**, **1**, Δ''' *and* Δ'' *bind in* $\exists\Delta_1.\mathtt{N}$
12.	$\Delta, \Delta_0 \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T/Z}}]\mathtt{B}_u}$	*by* **a**, *lemma 8*, **1**
13.	$\Delta, \Delta_0 \vdash \overline{[\overline{\mathtt{T/Z}}]\mathtt{B}_l <: \mathtt{T}}$	*by* **b**, *lemma 8*, **1**
14.	$\overline{\mathtt{Z}} \notin fv(\overline{\mathtt{B}''_l}, \overline{\mathtt{B}''_u})$	*by* **e**,**4**
15.	$\overline{\mathtt{Z}} \notin fv(\overline{\mathtt{B}'''_l}, \overline{\mathtt{B}'''_u})$	*by* **e**,**5**
16.	$\Delta, \Delta_0 \vdash \overline{\mathtt{X} <: [\overline{\mathtt{T/Z}}]\mathtt{B}''_u}$	*by* S-Bound, **14**
17.	$\Delta, \Delta_0 \vdash \overline{[\overline{\mathtt{T/Z}}]\mathtt{B}''_l <: \mathtt{X}}$	*by* S-Bound, **14**
18.	$\Delta, \Delta_0 \vdash \overline{\mathtt{X}' <: [\overline{\mathtt{T/Z}}]\mathtt{B}'''_u}$	*by* S-Bound, **15**
19.	$\Delta, \Delta_0 \vdash \overline{[\overline{\mathtt{T/Z}}]\mathtt{B}'''_l <: \mathtt{X}'}$	*by* S-Bound, **15**
20.	$\Delta \vdash \exists\Delta_0.[\overline{\mathtt{T/Z}}]\mathtt{N} \sqsubset: \exists\Delta_1.\mathtt{N}$	*by* **10**, **11**, **12**, **13**, **16**, **17**, XS-Env
21.	*done*	*by* **20**, **14**, **15**, **1**, **2**

□

Lemma 47 (*Close* gives subtyping under appropriate substitutions).

If:

- **a.** $\Delta \vdash \overline{\mathtt{U} <: [\overline{\mathtt{U/Z}}]\mathtt{B}_u}$
- **b.** $\Delta \vdash \overline{[\overline{\mathtt{U/Z}}]\mathtt{B}_l <: \mathtt{U}}$
- **c.** $fv(\overline{\mathtt{U}}) \subseteq dom(\Delta)$
- **d.** $\Delta, \overline{\mathtt{Z}\to\mathtt{[B}_l\ \mathtt{B}_u\mathtt{]}} \vdash \mathtt{T}$ OK
- **e.** $\Delta \vdash \overline{\mathtt{Z}\to\mathtt{[B}_l\ \mathtt{B}_u\mathtt{]}}$ OK

then:

$\Delta \vdash [\overline{\mathtt{U/Z}}]\mathtt{T} <: \Downarrow_{\overline{\mathtt{Z}\to\mathtt{[B}_l\ \mathtt{B}_u\mathtt{]}}} \mathtt{T}$

Proof *by structural induction on the derivation of* $\Downarrow_{\overline{\mathtt{Z}\to\mathtt{[B}_l\ \mathtt{B}_u\mathtt{]}}} \mathtt{T}$ *with a case analysis on the last step:*

Case 1

1.	$\mathtt{T} = \exists\emptyset.\mathtt{X}$	*by def* close
2.	$\mathtt{X} \notin \overline{\mathtt{Z}}$	
3.	$\Downarrow_{\overline{\mathtt{Z}\to\mathtt{[B}_l\ \mathtt{B}_u\mathtt{]}}} \mathtt{T} = \exists\emptyset.\mathtt{X}$	

4.	$[\overline{U/Z}]T = T$	*by* **2**, **1**
5.	*done*	*by reflexivity,* **4**, **3**, **1**

Case 2

1.	$T = \exists\emptyset.X$	*by def* close
2.	$X = Z_i$	*by def* close
3.	$\Downarrow_{\overline{Z\to[B_l\ B_u]}} T = \Downarrow_{\overline{Z\to[B_l\ B_u]}} B_u i$	*by def* close
4.	$[\overline{U/Z}]T = U_i$	*by* **1**, **2**
5.	$\Delta \vdash [\overline{U/Z}]T <: [\overline{U/Z}]B_u i$	*by* **a**, **4**
6.	$\Delta, \overline{Z\to[B_l\ B_u]} \vdash B_u i$ OK	*by* **e**, *def* F-Env
7.	$\Delta \vdash [\overline{U/Z}]B_u i <: \Downarrow_{\overline{Z\to[B_l\ B_u]}} B_u i$	*by* **a**, **b**, **c**, **6**, **e**, *ind hyp*
8.	*done*	*by* **5**, **7**, *transitivity,* **3**

Case 3

1.	$T = \exists\Delta'.N$	*by def* close
2.	$\Downarrow_{\overline{Z\to[B_l\ B_u]}} T = \exists\overline{Z\to[B_l\ B_u]}, \Delta'.N$	*by def* close
3.	$\Delta \vdash \Downarrow_{\overline{Z\to[B_l\ B_u]}} T$ OK	*by* **d**, **e**, *lemma 30*
4.	$\Delta \vdash \exists[\overline{U/Z}]\Delta'.[\overline{T/Z}]N \sqsubset: \exists\overline{Z\to[B_l\ B_u]}, \Delta''.N$	*by* **a**, **b**, **c**, **1**, **3**, *lemma 46*
5.	$\Delta \vdash [\overline{U/Z}]T \sqsubset: \exists\overline{Z\to[B_l\ B_u]}, \Delta'.N$	*by* **4**, **1**
6.	*done*	*by* **5**, **2**, S-SC

□

Lemma 48 (Reduction preserves heap judgements).

If:

a. $\Delta; \mathcal{H} \vdash e : T$

b. $e'; \mathcal{H} \leadsto e''; \mathcal{H}'$

then:

$\Delta; \mathcal{H}' \vdash e : T$

Proof *by structural induction on the derivation of* $e'; \mathcal{H} \leadsto e''; \mathcal{H}'$ *with a case analysis on the last step:*

Case 1 (R-Field, R-Invk, R-Cast, R-Cast-Null, R-Bad-Cast, *-Null)

1. *trivial*

Case 2 (RC-*)

1. $easy, byindhyp$

Case 3 (R-Assign)

1.	$\mathcal{H}(\iota) = \{N;\ \overline{f\to v}[f_i \mapsto v]\}$	*by premise of* R-Assign
2.	$\mathcal{H}' = \mathcal{H}[\iota \mapsto \{N;\ \overline{f\to v}[f_i \mapsto v]\}]$	*by premise of* R-Assign

3.	$\mathcal{H} = \overline{\iota \rightarrow \texttt{C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\iota'\texttt{>}; ...}$	*by* **a**, *def* H-T
4.	$\Delta, \overline{\iota \rightarrow \texttt{[}\bot\ \mathcal{T}\texttt{]}}; \overline{\iota\texttt{:C<}\overline{\mathcal{T}}\texttt{,}\mathcal{T}\texttt{,}\iota'\texttt{>}} \vdash \texttt{e} : \texttt{T}$	
5.	$\Delta; \mathcal{H}' \vdash \texttt{e} : \texttt{T}$	*by* **2**, **4**, H-T

Case 4 (R-New)

1. $easy, byweakening$

□

.

Theorem (Subject Reduction).

If:

a. $\emptyset; \mathcal{H} \vdash \texttt{e} : \texttt{T}$
b. $\texttt{e}; \mathcal{H} \rightsquigarrow \texttt{e}'; \mathcal{H}'$
c. $\vdash \mathcal{H}$ OK

then:

$\texttt{e}' = \texttt{err}$

or:

$\emptyset; \mathcal{H}' \vdash \texttt{e}' : \texttt{T}$
$\vdash \mathcal{H}'$ OK

Proof *by structural induction on the derivation of* $\texttt{e}; \mathcal{H} \rightsquigarrow \texttt{e}'; \mathcal{H}'$ *with a case analysis on the last step:*

Case 1 (R-New)

1.	$\texttt{e} = \texttt{new C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, }\star\texttt{ >}$	*by def* R-New
2.	$\texttt{e}' = \iota$	
3.	$\iota \notin dom(\mathcal{H})$	*by premises of* R-New
4.	$fields(\texttt{C}) = \overline{\texttt{f}}$	
5.	$\mathcal{H}' = \mathcal{H}, \iota \rightarrow \{\texttt{C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, }\iota\texttt{>}; \overline{\texttt{f}\rightarrow\texttt{null}}\}$	
6.	$\mathcal{H} \vdash \overline{\mathcal{T}}, \mathcal{T}$ OK	*by* **1**, **a**, *lemma 32*
7.	$\mathcal{H} \vdash \exists\texttt{O}\rightarrow\texttt{[}\bot\ \mathcal{T}\texttt{].C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, O>}$ OK	
8.	$\mathcal{H} \vdash \exists\texttt{O}\rightarrow\texttt{[}\bot\ \mathcal{T}\texttt{].C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, O>} <: \texttt{T}$	
9.	$\mathcal{H}' \vdash \exists\texttt{O}\rightarrow\texttt{[}\bot\ \mathcal{T}\texttt{].C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, O>} <: \texttt{T}$	*by* **8**, **5**, *lemma 8*
10.	$\emptyset; \mathcal{H}' \vdash \iota : \texttt{C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, }\iota\texttt{>}$	*by* **5**, H-T, T-Var
11.	$\mathcal{H} \vdash \texttt{T}$ OK	*by* **a**, F-Env-Empty, *lemma 31*
12.	$\mathcal{H}' \vdash \texttt{T}$ OK	*by* **11**, **5**, *lemma 9*
13.	$\emptyset \cap fv(...) = \emptyset$	*by def intersection*
14.	$fv(\iota) \in \emptyset$	*by def fv*
15.	$\mathcal{H}' \vdash \iota <: \mathcal{T}$	*by* **5**, *def* H-S, S-Bound
16.	$\mathcal{H}' \vdash \texttt{C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, }\iota\texttt{>} <: \exists\texttt{O}\rightarrow\texttt{[}\bot\ \mathcal{T}\texttt{].C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, O>}$	*by* **13**, **14**, **15**, XS-Bttm, XS-Env, S-SC
17.	$\mathcal{H}' \vdash \texttt{C<}\overline{\mathcal{T}}\texttt{, }\mathcal{T}\texttt{, }\iota\texttt{>} <: \texttt{T}$	*by* **16**, **9**, S-Trans
18.	$\emptyset; \mathcal{H}' \vdash \iota : \texttt{T}$	*by* **10**, **17**, **12**, T-Subs

19. $\mathcal{H} \vdash$ `O`$\rightarrow$`[`$\bot$ $\mathcal{T}$`]` OK
20. $\mathcal{H}$, `O`$\rightarrow$`[`$\bot$ $\mathcal{T}$`]C<`$\overline{\mathcal{T}}$`,` $\mathcal{T}$`, O>` $\vdash$ OK — *by* **7**, *def* F-Exists

Case analysis on `C`**:**

Case 1 `C` $\neq$ `Object`

1.1. $\mathcal{H}$, `O`$\rightarrow$`[`$\bot$ $\mathcal{T}$`]`$\overline{\mathcal{T}}$, $\mathcal{T}$, `O` $\vdash$ OK
1.2. `class C<`$\overline{\mathcal{X} \triangleleft \mathcal{T}_u}$`>` $\triangleleft$ ...
1.3. $\mathcal{H}$, `O`$\rightarrow$`[`$\bot$ $\mathcal{T}$`]` $\vdash \overline{\mathcal{T}}, \mathcal{T}$, `O` <: $\overline{[\overline{\mathcal{T}}, \mathcal{T}, \texttt{O}/\overline{\texttt{X}}]\mathcal{T}_u}$
1.4. ($\mathcal{H}$, `O`$\rightarrow$`[`$\bot$ $\mathcal{T}$`]`)(`O`) = `[`$\bot$ $\mathcal{T}$`']` — *by* **20**, *def* F-Class

1.5. $\mathcal{H}' \vdash \iota$ OK — *by* **5**, *def* H-F, F-Var
1.6. $\mathcal{H}'(\iota)$ = `[`$\bot$ $\mathcal{T}$`]` — *by* **5**, *def* H-F
1.7. $\mathcal{H}, \iota \rightarrow$`[`$\bot$ $\mathcal{T}$`]` $\vdash \overline{\mathcal{T}}, \mathcal{T}, \iota$ <: $\overline{[\overline{\mathcal{T}}, \mathcal{T}, \iota/\overline{\texttt{X}}]\mathcal{T}_u}$ — *by* **1.3**, **6**, *renaming*
1.8. $\mathcal{H}' \vdash \overline{\mathcal{T}}, \mathcal{T}, \iota$ <: $\overline{[\overline{\mathcal{T}}, \mathcal{T}, \iota/\overline{\texttt{X}}]\mathcal{T}_u}$ — *by* **1.7**, **5**, *def* H-S
1.9. $\mathcal{H}' \vdash$ `C<`$\overline{\mathcal{T}}$`,` $\mathcal{T}$`,` ι`>` OK — *by* **6**, *lemma 8*, **1.5**, **1.6**, **1.2**, **1.8**

Case 2 `C` = `Object`

2.1. $\overline{\mathcal{T}} = \emptyset$ — *by def syntax*
2.2. $\mathcal{H}$, `O`$\rightarrow$`[`$\bot$ $\mathcal{T}$`]`$\mathcal{T}$, `O` $\vdash$ OK
2.3. ($\mathcal{H}$, `O`$\rightarrow$`[`$\bot$ $\mathcal{T}$`]`)(`O`) = `[`$\bot$ $\mathcal{T}$`']` — *by* **20**, *def* F-Object

2.4. $\mathcal{H}' \vdash \iota$ OK — *by* **5**, *def* H-F, F-Var
2.5. $\mathcal{H}'(\iota)$ = `[`$\bot$ $\mathcal{T}$`]` — *by* **5**, *def* H-F
2.6. $\mathcal{H}' \vdash$ `Object<`$\mathcal{T}$`,` ι`>` OK — *by* **6**, *lemma 8*, **2.4**, **2.5**,

21. $\mathcal{H}' \vdash$ `C<`$\overline{\mathcal{T}}$`,` $\mathcal{T}$`,` ι`>` OK — *by by cases*
22. *let* $\overline{fType(\texttt{f}, \texttt{C<}\overline{\mathcal{T}}\texttt{,}\ \mathcal{T}\texttt{,}\ \iota\texttt{>}) = U}$
23. $\mathcal{H} \vdash \iota \rightarrow$ `[`$\bot$ $\mathcal{T}$`]` OK — *by* **7**, *def* F-Exists, F-Env
24. $\mathcal{H}' \vdash \overline{U}$ OK — *by* **22**, **21**, **c**, **23**, *def* H-F, *lemma 27*
25. $\emptyset; \mathcal{H}' \vdash \overline{\texttt{null} : U}$ — *by* **24**, *lemma 10*, T-Null
26. $\vdash \mathcal{H}'$ OK — *by* **5**, **c**, **7**, **22**, **25**, *def* F-Heap
27. *done* — *by* **18**, **2**, **26**

Case 2 (R-Field)

1. `e` = ι`.f`$_i$
2. `e'` = `v`$_i$
3. $\mathcal{H}' = \mathcal{H}$ — *by def* R-Field
4. $\mathcal{H}(\iota)$ = {`N`; $\overline{\texttt{f}\rightarrow\texttt{v}}$} — *by premise of* R-Field

5.	$\emptyset;\mathcal{H} \vdash \iota : \exists\Delta'.\mathtt{N}'$	
6.	$fType(\mathtt{f}, \mathtt{N}') = \mathtt{T}'$	*by* **1**, **a**, F-Env-Empty, *lemma 33*
7.	$\emptyset \vdash \Downarrow_{\Delta'} \mathtt{T}' <: \mathtt{T}$	
8.	$\emptyset \vdash \exists\emptyset.\mathtt{N} <: \exists\Delta'.\mathtt{N}'$	*by* **4**, **5**, *lemma 45*
9.	$\emptyset \vdash \mathtt{T}$ OK	*by* **a**, F-Env-Empty, *lemma 31*
10.	$\emptyset \vdash \exists\emptyset.\mathtt{N} \sqsubset: \exists\Delta'.\mathtt{N}'$	*by* **8**, F-Env-Empty, *lemma 16*
11.	*let* $\Delta' = \overline{\mathtt{Z}\rightarrow[\mathtt{B}_l\ \mathtt{B}_u]}$	
12.	*There exists* $\overline{\mathtt{T}_s}$	
13.	$\vdash \mathtt{N} \sqsubset\!\!\sqsubset: [\overline{\mathtt{T}_s/\mathtt{Z}}]\mathtt{N}'$	
14.	$\emptyset \vdash \overline{\mathtt{T}_s <: [\overline{\mathtt{T}_s/\mathtt{Z}}]\mathtt{B}_u}$	*by* **10**, F-Env-Empty, *lemma 39*
15.	$\emptyset \vdash \overline{[\overline{\mathtt{T}_s/\mathtt{Z}}]\mathtt{B}_l <: \mathtt{T}_s}$	
16.	$fv(\overline{\mathtt{T}_s}) = \emptyset$	
17.	$fType(\mathtt{f}_i, \mathtt{N}) = \mathtt{U}_i$	*by* **4**, **c**, *def* F-Heap
18.	$\emptyset, \mathcal{H} \vdash \mathtt{v}_i : \mathtt{U}_i$	
19.	$\mathtt{U}_i = fType(\mathtt{f}_i, [\overline{\mathtt{T}_s/\mathtt{Z}}]\mathtt{N}')$	*by* **13**, **6**, **17**, *lemma 23*
20.	$\mathtt{U}_i = [\overline{\mathtt{T}_s/\mathtt{Z}}]fType(\mathtt{f}_i, \mathtt{N}')$	*by* **19**, *lemma 5*
21.	$\mathtt{U}_i = [\overline{\mathtt{T}_s/\mathtt{Z}}]\mathtt{T}'$	*by* **20**, **6**
22.	$\emptyset \vdash \exists\Delta'.\mathtt{N}'$ OK	*by* **5**, F-Env-Empty, **c**, *lemma 31*
23.	$\Delta' \vdash \mathtt{T}'$ OK	*by* **22**, **6**, F-Env-Empty, *lemma 27*
24.	$\emptyset \vdash \Delta'$ OK	*by* **22**, *def* F-Exists
25.	$\emptyset \vdash [\overline{\mathtt{T}_s/\mathtt{Z}}]\mathtt{T}' <: \Downarrow_{\Delta'} \mathtt{T}'$	*by* **14**, **15**, **16**, **23**, **11**, **24**, F-Env-Empty, *le*
26.	$\emptyset \vdash \mathtt{U}_i <: \Downarrow_{\Delta'} \mathtt{T}'$	*by* **25**, **21**
27.	$\emptyset \vdash \mathtt{U}_i <: \mathtt{T}$	*by* **26**, **7**, *transitivity*
28.	$\emptyset;\mathcal{H} \vdash \mathtt{v}_i : \mathtt{T}$	*by* **18**, **27**, F-Env-Empty, **9**, T-Subs
29.	$\vdash \mathcal{H}'$ OK	*by* **3**, **c**
30.	*done*	*by* **28**, **2**, **29**

Case 3 (R-Assign)

1.	$\mathtt{e} = \iota.\mathtt{f}_i\ \mathtt{=}\ \mathtt{v}$	*by def* R-Assign
2.	$\mathtt{e}' = \mathtt{v}$	
3.	$\mathcal{H}(\iota) = \{\mathtt{N}; \overline{\mathtt{f}\rightarrow\mathtt{v}}\}$	*by premises of* R-Assign
4.	$\mathcal{H}' = \mathcal{H}[\iota \mapsto \{\mathtt{N}; \overline{\mathtt{f}\rightarrow\mathtt{v}}[\mathtt{f}_i \mapsto \mathtt{v}]\}]$	
5.	$\emptyset;\mathcal{H} \vdash \iota : \exists\Delta'.\mathtt{N}'$	
6.	$fType(\mathtt{f}_i, \mathtt{N}') = \mathtt{U}$	
7.	$\emptyset;\mathcal{H} \vdash \mathtt{v} : \mathtt{U}'$	*by* **1**, **a**, F-Env-Empty, *lemma 34*
8.	$\Delta' \vdash \mathtt{U}' <: \mathtt{U}$	
9.	$\emptyset \vdash \mathtt{U}' <: \mathtt{T}$	
10.	$\emptyset \vdash \mathtt{T}$ OK	*by* **a**, F-Env-Empty, *lemma 31*
11.	$\emptyset;\mathcal{H} \vdash \mathtt{v} : \mathtt{T}$	*by* **7**, **9**, **10**, T-Subs
12.	$\emptyset;\mathcal{H}' \vdash \mathtt{v} : \mathtt{T}$	*by* **11**, **4**, *def* H-T

13.	$\emptyset \vdash \mathtt{N}$ OK	*by* **3**, **c**, *def* F-Heap
14.	$\overline{fType(\mathtt{f},\mathtt{N}) = \mathtt{U}}$	
15.	$\emptyset;\mathcal{H} \vdash \overline{\mathtt{v} : \mathtt{U}}$	
16.	$\emptyset \vdash \exists\emptyset.\mathtt{N} <: \exists\Delta'.\mathtt{N}'$	*by* **3**, **5**, *lemma 45*
17.	$\emptyset \vdash \exists\emptyset.\mathtt{N} \sqsubset: \exists\Delta'.\mathtt{N}'$	*by* **34**, F-Env-Empty, *lemma 16*
18.	*let* $\Delta' = \overline{\mathtt{Z}\rightarrow[\mathtt{B}_l\ \mathtt{B}_u]}$	
19.	*There exists* $\overline{\mathtt{T}_s}$	*by* **17**, F-Env-Empty, *lemma 39*
20.	$\vdash \mathtt{N} \sqsubset\!\sqsubset: [\overline{\mathtt{T}_s/\mathtt{Z}}]\mathtt{N}'$	
21.	$\emptyset \vdash \overline{\mathtt{T}_s <: [\overline{\mathtt{T}_s/\mathtt{Z}}]\mathtt{B}_u}$	
22.	$\emptyset \vdash \overline{[\overline{\mathtt{T}_s/\mathtt{Z}}]\mathtt{B}_l <: \mathtt{T}_s}$	
23.	$fv(\overline{\mathtt{T}_s}) = \emptyset$	
24.	$\mathtt{U}_i = fType(\mathtt{f}_i, [\overline{\mathtt{T}_s/\mathtt{Z}}]\mathtt{N}')$	*by* **20**, **6**, **14**, *lemma 23*
25.	$\mathtt{U}_i = [\overline{\mathtt{T}_s/\mathtt{Z}}]fType(\mathtt{f}_i,\mathtt{N}')$	*by* **24**, *lemma 5*
26.	$\mathtt{U}_i = [\overline{\mathtt{T}_s/\mathtt{Z}}]\mathtt{U}$	*by* **25**, **6**
27.	$\emptyset \vdash \exists\Delta'.\mathtt{N}'$ OK	*by* **5**, F-Env-Empty, *lemma 31*
28.	$\Delta' \vdash \mathtt{N}'$ OK	*by* **27**, *def* F-Exists
29.	$\emptyset \vdash \Delta'$ OK	
30.	$\Delta' \vdash \mathtt{U}$ OK	*by* **6**, **28**, **29***lemma 27*
31.	$\mathtt{U}_i = \mathtt{U}$	*by* **26**, **30**
32.	$\emptyset;\mathcal{H} \vdash \overline{\mathtt{v} : \mathtt{U}_i}$	*by* **15**, **31**
33.	$\vdash \mathcal{H}'$ OK	*by* **4**, **c**, **13**, **15**, **14**, **32**, *def* F-Heap
34.	*done*	*by* **12**, **2**, **33**

Case 4 (R-Invk)

1.	$\mathtt{e} = \iota.\mathtt{<}\overline{\mathtt{P}}\mathtt{>}(\overline{\iota})$	*by def* R-Invk
2.	$\mathtt{e}' = [\overline{\mathtt{T}/\mathtt{Y}},\ \overline{\iota/\mathtt{x}},\ \iota/\mathtt{this}]\mathtt{e}_0$	
3.	$\mathcal{H}' = \mathcal{H}$	
4.	$\overline{\mathcal{H}(\iota) = \{\mathtt{N}'\}}$	*by premises* R-Invk
5.	$\overline{\mathcal{H}(\iota) = \{\mathtt{N}'\}}$	
6.	$mBody(\mathtt{m},\mathtt{C<}\overline{\mathtt{T}'}\mathtt{>}) = (\overline{\mathtt{x}};\mathtt{e}_0)$	
7.	$mType(\mathtt{m},\mathtt{C<}\overline{\mathtt{T}'}\mathtt{>}) = \mathtt{<}\overline{\mathtt{Y}\lhd\ \mathtt{B}}\mathtt{>}\overline{\mathtt{U}} \rightarrow \mathtt{U}$	
8.	$match(sift(\overline{\mathtt{N}},\overline{\mathtt{U}},\overline{\mathtt{Y}}),\overline{\mathtt{P}},\overline{\mathtt{Y}},\overline{\mathtt{T}})$	
9.	$\emptyset;\mathcal{H} \vdash \iota : \exists\Delta'.\mathtt{N}$	*by* **1**, **a**, *lemma 35*
10.	$mType(\mathtt{m},\ \mathtt{N}) = \mathtt{<}\overline{\mathtt{Y}'\lhd\ \mathtt{B}'}\mathtt{>}\overline{\mathtt{U}''} \rightarrow \mathtt{U}''$	
11.	$\emptyset;\mathcal{H} \vdash \overline{\iota : \exists\Delta.\mathtt{R}}$	
12.	$match(sift(\overline{\mathtt{R}},\overline{\mathtt{U}''},\overline{\mathtt{Y}'}),\overline{\mathtt{P}},\overline{\mathtt{Y}'},\overline{\mathtt{T}''})$	
13.	$\emptyset \vdash \overline{\mathtt{P}}$ OK	
14.	$\Delta',\overline{\Delta} \vdash \overline{\mathtt{T}'' <: [\overline{\mathtt{T}''/\mathtt{Y}'}]\mathtt{B}'}$	
15.	$\Delta',\overline{\Delta} \vdash \overline{\exists\emptyset.\mathtt{R} <: [\overline{\mathtt{T}''/\mathtt{Y}'}]\mathtt{U}''}$	
16.	$\emptyset \vdash \Downarrow_{\Delta',\overline{\Delta}} [\overline{\mathtt{T}''/\mathtt{Y}'}]\mathtt{U}'' <: \mathtt{T}$	

17.	$\emptyset \vdash \exists\Delta'.\mathtt{N}$ OK	*by* **9**, F-Env-Empty, *lemma 31*
18.	$\emptyset \vdash \exists\emptyset.\mathtt{N}' <: \exists\Delta'.\mathtt{N}$	*by* **4**, **9**, *lemma 45*
19.	$\emptyset \vdash \mathtt{N}'$ OK	*by* **4**, *def* F-Heap
20.	$\emptyset \vdash \overline{\exists\emptyset.\mathtt{N}' <: \exists\Delta.\mathtt{R}}$	*by* **5**, **11**, *lemma 45*
21.	$\emptyset \vdash \overline{\mathtt{N}}'$ OK	*by* **5**, *def* F-Heap
22.	$\emptyset \vdash \exists\emptyset.\mathtt{N}' \sqsubset: \exists\Delta'.\mathtt{N}$	*by* **18**, F-Env-Empty, *lemma 16*
23.	$\emptyset \vdash \mathtt{T}$ OK	*by* **a**, F-Env-Empty, *lemma 31*
24.	$\emptyset \vdash \overline{\exists\Delta.\mathtt{R}}$ OK	*by* **11**, F-Env-Empty, *lemma 31*
25.	$\overline{\Delta} \vdash \overline{\mathtt{R}}$ OK	*by* **24**, *def* F-Exists
26.	$\overline{\Delta} \vdash \overline{\mathtt{T}''}$ OK	*by* **13**, **12**, **24**, F-Env-Empty, *lemma 29*
27.	$match(sift(\overline{\mathtt{R}}, \overline{\mathtt{U}}, \overline{\mathtt{Y}'}), \overline{\mathtt{P}}, \overline{\mathtt{Y}'}, \overline{\mathtt{T}''})$	*by* **22**, **10**, **7**, **12**, F-Env-Empty, **26**, *lemma 40*
28.	*there exists* $\overline{\mathtt{N}_{fresh}}$: $\overline{\mathtt{R}} = \overline{\mathtt{N}_{fresh}}$	*by* **24**, *def* F-Var
29.	$\emptyset \vdash \overline{\exists\emptyset.\mathtt{N}' \sqsubset: \exists\Delta.\mathtt{R}}$	*by* **20**, **28**, F-Env-Empty, *lemma 16*
30.	*let* $\Delta' = \overline{\mathtt{X}_x \to [\mathtt{B}_{xl}\ \ \mathtt{B}_{xu}]}$	
31.	*There exists* $\overline{\mathtt{U}_x}$	*by* **22**, **30**, F-Env-Empty, *lemma 39* (for lines 31–35)
32.	$\vdash \mathtt{N}' \sqsubset: [\overline{\mathtt{U}_x/\mathtt{X}_x}]\mathtt{N}$	
33.	$\emptyset \vdash \overline{\mathtt{U}_x <: [\overline{\mathtt{U}_x/\mathtt{X}_x}]\mathtt{B}_{xu}}$	
34.	$\emptyset \vdash \overline{[\overline{\mathtt{U}_x/\mathtt{X}_x}]\mathtt{B}_{xl} <: \mathtt{U}_x}$	
35.	$fv(\overline{\mathtt{U}_x}) = \emptyset$	
36.	$mType(\mathtt{m},\ [\overline{\mathtt{U}_x/\mathtt{X}_x}]\mathtt{N}) = [\overline{\mathtt{U}_x/\mathtt{X}_x}]\mathtt{<}\overline{\mathtt{Y}' \lhd\ \mathtt{B}'}\mathtt{>}\overline{\mathtt{U}''} \to\ \mathtt{U}''$	*by* **10**, *lemma 6*
37.	$mType(\mathtt{m},\ \mathtt{N}') = [\overline{\mathtt{U}_x/\mathtt{X}_x}]\mathtt{<}\overline{\mathtt{Y}' \lhd\ \mathtt{B}'}\mathtt{>}\overline{\mathtt{U}''} \to\ \mathtt{U}''$	*by* **32**, **36**, *lemma 24*
38.	$\mathtt{<}\overline{\mathtt{Y} \lhd\ \mathtt{B}}\mathtt{>}\overline{\mathtt{U}}\ \to\ \mathtt{U} = [\overline{\mathtt{U}_x/\mathtt{X}_x}]\mathtt{<}\overline{\mathtt{Y}' \lhd\ \mathtt{B}'}\mathtt{>}\overline{\mathtt{U}''} \to\ \mathtt{U}''$	*by* **37**, **7**
39.	$\overline{\mathtt{Y}} = \overline{\mathtt{Y}'}$	*by* **38** (for lines 39–42)
40.	$\overline{\mathtt{U}} = [\overline{\mathtt{U}_x/\mathtt{X}_x}]\overline{\mathtt{U}''}$	
41.	$\mathtt{U} = [\overline{\mathtt{U}_x/\mathtt{X}_x}]\mathtt{U}''$	
42.	$\overline{\mathtt{B}} = [\overline{\mathtt{U}_x/\mathtt{X}_x}]\overline{\mathtt{B}'}$	
43.	$match(sift(\overline{\mathtt{R}}, \overline{\mathtt{U}}, \overline{\mathtt{Y}}), \overline{\mathtt{P}}, \overline{\mathtt{Y}}, \overline{\mathtt{T}''})$	*by* **27**, **39**
44.	$\overline{\mathtt{Y} \to [\bot\ \ \mathtt{B}]} \vdash \overline{\mathtt{U}}$ OK	*by* **7**, F-Env-Empty, **19**, *lemma 28*
45.	*let* $\overline{\Delta} = \overline{\mathtt{X}_s \to [\mathtt{B}_{sl}\ \ \mathtt{B}_{su}]}$	
46.	$\emptyset \vdash \overline{\exists\emptyset.\mathtt{N}'}$ OK	*by* **21**, F-Env-Empty, F-Exists
47.	*There exists* $\overline{\mathtt{U}_s}$	*by* **29**, **43**, **8**, **44**, **45**, F-Env-Empty, **46**, **13**, *lemma 41* (for lines 47–52)
48.	$\overline{\mathtt{T} = [\overline{\mathtt{U}_s/\mathtt{X}_s}]\mathtt{T}''}$	
49.	$\emptyset \vdash \overline{\mathtt{U}_s <: [\overline{\mathtt{U}_s/\mathtt{X}_s}]\mathtt{B}_{su}}$	
50.	$\emptyset \vdash \overline{[\overline{\mathtt{U}_s/\mathtt{X}_s}]\mathtt{B}_{sl} <: \mathtt{U}_s}$	
51.	$\vdash \overline{\mathtt{N}'} \sqsubset: \overline{[\overline{\mathtt{U}_s/\mathtt{X}_s}]\mathtt{R}}$	
52.	$fv(\overline{\mathtt{U}_s}) = \emptyset$	
53.	$\Delta', \overline{\mathtt{Y}' \to [\bot\ \ \mathtt{B}']} \vdash \mathtt{U}''$ OK	*by* **10**, **17**, F-Env-Empty, *lemma 28* (for lines 53–55)
54.	$\Delta', \overline{\mathtt{Y}' \to [\bot\ \ \mathtt{B}']} \vdash \overline{\mathtt{U}''}$ OK	
55.	$\Delta', \overline{\mathtt{Y}' \to [\bot\ \ \mathtt{B}']} \vdash \overline{\mathtt{B}'}$ OK	

56. $\Delta' \vdash \overline{[\overline{U_s/X_s}]T'' <: [\overline{U_s/X_s}]\,[\overline{T''/Y'}]B}'$ *by* **14**, **49**, **50**, **52**, *lemma 17*

57. $\Delta' \vdash \overline{[\overline{U_s/X_s}]T'' <: [\overline{[\overline{U_s/X_s}]T''/Y'}]B}'$ *by* **56**, **55**

58. $\Delta' \vdash \overline{T <: [\overline{T/Y}]B}'$ *by* **57**, **48**, **39**

59. $\emptyset \vdash \overline{[\overline{U_x/X_x}]T <: [\overline{U_x/X_x}]\,[\overline{T/Y}]B}'$ *by* **58**, **33**, **34**, **35**, *lemma 17*

60. $\emptyset \vdash \overline{T <: [\overline{T/Y}]\,[\overline{U_x/X_x}]B}'$ *by* **59**, **23**, **7**, **30**, *lemma 14*

61. $\emptyset \vdash \overline{T <: [\overline{T/Y}]B}$ *by* **60**, **42**

62. $\Delta' \vdash \overline{\exists\emptyset.[\overline{U_s/X_s}]R <: [\overline{U_s/X_s}]\,[\overline{T''/Y'}]U}''$ *by* **15**, **49**, **50**, **52**, *lemma 17*

63. $\Delta' \vdash \overline{\exists\emptyset.[\overline{U_s/X_s}]R <: [\overline{[\overline{U_s/X_s}]T''/Y'}]U}''$ *by* **57**, **54**

64. $\Delta' \vdash \overline{\exists\emptyset.[\overline{U_s/X_s}]R <: [\overline{T/Y}]U}''$ *by* **63**, **48**, **39**

65. $\emptyset \vdash \overline{\exists\emptyset.[\overline{U_x/X_x}]\,[\overline{U_s/X_s}]R <: [\overline{U_x/X_x}]\,[\overline{T/Y}]U}''$ *by* **64**, **33**, **34**, **35**, *lemma 17*

66. $\emptyset \vdash \overline{\exists\emptyset.[\overline{U_x/X_x}]\,[\overline{U_s/X_s}]R <: [\overline{T/Y}]\,[\overline{U_x/X_x}]U}''$ *by* **65**, **23**, **7**, **30**, *lemma 14*

67. $\emptyset \vdash \overline{\exists\emptyset.[\overline{U_x/X_x}]\,[\overline{U_s/X_s}]R <: [\overline{T/Y}]U}$ *by* **66**, **40**

68. $\emptyset \vdash \exists\emptyset.N' \sqsubset: \overline{\exists\emptyset.[\overline{U_s/X_s}]R}$ *by* **51**, *lemma 38*

69. $\emptyset \vdash \exists\emptyset.[\overline{U_x/X_x}]N' \sqsubset: \overline{\exists\emptyset.[\overline{U_x/X_x}]\,[\overline{U_s/X_s}]R}$ *by* **68**, **33**, **34**, **35**, *lemma 17*

70. $\emptyset \vdash \overline{\exists\emptyset.[\overline{U_x/X_x}]N' <: [\overline{T/Y}]U}$ *by* **67**, **69**, XS-Trans, S-SC

71. $\emptyset \vdash \overline{\exists\emptyset.N' <: [\overline{T/Y}]U}$ *by* **70**, **21**

72. *let* $U_c = \overline{[\overline{T''/Y'}]U}''$

73. *let* $U'_c = [\overline{U_s/X_s}](\overline{[\overline{[\overline{U_s/X_s}]U_x/X_x}]U_c})$

74. $\emptyset \vdash \Delta'$ OK *by* **17**, F-Exists

75. $\emptyset \vdash \overline{\Delta}$ OK *by* **24**, F-Exists

76. $fv(\overline{B_{xl}}, fv(\overline{B_{xu}}, fv(\overline{B_{sl}}, fv(\overline{B_{su}}) = \emptyset$ *by* **74**, **75**

77. $\emptyset \vdash \overline{U_x) <: [\overline{U_s/X_s}](\overline{[\overline{[\overline{U_s/X_s}]U_x/X_x}]B_{xu}}}$ *by* **33**, **35**, **76**

78. $\emptyset \vdash \overline{[\overline{U_s/X_s}](\overline{[\overline{[\overline{U_s/X_s}]U_x/X_x}]B_{xl}} <: U_x}$ *by* **34**, **35**, **76**

79. $\emptyset \vdash \overline{U_s <: [\overline{U_s/X_s}](\overline{[\overline{[\overline{U_s/X_s}]U_x/X_x}]B_{su}}}$ *by* **49**, **52**, **76**

80. $\emptyset \vdash \overline{[\overline{U_s/X_s}](\overline{[\overline{[\overline{U_s/X_s}]U_x/X_x}]B_{sl}} <: U_s}$ *by* **50**, **52**, **76**

81. $fv(\overline{[\overline{U_s/X_s}]U_x}, \overline{U_s}) \subseteq dom(\Delta)$ *by* **52**, **35**

82. $\emptyset \vdash \Delta', \overline{\Delta}$ OK *by* **74**, **75**, **16** *gives* Δ' *and* $\overline{\Delta}$ *are disjoint*

83. $\Delta', \overline{\Delta}, \overline{Y' \to [\bot\ B']} \vdash U''$ OK *by* **53**, *lemma 9*

84. $\Delta', \overline{\Delta} \vdash \overline{T''}$ OK *by* **26**, *lemma 9*

85. $\Delta', \overline{\Delta} \vdash U_c$ OK *by* **72**, **83**, **84**, **14**, SC-Bottom, **82**, *lemma 1*

86. $\emptyset \vdash U'_c <:\Downarrow_{\Delta', \overline{\Delta}} U_c$ *by* **73**, **77**, **78**, **79**, **80**, **81**, **85**, **82**, *lemma 47*

87. $\emptyset \vdash [\overline{U_x/X_x}, \overline{U_s/X_s}]U_c <:\Downarrow_{\Delta', \overline{\Delta}} U_c$ *by* **86**, *def subst*, **73**

88. $\emptyset \vdash [\overline{U_s/X_s}]\,[\overline{U_x/X_x}]\,[\overline{T''/Y'}]U'' <: T$ *by* **87**, **72**, **73**, **16**, *transitivity*

89. $\emptyset \vdash [\overline{U_s/X_s}]\,[\overline{T''/Y'}]U <: T$ *by* **88**, **41**, **26**

90. $\emptyset \vdash [\overline{T/Y}]\,[\overline{U_s/X_s}]U <: T$ *by* **89**, **48**, **39**

91. $fv(U'') \subseteq \overline{Y'}, \overline{X_x}$ *by* **53**

92. $fv(U'') \subseteq \overline{Y'}$ *by* **91**, **41**, **44**, **39**

93.	$\emptyset \vdash [\overline{\texttt{T/Y}}]\texttt{U} <: \texttt{T}$	*by* **90**, **92**
94.	$\overline{\texttt{Y}\rightarrow[\bot\ \texttt{B}]};\overline{\texttt{x:U}},\texttt{this:C<}\overline{\texttt{T}'}\texttt{>} \vdash \texttt{e}_0 : \texttt{U}$	*by* F-Env-Empty, **19**, **6**, **7**, *lemma 42*
95.	$\emptyset \vdash \overline{\texttt{T}}$ OK	*by* **13**, **46**, F-Exist, F-Env-Empty, **8**, *lemma 29*
96.	$\emptyset;\overline{\texttt{x}:[\overline{\texttt{T/Y}}]\texttt{U}},\texttt{this}:[\overline{\texttt{T/Y}}]\texttt{C<}\overline{\texttt{T}'}\texttt{>} \vdash [\overline{\texttt{T/Y}}]\texttt{e}_0 : [\overline{\texttt{T/Y}}]\texttt{U}$	*by* **94**, **61**, XS-Bttm, F-Env-Empty **95**, *lemma 21*
97.	$\emptyset;\overline{\texttt{x}:[\overline{\texttt{T/Y}}]\texttt{U}},\texttt{this}:\texttt{N}' \vdash [\overline{\texttt{T/Y}}]\texttt{e}_0 : [\overline{\texttt{T/Y}}]\texttt{U}$	*by* **96**, **19**
98.	$\emptyset;\mathcal{H} \vdash \iota : \exists\emptyset.\texttt{N}'$	*by* **4**, H-T, T-Var
99.	$\emptyset;\mathcal{H} \vdash \overline{\iota : \exists\emptyset.\texttt{N}'}$	*by* **5**, H-T, T-Var
100.	$\emptyset \vdash \overline{[\overline{\texttt{T/Y}}]\texttt{U}}$ OK	*by* **44**, **95**, F-Env-Empty, XS-Bttm, **61**, *lemma 18*
101.	$\emptyset;\mathcal{H} \vdash [\overline{\texttt{T/Y}},\ \overline{\iota\texttt{/x}},\ \iota\texttt{/this}]\texttt{e}_0 : [\overline{\texttt{T/Y}}]\texttt{U}$	*by* **97**, **98**, **99**, **19**, **100**, **71**, *lemma 25*
102.	$\emptyset;\mathcal{H} \vdash [\overline{\texttt{T/Y}},\ \overline{\iota\texttt{/x}},\ \iota\texttt{/this}]\texttt{e}_0 : \texttt{T}$	*by* **101**, **93**, F-Env-Empty, T-Subs
103.	$\vdash \mathcal{H}'$ OK	*by* **3**, **c**
104.	*done*	*by* **102**, **2**, **103**

Case 5 (R-Cast)

1.	$\texttt{e} = (T\texttt{'})\iota$	*by def* R-Cast (1–3)
2.	$\texttt{e}' = \iota$	
3.	$\mathcal{H}' = \mathcal{H}$	
4.	$\mathcal{H}(\iota) = \{N; ...\}$	*by premise of* R-Cast (4–5)
5.	$\emptyset \vdash N <: T'$	
6.	$\emptyset;\mathcal{H} \vdash \iota : U$	*by* **1**, **a**, *lemma 37* (6–9)
7.	$\emptyset \vdash T' <: U$	
8.	$\emptyset \vdash T'$ OK	
9.	$\emptyset \vdash T' <: T$	
10.	$\emptyset;\mathcal{H} \vdash \iota : N$	*by* **5**, H-T, T-Var
11.	$\emptyset \vdash \texttt{T}$ OK	*by* **a**, **c**, *lemma 31*
12.	$\emptyset \vdash N <: T$	*by* **5**, **9**, S-Trans
13.	$\emptyset;\mathcal{H} \vdash \iota : T$	*by* **10**, **12**, **11**, T-Subs
14.	$\emptyset;\mathcal{H} \vdash e' : T$	*by* **13**, **2**
15.	$\vdash \mathcal{H}'$ OK	*by* **3**, **c**
16.	*done*	*by* **14**, **15**

Case 6 (R-Cast-Null)

1.	$\texttt{e} = (T\texttt{'})\texttt{null}$	*by def* R-Cast (1–3)
2.	$\texttt{e}' = \texttt{null}$	
3.	$\mathcal{H}' = \mathcal{H}$	

4.	$\emptyset \vdash \mathtt{T}\ \text{OK}$	*by* **a**, **c**, *lemma 31*
5.	$\emptyset; \mathcal{H} \vdash \mathtt{null} : T$	*by* **4**, T-Null
6.	$\emptyset; \mathcal{H} \vdash e' : T$	*by* **5**, **2**
7.	$\vdash \mathcal{H}'\ \text{OK}$	*by* **3**, **c**
8.	*done*	*by* **6**, **7**

Case 7 (RC-Field)

1.	$\mathtt{e} = \mathtt{e}_r.\mathtt{f}$	*by def* RC-Field (1–2)
2.	$\mathtt{e}' = \mathtt{e}'_r.\mathtt{f}$	
3.	$\mathtt{e}_r; \mathcal{H} \leadsto \mathtt{e}'_r; \mathcal{H}'$	*by premise* RC-Field (3–4)
4.	$\mathtt{e}'_r \neq \mathtt{err}$	
5.	$\emptyset; \mathcal{H} \vdash \mathtt{e}_r : \exists\Delta_n.\mathtt{N}$	*by* **1**, **a**, F-Env-Empty, *lemma 33* (5–6)
6.	$\emptyset \vdash \Downarrow_{\Delta_n} fType(\mathtt{f}, \mathtt{N}) <: \mathtt{T}$	
7.	$\emptyset; \mathcal{H} \vdash \mathtt{e}'_r : \exists\Delta_n.\mathtt{N}$	*by* **3**, **5**, *ind hyp* (7–8)
8.	$\vdash \mathcal{H}'\ \text{OK}$	
9.	$\emptyset; \mathcal{H} \vdash \mathtt{e}'_r.\mathtt{f} : \Downarrow_{\Delta_n} fType(\mathtt{f}, \mathtt{N})$	*by* **7**, T-Field
10.	$\emptyset \vdash \mathtt{T}\ \text{OK}$	*by* **a**, **c**, *lemma 31*
11.	$\emptyset; \mathcal{H} \vdash \mathtt{e}'_r.\mathtt{f} : \mathtt{T}$	*by* **9**, **6**, **10**, T-Subs
12.	*done*	*by* **11**, *lemma 48*, **2**, **8**

Case 8 (RC-Assign-1)

1.	$\mathtt{e} = \mathtt{e}_1.\mathtt{f} \texttt{ = } \mathtt{e}_2$	*by def* RC-Assign-1 (1–2)
2.	$\mathtt{e}' = \mathtt{e}'_1.\mathtt{f} \texttt{ = } \mathtt{e}_2$	
3.	$\mathtt{e}_1; \mathcal{H} \leadsto \mathtt{e}'_1; \mathcal{H}'$	*by premise* RC-Assign-1
4.	$\emptyset; \mathcal{H} \vdash \mathtt{e}_1 : \exists\Delta'.\mathtt{N}$	*by* **1**, **a**, F-Env-Empty, *lemma 34* (4–8)
5.	$fType(\mathtt{f}, \mathtt{N}) = \mathtt{U}$	
6.	$\emptyset; \mathcal{H} \vdash \mathtt{e}_2 : \mathtt{U}'$	
7.	$\Delta' \vdash \mathtt{U}' <: \mathtt{U}$	
8.	$\emptyset \vdash \mathtt{U}' <: \mathtt{T}$	
9.	$\emptyset; \mathcal{H} \vdash \mathtt{e}'_1 : \exists\Delta'.\mathtt{N}$	*by* **3**, **4**, *ind hyp* (9–10)
10.	$\vdash \mathcal{H}'\ \text{OK}$	
11.	$\emptyset; \mathcal{H} \vdash \mathtt{e}'_1.\mathtt{f} \texttt{ = } \mathtt{e}_2 : \mathtt{U}'$	*by* **9**, **5**, **6**, **7**, T-Assign
12.	$\emptyset \vdash \mathtt{T}\ \text{OK}$	*by* **a**, **c**, *lemma 31*
13.	$\emptyset; \mathcal{H} \vdash \mathtt{e}'_1.\mathtt{f} \texttt{ = } \mathtt{e}_2 : \mathtt{T}$	*by* **11**, **8**, **12**, T-Subs
14.	*done*	*by* **13**, *lemma 48*, **2**, **10**

Case 9 (RC-Assign-2)

1.	$\mathtt{e} = \iota\mathtt{.f = e}_2$	*by def* RC-Assign-2 (1–2)
2.	$\mathtt{e}' = \iota\mathtt{.f = e}'_2$	
3.	$\mathtt{e}_2; \mathcal{H} \rightsquigarrow \mathtt{e}'_2; \mathcal{H}'$	*by premise* RC-Assign-2
4.	$\emptyset; \mathcal{H} \vdash \iota : \exists\Delta'.\mathtt{N}$	*by* **1**, **a**, F-Env-Empty, *lemma 34* (4–8)
5.	$fType(\mathtt{f}, \mathtt{N}) = \mathtt{U}$	
6.	$\emptyset; \mathcal{H} \vdash \mathtt{e}_2 : \mathtt{U}'$	
7.	$\Delta' \vdash \mathtt{U}' <: \mathtt{U}$	
8.	$\emptyset \vdash \mathtt{U}' <: \mathtt{T}$	
9.	$\emptyset; \mathcal{H} \vdash \mathtt{e}'_2 : \exists\Delta'.\mathtt{N}$	*by* **3**, **4**, *ind hyp* (9–10)
10.	$\vdash \mathcal{H}'$ OK	
11.	$\emptyset; \mathcal{H} \vdash \iota\mathtt{.f = e}'_2 : \mathtt{U}'$	*by* **4**, **5**, **9**, **7**, T-Assign
12.	$\emptyset \vdash \mathtt{T}$ OK	*by* **a**, **c**, *lemma 31*
13.	$\emptyset; \mathcal{H} \vdash \iota\mathtt{.f = e}'_2 : \mathtt{T}$	*by* **11**, **8**, **12**, T-Subs
14.	*done*	*by* **13**, *lemma 48*, **2**, **10**

Case 10 (RC-Invk-Recv)

1.	$\mathtt{e} = \mathtt{e}_r\mathtt{.<}\overline{\mathtt{P}}\mathtt{>m(}\overline{\mathtt{e}}\mathtt{)}$	*by def* RC-Invk-Recv (1–2)
2.	$\mathtt{e}' = \mathtt{e}'_r\mathtt{.<}\overline{\mathtt{P}}\mathtt{>m(}\overline{\mathtt{e}}\mathtt{)}$	
3.	$\mathtt{e}_r; \mathcal{H} \rightsquigarrow \mathtt{e}'_r; \mathcal{H}'$	*by premise* RC-Invk-Recv
4.	$\emptyset; \mathcal{H} \vdash \mathtt{e}_r : \exists\Delta''.\mathtt{N}$	*by* **1**, **a**, F-Env-Empty, *lemma 35* (4–11)
5.	$mType(\mathtt{m}, \mathtt{N}) = \mathtt{<}\overline{\mathtt{Y} \lhd \mathtt{B}}\mathtt{>}\overline{\mathtt{U}} \rightarrow \mathtt{U}$	
6.	$\emptyset; \mathcal{H} \vdash \overline{\mathtt{e} : \exists\Delta.\mathtt{R}}$	
7.	$match(sift(\overline{\mathtt{R}}, \overline{\mathtt{U}}, \overline{\mathtt{Y}}), \overline{\mathtt{P}}, \overline{\mathtt{Y}}, \overline{\mathtt{T}})$	
8.	$\emptyset \vdash \overline{\mathtt{P}}$ OK	
9.	$\Delta'', \overline{\Delta} \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T/Y}}]\mathtt{B}}$	
10.	$\Delta'', \overline{\Delta} \vdash \overline{\exists\emptyset.\mathtt{R} <: [\overline{\mathtt{T/Y}}]\mathtt{U}}$	
11.	$\emptyset \vdash \Downarrow_{\Delta'',\overline{\Delta}} [\overline{\mathtt{T/Y}}]\mathtt{U} <: \mathtt{T}$	
12.	$\emptyset; \mathcal{H} \vdash \mathtt{e}'_r : \exists\Delta''.\mathtt{N}$	*by* **3**, **4**, *ind hyp* (12–13)
13.	$\vdash \mathcal{H}'$ OK	
14.	$\emptyset; \mathcal{H} \vdash \mathtt{e}'_r\mathtt{.<}\overline{\mathtt{P}}\mathtt{>m(}\overline{\mathtt{e}}\mathtt{)} : \Downarrow_{\Delta'',\overline{\Delta}} [\overline{\mathtt{T/Y}}]\mathtt{U}$	*by* **12**, **5**, **6**, **7**, **8**, **9**, **10**, T-Invk
15.	$\emptyset \vdash \mathtt{T}$ OK	*by* **a**, F-Env-Empty, *lemma 31*
16.	$\emptyset; \mathcal{H} \vdash \mathtt{e}'_r\mathtt{.<}\overline{\mathtt{P}}\mathtt{>m(}\overline{\mathtt{e}}\mathtt{)} : \mathtt{T}$	*by* **14**, **11**,**15**, T-Subs
17.	*done*	*by* **16**, *lemma 48*, **2**, **13**

Case 11 (RC-Invk-Arg)

1.	$\mathtt{e} = \mathtt{e}_r\mathtt{.<}\overline{\mathtt{P}}\mathtt{>m(}\overline{\mathtt{e}}\mathtt{)}$	*by def* RC-Invk-Arg (1–4)
2.	$\mathtt{e}' = \mathtt{e}_r\mathtt{.<}\overline{\mathtt{P}}\mathtt{>m(}\overline{\mathtt{e}'}\mathtt{)}$	
3.	$\overline{\mathtt{e}} = \ldots \mathtt{e}_i \ldots$	
4.	$\overline{\mathtt{e}'} = \ldots \mathtt{e}'_i \ldots$	

5.	$\mathtt{e}_i; \mathcal{H} \rightsquigarrow \mathtt{e}'_i; \mathcal{H}'$	*by premise* RC-Invk-Arg
6.	$\emptyset; \mathcal{H} \vdash \mathtt{e}_r : \exists\Delta''.\mathtt{N}$	*by* **1**, **a**, F-Env-Empty, *lemma 35*
7.	$mType(\mathtt{m},\ \mathtt{N}) = \mathtt{<}\overline{\mathtt{Y}\lhd\ \mathtt{B}}\mathtt{>}\overline{\mathtt{U}} \rightarrow \mathtt{U}$	
8.	$\emptyset; \mathcal{H} \vdash \overline{\mathtt{e} : \exists\Delta.\mathtt{R}}$	
9.	$match(sift(\overline{\mathtt{R}}, \overline{\mathtt{U}}, \overline{\mathtt{Y}}), \overline{\mathtt{P}}, \overline{\mathtt{Y}}, \overline{\mathtt{T}})$	
10.	$\emptyset \vdash \overline{\mathtt{P}}$ OK	
11.	$\Delta, \Delta'', \overline{\Delta} \vdash \overline{\mathtt{T} <: [\overline{\mathtt{T/Y}}]\mathtt{B}}$	
12.	$\Delta, \Delta'', \overline{\Delta} \vdash \overline{\exists\emptyset.\mathtt{R} <: [\overline{\mathtt{T/Y}}]\mathtt{U}}$	
13.	$\emptyset \vdash \Downarrow_{\Delta'',\overline{\Delta}} [\overline{\mathtt{T/Y}}]\mathtt{U} <: \mathtt{T}$	
14.	$\emptyset; \mathcal{H} \vdash \mathtt{e}'_i : \exists\Delta_i.\mathtt{R}_i$	*by* **5**, **8**, *ind hyp*
15.	$\vdash \mathcal{H}'$ OK	
16.	$\emptyset; \mathcal{H} \vdash \mathtt{e}_r.\mathtt{<}\overline{\mathtt{P}}\mathtt{>m(}\overline{\mathtt{e}'}\mathtt{)} : \Downarrow_{\Delta'',\overline{\Delta}} [\overline{\mathtt{T/Y}}]\mathtt{U}$	*by* **6**, **7**, **8**, **14**, **9**, **10**, **11**, **12**, T-Invk
17.	$\emptyset \vdash \mathtt{T}$ OK	*by* **a**, F-Env-Empty, *lemma 31*
18.	$\emptyset; \mathcal{H} \vdash \mathtt{e}_r.\mathtt{<}\overline{\mathtt{P}}\mathtt{>m(}\overline{\mathtt{e}'}\mathtt{)} : \mathtt{T}$	*by* **16**, **13**, **17**, T-Subs
19.	*done*	*by* **18**, *lemma 48*, **2**, **15**

Case 12 (RC-Cast)

1.	$\mathtt{e} = (T')e_0$	*by def* RC-Cast
2.	$\mathtt{e}' = (T')e'_0$	
3.	$\mathtt{e}_0; \mathcal{H} \rightsquigarrow \mathtt{e}'_0; \mathcal{H}'$	*by premise* RC-Cast
4.	$\mathcal{H}; \emptyset \vdash e_0 : U$	*by* **1**, **a**, F-Env-Empty, *lemma 37*
5.	$\mathcal{H} \vdash T' <: U$	
6.	$\mathcal{H} \vdash T'$ OK	
7.	$\mathcal{H} \vdash T' <: T$	
8.	$\emptyset; \mathcal{H}' \vdash \mathtt{e}'_0 : U$	*by* **3**, **4**, *ind hyp*
9.	$\vdash \mathcal{H}'$ OK	
10.	$\emptyset; \mathcal{H} \vdash (T')e'_0 : T'$	*by* **8**, **5**, **6**, T-Cast
11.	$\emptyset \vdash \mathtt{T}$ OK	*by* **a**, F-Env-Empty, *lemma 31*
12.	$\emptyset; \mathcal{H} \vdash (T')e'_0 : \mathtt{T}$	*by* **10**, **7**,**11**, T-Subs
13.	*done*	*by* **12**, *lemma 48*, **2**, **9**

Case 13 (*-Null, *-Err, R-Bad-Cast)

1. *trivial*

□

Theorem (Progress).

If:

a. $\emptyset; \mathcal{H} \vdash \texttt{e} : \texttt{T}$

b. $\vdash \mathcal{H}$ OK

then:

there exists $\texttt{e}'$ *and* $\mathcal{H}'$ *such that* $\texttt{e}; \mathcal{H} \rightsquigarrow \texttt{e}'; \mathcal{H}'$

or:

there exists $\texttt{v}$ *such that* $\texttt{e} = \texttt{v}$

Proof *by structural induction on the derivation of* $\emptyset; \mathcal{H} \vdash \texttt{e} : \texttt{T}$ *with a case analysis on the last step:*

Case 1 (T-Var)

1.	$\texttt{e} = \gamma$	*by def* T-Var
2.	$\texttt{T} = \mathcal{H}(\gamma)$	
3.	$\texttt{e} = \iota$	*by* **1**, **2**, *def* H-T
4.	*done by* $\texttt{e} = \texttt{v}$	*by* **3**

Case 2 (T-New)

1.	$\texttt{e} = \texttt{new C<}\overline{\mathcal{T}}\texttt{,}\ \tau_o\texttt{,}\ \star\ \texttt{>}$	*by def* T-New
2.	*done*	*by* RC-New

Case 3 (T-Field)

1.	$\texttt{e} = \texttt{e}_r\texttt{.f}$	*by def* T-Field
2.	$\emptyset; \mathcal{H} \vdash \texttt{e}_r : \exists\Delta.\texttt{N}$	*by premises* T-Field
3.	$\Downarrow_\Delta fType(\texttt{f}, \texttt{N}) = \texttt{T}$	
4.	$\texttt{e}_r; \mathcal{H} \rightsquigarrow \texttt{e}_r'; \mathcal{H}'$ *or there exists* $\texttt{v}_r$ *where* $\texttt{e}_r = \texttt{v}_r$	*by* **2**, *ind hyp*

Case analysis on $\texttt{e}_r$**:**

Case 1 $\texttt{e}_r; \mathcal{H} \rightsquigarrow \texttt{e}_r'; \mathcal{H}'$

1.1.	*done*	*by* RC-Field *or* RC-Field-Err

Case 2 **there exists** $\texttt{v}_r$ **where** $\texttt{e}_r = \texttt{v}_r$

2.1.	*if* $\texttt{v}_r = \texttt{null}$ *done by* $R - Field - Null$	
2.2.	*let* $\texttt{v}_r = \iota$	*by syntax of* $\texttt{v}$
2.3.	$\mathcal{H}(\iota) = \{\texttt{C<}\overline{\texttt{T}}\texttt{>}; \overline{\texttt{f}\rightarrow\texttt{v}}\}$	*by* **2**, **2.2**, *def* H-T
2.4.	$\emptyset \vdash \exists\emptyset.\texttt{C<}\overline{\texttt{T}}\texttt{>} <: \exists\Delta.\texttt{N}$	*by* **2.3**, **2**, *lemma 45*
2.5.	$fields(\texttt{C}) = \overline{\texttt{f}}$	*by* **b**, *def* F-Heap
2.6.	$\emptyset \vdash \exists\emptyset.\texttt{C<}\overline{\texttt{T}}\texttt{>} \sqsubset: \exists\Delta.\texttt{N}$	*by* **2.4**, F-Env-Empty, *lemma 16*
2.7.	*there exists* $\overline{\texttt{U}}$	*by* **2.6**, F-Env-Empty, *lemma 39*
2.8.	$dom(\Delta) = \overline{\texttt{Z}}$	
2.9.	$\vdash \texttt{C<}\overline{\texttt{T}}\texttt{>} \sqsubset: [\overline{\texttt{U/Z}}]\texttt{N}$	

2.10.	$fType(\texttt{f}, [\overline{\texttt{U/Z}}]\texttt{N}) = [\overline{\texttt{U/Z}}]fType(\texttt{f},\texttt{N})$	*by lemma 5*
2.11.	$fType(\texttt{f},\texttt{C<}\overline{\texttt{T}}\texttt{>}) = [\overline{\texttt{U/Z}}]fType(\texttt{f},\texttt{N})$	*by* **2.10**, **2.9**, *lemma 23*
2.12.	$\texttt{f} \in \overline{\texttt{f}}$	*by* **2.5**, **2.11**, **3**, *lemma 44*
2.13.	*done*	*by* **2.2**, **2.3**, **2.12**, R-Field

Case 4 (T-Assign)

1.	$\texttt{e} = \texttt{e}_1\texttt{.f = e}_2$	*by def* T-Field
2.	$\emptyset;\mathcal{H} \vdash \texttt{e}_1 : \exists\Delta.\texttt{N}$	*by premises* T-Assign
3.	$fType(\texttt{f},\texttt{N}) = \texttt{U}$	*by premises* T-Assign
4.	$\emptyset;\mathcal{H} \vdash \texttt{e}_2 : \texttt{T}$	*by premises* T-Assign
5.	$\emptyset \vdash \texttt{T} <: \texttt{U}$	*by premises* T-Assign
6.	$\texttt{e}_1;\mathcal{H} \rightsquigarrow \texttt{e}'_1;\mathcal{H}'$ *or there exists* $\texttt{v}$ *where* $\texttt{e}_1 = \texttt{v}$	*by* **2**, *ind hyp*
7.	$\texttt{e}_2;\mathcal{H} \rightsquigarrow \texttt{e}'_2;\mathcal{H}'$ *or there exists* $\texttt{v}$ *where* $\texttt{e}_2 = \texttt{v}$	*by* **4**, *ind hyp*

Case analysis on $\texttt{e}_1, \texttt{e}_2$:

Case 1 $\texttt{e}_1;\mathcal{H} \rightsquigarrow \texttt{e}'_1;\mathcal{H}'$, $\texttt{e}_2;\mathcal{H} \rightsquigarrow \texttt{e}'_2;\mathcal{H}'$

1.1.	*done*	*by* RC-Assign-1 *or* RC-Assign-1-Err

Case 2 $\texttt{e}_1;\mathcal{H} \rightsquigarrow \texttt{e}'_1;\mathcal{H}'$, *there exists* $\texttt{v}$ *where* $\texttt{e}_2 = \texttt{v}$

2.1.	*done*	*by* RC-Assign-1 *or* RC-Assign-1-Err

Case 3 *there exists* $\texttt{v}$ *where* $\texttt{e}_1 = \texttt{v}$, $\texttt{e}_2;\mathcal{H} \rightsquigarrow \texttt{e}'_2;\mathcal{H}'$

3.1.	*done*	*by* RC-Assign-2 *or* RC-Assign-2-Err

Case 4 *there exists* $\texttt{v}$ *where* $\texttt{e}_1 = \texttt{v}$, *there exists* $\texttt{v}'$ *where* $\texttt{e}_2 = \texttt{v}'$

4.1.	$if\ \texttt{v} = \texttt{null}\ done\ by\ R-Assign-Null$	
4.2.	*let* $\texttt{v} = \iota$	*by syntax of* $\texttt{v}$
4.3.	$\mathcal{H}(\iota) = \{\texttt{C<}\overline{\texttt{T}}\texttt{>};\overline{\texttt{f}\rightarrow\texttt{v}}\}$	*by* **2**, **4.2**, *def* H-T
4.4.	$\emptyset \vdash \exists\emptyset.\texttt{C<}\overline{\texttt{T}}\texttt{>} <: \exists\Delta.\texttt{N}$	*by* **4.3**, **2**, *lemma 45*
4.5.	$fields(\texttt{C}) = \overline{\texttt{f}}$	*by* **b**, *def* F-Heap
4.6.	$\emptyset \vdash \exists\emptyset.\texttt{C<}\overline{\texttt{T}}\texttt{>} \sqsubset: \exists\Delta.\texttt{N}$	*by* **4.4**, F-Env-Empty, *lemma 16*
4.7.	*there exists* $\overline{\texttt{U}}$	*by* **4.6**, F-Env-Empty, *lemma 39*
4.8.	$dom(\Delta) = \overline{\texttt{Z}}$	*by* **4.6**, F-Env-Empty, *lemma 39*
4.9.	$\vdash \texttt{C<}\overline{\texttt{T}}\texttt{>} \sqsubset\!\sqsubset: [\overline{\texttt{U/Z}}]\texttt{N}$	*by* **4.6**, F-Env-Empty, *lemma 39*
4.10.	$fType(\texttt{f}, [\overline{\texttt{U/Z}}]\texttt{N}) = [\overline{\texttt{U/Z}}]fType(\texttt{f},\texttt{N})$	*by lemma 5*
4.11.	$fType(\texttt{f},\texttt{C<}\overline{\texttt{T}}\texttt{>}) = [\overline{\texttt{U/Z}}]fType(\texttt{f},\texttt{N})$	*by* **4.10**, **4.9**, *lemma 23*
4.12.	$\texttt{f} \in \overline{\texttt{f}}$	*by* **4.5**, **4.11**, **3**, *lemma 44*
4.13.	*done*	*by* **4.2**, **4.3**, **4.12**, R-Assign

Case 5 (T-Subs)

1. $\emptyset;\mathcal{H} \vdash \texttt{e} : \texttt{U}$ — *by premise of* T-Subs
2. *done* — *by* **1**, **b**, *ind hyp*

Case 6 (T-Invk)

1. $\texttt{e} = \texttt{e}_r\texttt{.<}\overline{\texttt{P}}\texttt{>m(}\overline{\texttt{e}}\texttt{)}$ — *by def* T-Invk
2. $\emptyset;\mathcal{H} \vdash \texttt{e}_r : \exists\Delta'\texttt{.N}$
3. $\emptyset;\mathcal{H} \vdash \overline{\texttt{e} : \exists\Delta\texttt{.R}}$
4. $mType(\texttt{m},\texttt{N}) = \texttt{<}\overline{\texttt{Y}\lhd\ \texttt{T}_u}\texttt{>}\overline{\texttt{U}}\rightarrow\texttt{U}$
5. $match(sift(\overline{\texttt{R}},\overline{\texttt{U}},\overline{\texttt{Y}}),\overline{\texttt{P}},\overline{\texttt{Y}},\overline{\texttt{T}})$
6. $\emptyset \vdash \overline{\texttt{P}}$ OK

(2–6: *by premises* T-Invk)

7. $\texttt{e}_r;\mathcal{H} \rightsquigarrow \texttt{e}'_r\mathcal{H}'$ *or* (*there exists* $\texttt{v}_r$ *with* $\texttt{e}_r = \texttt{v}_r$) *by* **2**, *ind hyp*
8. $\forall\texttt{e}_i \in \overline{\texttt{e}}:$ (*there exists* $\texttt{v}_i$ *with* $\texttt{e}_i = \texttt{v}_i$) *or* (*there exists* $\texttt{e}_i \in \overline{\texttt{e}}:\ \texttt{e}_i;\mathcal{H} \rightsquigarrow \texttt{e}'_i;\mathcal{H}'$)
3, *ind hyp*
9. $\emptyset \vdash \exists\Delta'\texttt{.N}$ OK — *by* **2**, F-Env-Empty, *lemma 31*

Case analysis on $\texttt{e}_r, \overline{\texttt{e}}$:

Case 1 $\texttt{e}_r;\mathcal{H} \rightsquigarrow \texttt{e}'_r;\mathcal{H}'$

1.1. *done* — *by* RC-Inv-Recv *or* RC-Inv-Recv-Err

Case 2 **there exists** $\texttt{e}_i \in \overline{\texttt{e}}$ *where* $\texttt{e}_i;\mathcal{H} \rightsquigarrow \texttt{e}'_i;\mathcal{H}'$

2.1. *done* — *by* RC-Inv-Arg *or* RC-Inv-Arg-Err

Case 3 **there exists** $\texttt{v}_r$ *where* $\texttt{e}_r = \texttt{v}_r$ *and* $\forall\texttt{e}_i \in \overline{\texttt{e}}:\ \exists\texttt{v}_i$ *where* $\texttt{e}_i =$
$\texttt{v}_i$

3.1. $if\ \texttt{v}_r = \texttt{null}\ done\ by\ R - Invk - Null$
3.2. *let* $\texttt{v}_r = \iota$
3.3. *let* $\overline{\texttt{v}}_r = \overline{\iota}$
3.4. $\overline{\mathcal{H}(\iota) = \{\texttt{N}';\overline{\texttt{f}\rightarrow\texttt{v}}\}}$ — *by* **2**, **3.2**, *def* H-T
3.5. $\overline{\mathcal{H}(\iota) = \{\texttt{N}';\overline{\texttt{f}\rightarrow\texttt{v}}\}}$ — *by* **3**, **3.3**, *def* H-T
3.6. $\emptyset \vdash \exists\emptyset\texttt{.N}' <: \exists\Delta\texttt{.N}$ — *by* **3.3**, **3**, *lemma 45*
3.7. $\emptyset \vdash \texttt{N}'$ OK — *by* **b**, *def* F-Heap
3.8. $\emptyset \vdash \exists\emptyset\texttt{.N}' \sqsubset: \exists\Delta'\texttt{.N}$ — *by* **3.6**, F-Env-Empty, *lemma 16*
3.9. *let* $\Delta' = \overline{\texttt{Z}\rightarrow\texttt{[B}_l\ \texttt{B}_u\texttt{]}}$
3.10. $\vdash \texttt{N}' \sqsubset: \texttt{[}\overline{\texttt{U}'/\texttt{Z}}\texttt{]N}$
3.11. $\emptyset \vdash \overline{\texttt{U}' <: \texttt{[}\overline{\texttt{U}'/\texttt{Z}}\texttt{]B}_u}$
3.12. $\emptyset \vdash \overline{\texttt{[}\overline{\texttt{U}'/\texttt{Z}}\texttt{]B}_l <: \texttt{U}'}$
3.13. $fv(\overline{\texttt{U}'}) \subseteq dom(\Delta)$

(3.10–3.13: *by* **3.8**, **3.9**, *lemma 39*)

3.14. $mType(\mathtt{m}, [\overline{\mathtt{U'/Z}}]\mathtt{N}) = [\overline{\mathtt{U'/Z}}]\mathtt{<}\overline{\mathtt{Y}\lhd\ \mathtt{T}_u}\mathtt{>}\overline{\mathtt{U}}\to\mathtt{U}$	*by* **4**, *lemma 6*
3.15. $mType(\mathtt{m}, \mathtt{N'}) = [\overline{\mathtt{U'/Z}}]\mathtt{<}\overline{\mathtt{Y}\lhd\ \mathtt{T}_u}\mathtt{>}\overline{\mathtt{U}}\to\mathtt{U}$	*by* **3.14**, **3.10**, *lemma 24*
3.16. $mBody(\mathtt{m}, \mathtt{N'})$ *defined*	*by* **3.15**, *lemma 43*
3.17. $\emptyset \vdash \overline{\exists\Delta.\mathtt{R}}$ OK	*by* **3**, F-Env-Empty, *lemma 31*
3.18. $\overline{\Delta} \vdash \overline{\mathtt{T}}$ OK	*by* **6**, **3.17**, F-Env-Empty, **5**, *lemma 29*
3.19. $match(sift(\overline{\mathtt{R}}, \overline{[\overline{\mathtt{U'/Z}}]\mathtt{U}}, \overline{\mathtt{Y}}), \overline{\mathtt{P}}, \overline{\mathtt{Y}}, \overline{\mathtt{T}})$	*by* **5**, **3.15**, **4**, **3.8**, F-Env-Empty, **3.18**, *lemma 40*
3.20. $\emptyset \vdash \overline{\mathtt{N'}}$ OK	*by* **3.3**, **3**, *lemma 32* (3.20–3.21)
3.21. $\emptyset \vdash \overline{\exists\emptyset.\mathtt{N'} <: \exists\Delta.\mathtt{R}}$	
3.22. $\exists\overline{\mathtt{N}_{fresh}}$ *such that* $\overline{\mathtt{R}} = \overline{\mathtt{N}_{fresh}}$	*by* **3.17**
3.23. $\emptyset \vdash \overline{\exists\emptyset.\mathtt{N'} \sqsubset: \exists\Delta.\mathtt{R}}$	*by* **3.21**, **3.22**, F-Env-Empty, *lemma 16*
3.24. $\emptyset \vdash \overline{\exists\emptyset.\mathtt{N'}}$ OK	*by* **3.20**, F-Env-Empty, F-Exists
3.25. *let* $\overline{\mathtt{X}_s \to [\mathtt{B}_{sl}\ \ \mathtt{B}_{su}]} = \overline{\Delta}$	
3.26. *wlog assume* $\overline{\mathtt{X}_s}$ *are fresh*	*by* **3.25**, *Barendregt*
3.27. $match(sift(\overline{\mathtt{N'}}, \overline{[\overline{\mathtt{U'/Z}}]\mathtt{U}}, \overline{\mathtt{Y}}), \overline{\mathtt{P}}, \overline{\mathtt{Y}}, \overline{[\overline{\mathtt{U}_s/\mathtt{X}_s}]\mathtt{T}})$	*by* **3.23**, **3.19**, **3.26**, **3.25**, F-Env-Empty, **3.24**, **6**, *lemma 41* (3.27–3.30)
3.28. $\overline{\emptyset} \vdash \overline{\mathtt{U}_s <: [\overline{\mathtt{U}_s/\mathtt{X}_s}]\mathtt{B}_u}$	
3.29. $\overline{\emptyset} \vdash \overline{[\overline{\mathtt{U}_s/\mathtt{X}_s}]\mathtt{B}_l <: \mathtt{U}_s}$	
3.30. $\vdash \overline{\mathtt{N'} \sqsubset: [\overline{\mathtt{U}_s/\mathtt{X}_s}]\mathtt{R}}$	
3.31. *done*	*by* **3.4**, **3.5**, **3.15**, **3.16**, **3.27**, R-Invk

Case 7 (T-Cast)

1.	$\mathtt{e} = (T\text{'})e_0$	*by def* T-Cast
2.	$\emptyset; \mathcal{H} \vdash \mathtt{e}_0 : U$	*by premises* T-Invk (2–4)
3.	$\mathcal{H} \vdash T' <: U$	
4.	$\mathcal{H} \vdash T'$ OK	
5.	$\mathtt{e}_0; \mathcal{H} \rightsquigarrow \mathtt{e}'_0\mathcal{H}'$ *or* (*there exists* $\mathtt{v}_0$ *with* $\mathtt{e}_0 = \mathtt{v}_0$)	*by* **2**, *ind hyp*
6.	*done*	*by* **5**, *and* R-Cast, R-Cast-Null, R-Bad-C

Case 8 (T-Null)

1.	*easy*	*by syntax of values*

□