The Logic of Metaslang

Alessandro Coglio

May 6, 2007

DRAFT; PLEASE DO NOT DISTRIBUTE

1 Introduction

The Metaslang language [1] describes the construction and the manipulation of spec(ification)s and related entities (e.g. morphisms). A spec is a logical theory: it introduces symbols and asserts properties of those symbols; further properties are derived via logical inference.

This document formalizes the logic of the Metaslang language, i.e. which judgements can be asserted and how judgements are inferred from other judgements. This document does not describe the full semantics of the Metaslang language, but only its underlying logic. The full Metaslang semantics is formalized in [2]: that document is based on the formalization of the logic in this document. The Metaslang logic is defined on a relatively small "core" subset of Metaslang; more complex Metaslang constructs are defined in terms of the core construct [2].

The Metaslang logic consists of standard higher-order logic [3] plus record types, polymorphic types, predicate subtypes, and an if-then-else. Record types are straightfoward. Polymorphic types are as in the HOL system [4] and are not particularly difficult. Predicates subtypes are as in the PVS system [5]: they make well-typedness undecidable because in order to establish whether an expression of a supertype has also a subtype it is necessary to prove that the predicate that defines the subtype is true for the expression, in the context where the expression occurs. Thus, inference rules to derive theorems and inference rules to derive well-typedness judgements are mutually recursive, together with the rules to derive other kinds of judgements. Unlike an ordinary ternary function, the contexts to establish the well-typedness of the second and third argument of an if-then-else are extended with the truth and falsehood of the first argument; this is why if-then-else is a primitive construct in the Metaslang logic.

The rest of this section introduces the mathematical notation used in this document. §2 defines the core (abstract) syntax of the Metaslang logic while §3 defines its proof theory. §4 states properties of the syntax and proof theory. §5 defines a set-theoretic semantics of the Metaslang logic. Finally, §A collects all the (meta) proofs of the (meta) theorems stated in the other sections.

1.1 Notation

We define the Metaslang logic in the usual semi-formal notation consisting of naive set theory and natural language. However, it is possible to define the Metaslang logic in axiomatic set theory or any other sufficiently expressive formal language.[1]

The (meta) logical notations $=$, $\forall$, $\exists$, $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, and $.\not.$ (e.g. $\neq$) have the usual meaning.

The set-theoretic notations $\in$, $\emptyset$, $\{\ldots \mid \ldots\}$, $\{\ldots\}$, $\cup$, $\cap$, and $\subseteq$ have the usual meaning.

$\mathbf{N}$ is the set of natural numbers, i.e. $\{0, 1, 2, \ldots\}$.

If A and B are sets, $A - B$ is their difference, i.e. $\{x \in A \mid x \notin B\}$.

If A and B are sets, $A \times B$ is their cartesian product, i.e. $\{\langle a, b\rangle \mid a \in A \wedge b \in B\}$. This generalizes to $n > 2$ sets.

[1] For example, Peter Homeier has formalized the Metaslang logic in the logic of the HOL theorem prover (see Acknowledgments section for details).

If A and B are sets, $A + B$ is their disjoint union, i.e. $\{\langle 0, a\rangle \mid a \in A\} \cup \{\langle 1, b\rangle \mid b \in B\}$. The "tags" 0 and 1 are always left implicit. This generalizes to $n > 2$ sets.

If A and B are sets: $A \xrightarrow{\text{p}} B$ is the set of all partial functions from A to B, i.e. $\{f \subseteq A \times B \mid \forall \langle a, b_1\rangle, \langle a, b_2\rangle \in f.\ b_1 = b_2\}$; $A \to B$ is the set of all total functions from A to B, i.e. $\{f \in A \xrightarrow{\text{p}} B \mid \forall a \in A.\ \exists b \in B.\ \langle a, b\rangle \in f\}$; and $A \hookrightarrow B$ is the set of all total injective functions from A to B, i.e. $\{f \in A \to B \mid \forall \langle a_1, b\rangle, \langle a_2, b\rangle \in f.\ a_1 = a_2\}$. We write $f : A \xrightarrow{\text{p}} B$, $f : A \to B$, and $f : A \hookrightarrow B$ for $f \in A \xrightarrow{\text{p}} B$, $f \in A \to B$, and $f \in A \hookrightarrow B$, respectively.

If f is a function from A to B: $\mathcal{D}(f)$ is the domain of f, i.e. $\{a \in A \mid \exists b \in B.\ \langle a, b\rangle \in f\}$; $\mathcal{R}(f)$ is the range of f, i.e. $\{b \in B \mid \exists a \in A.\ \langle a, b\rangle \in f\}$.

If f is a function and $a \in \mathcal{D}(f)$, $f(a)$ denotes the unique value such that $\langle a, f(a)\rangle \in f$.

If f is a function and A is a set: $f{\downarrow_A}$ denotes the restriction of f to A, i.e. $\{\langle a, f(a)\rangle \mid a \in \mathcal{D}(f) \cap A\}$ (we may have $A \subseteq \mathcal{D}(f)$ or not); $f{\uparrow_A}$ denotes the restriction of f to $\mathcal{D}(f) - A$, i.e. $f{\downarrow_{\mathcal{D}(f)-A}}$.

If A is a set, $\mathcal{P}_\omega(A)$ is the set of all finite subsets of A, i.e. $\{S \subseteq A \mid S \text{ finite}\}$.

If A is a set, A^* is the set of all finite sequences of elements of A, i.e. $\{x_1, \ldots, x_n \mid x_1 \in A \wedge \ldots \wedge x_n \in A\}$; A^+, $A^{(*)}$, and $A^{(+)}$ are the subsets of A^* of non-empty sequences, sequences without repeated elements, and non-empty sequences without repeated elements, respectively.[2] The empty sequence is written ϵ. A sequence $x_1, \ldots, x_n$ is often written $\overline{x}$, leaving n implicit. The length of a sequence s is written $|s|$. When a sequence is written where a set is expected, it stands for the set of its elements.

2 Syntax

2.1 Names

We postulate the existence of an infinite set of names

$$\mathcal{N}$$

2.2 Types

We inductively define the set of types as

$$\begin{aligned} \mathit{Type} = {} & \{\mathsf{Bool}\} \\ & + \{\beta \mid \beta \in \mathcal{N}\} \\ & + \{\tau[\overline{T}] \mid \tau \in \mathcal{N} \ \wedge\ \overline{T} \in \mathit{Type}^*\} \\ & + \{T_1 \to T_2 \mid T_1, T_2 \in \mathit{Type}\} \\ & + \{f_1\ T_1 \times \cdots \times f_n\ T_n \mid \overline{f} \in \mathcal{N}^{(*)} \ \wedge\ \overline{T} \in \mathit{Type}^*\} \\ & + \{T|r \mid T \in \mathit{Type} \ \wedge\ r \in \mathit{Exp}\} \end{aligned}$$

where *Exp* is defined later.[3]

Explanation:

- There is a type Bool for boolean (i.e. truth) values.
- A name β is a type variable.
- A type instance $\tau[\overline{T}]$ is obtained by combining a type name τ with zero or more argument types $\overline{T}$. We may write $\tau[\epsilon]$ as just τ.
- An arrow type $T_1 \to T_2$ consists of a domain T_1 and a range T_2.

[2] Strictly speaking, our notation $x_1, \ldots, x_n$ for sequences may lead to ambiguities, e.g. if s_1 and s_2 are sequences, is s_1, s_2 the sequence of length 2 whose elements are s_1 and s_2, or is it the concatenation of s_1 and s_2? However, in this document, the intended meaning should be always clear from the symbols used and from mathematical context.

[3] Types depend on expressions, which depend on types. Thus, types and expressions are inductively defined together, not separately. Their definitions are presented separately only for readability.

- A record type $\prod_i f_i\ T_i$ consists of typed fields f_i. There is one record type that has no fields (i.e. $n = 0$), denoted $\prod \epsilon$, whose only inhabitant is the empty record. All the fields of a record type are distinct names.
- A restriction type $T|r$ is obtained by combining a type T with an expression r (meant to be a predicate on T). Restriction types create the dependency of types on expressions.

2.3 Expressions

We inductively define the set of expressions as

$$\begin{aligned}
\mathit{Exp} = {} & \{v \mid v \in \mathcal{N}\} \\
& + \{o[\overline{T}] \mid o \in \mathcal{N} \ \wedge\ \overline{T} \in \mathit{Type}^*\} \\
& + \{e_1\ e_2 \mid e_1, e_2 \in \mathit{Exp}\} \\
& + \{\lambda v{:}T.\ e \mid v \in \mathcal{N} \ \wedge\ T \in \mathit{Type} \ \wedge\ e \in \mathit{Exp}\} \\
& + \{e_1 \equiv e_2 \mid e_1, e_2 \in \mathit{Exp}\} \\
& + \{\mathsf{if}\ e_0\ e_1\ e_2 \mid e_0, e_1, e_2 \in \mathit{Exp}\} \\
& + \{\iota_T \mid T \in \mathit{Type}\} \\
& + \{\mathsf{proj}_{\prod_i f_i\ T_i}\ f_j \mid \textstyle\prod_i f_i\ T_i \in \mathit{Type}\}
\end{aligned}$$

Explanation:

- A name v is a variable.
- An op(eration) instance $o[\overline{T}]$ consists of an op name o and zero or more types $\overline{T}$ that instantiate the (generally, polymorphic) type of the op. We may write $o[\epsilon]$ as just o.
- An application $e_1\ e_2$ consists of a function e_1 juxtaposed to an argument e_2.
- A (lambda) abstraction $\lambda v{:}T.\ e$ consists of an argument v with an explicit type T and a body e.
- An equality $e_1 \equiv e_2$ consists of a left-hand side e_1 and a right-hand side e_2.
- A conditional $\mathsf{if}\ e_0\ e_1\ e_2$ consists of a condition e_0, a "then" branch e_1, and an "else" branch e_2.
- The description operator ι_T is tagged by a type. It operates on predicates over T (i.e. over values of type $T \to \mathsf{Bool}$) that are satisfied by a unique value of T, and its result is *the* value that satisfies the predicate.
- A projector $\mathsf{proj}_{\prod_i f_i\ T_i}\ f_j$ is tagged by a record type and by a field of that record type. We may write $\mathsf{proj}_{\prod_i f_i\ T_i}\ f_j$ as just $\mathsf{proj}\ f_j$ when the record type is inferrable or irrelevant.

We introduce the abbreviations

$$
\begin{array}{rcl}
\mathsf{true} & \longrightarrow & \lambda\gamma{:}\mathsf{Bool}.\ \gamma \equiv \lambda\gamma{:}\mathsf{Bool}.\ \gamma \\
\mathsf{false} & \longrightarrow & \lambda\gamma{:}\mathsf{Bool}.\ \gamma \equiv \lambda\gamma{:}\mathsf{Bool}.\ \mathsf{true} \\
\neg & \longrightarrow & \lambda\gamma{:}\mathsf{Bool}.\ (\gamma \equiv \mathsf{false}) \\
e_1 \wedge e_2 & \longrightarrow & \mathsf{if}\ e_1\ e_2\ \mathsf{false} \\
e_1 \vee e_2 & \longrightarrow & \mathsf{if}\ e_1\ \mathsf{true}\ e_2 \\
e_1 \Rightarrow e_2 & \longrightarrow & \mathsf{if}\ e_1\ e_2\ \mathsf{true} \\
\Leftrightarrow & \longrightarrow & \lambda\gamma{:}\mathsf{Bool}.\ \lambda\gamma'{:}\mathsf{Bool}.\ (\gamma \equiv \gamma') \\
e_1 \Leftrightarrow e_2 & \longrightarrow & \Leftrightarrow\ e_1\ e_2 \\
e_1 \not\equiv e_2 & \longrightarrow & \neg\,(e_1 \equiv e_2) \\
\iota v{:}T.e & \longrightarrow & \iota_T\ (\lambda v{:}T.\ e) \\
\forall_T & \longrightarrow & \lambda\psi{:}T \to \mathsf{Bool}.\ (\psi \equiv \lambda\gamma{:}T.\ \mathsf{true}) \\
\forall v{:}T.\ e & \longrightarrow & \forall_T\ (\lambda v{:}T.\ e) \\
\forall v_1{:}T_1, \ldots, v_n{:}T_n.\ e & \longrightarrow & \forall v_1{:}T_1.\ \ldots \forall v_n{:}T_n.\ e \\
\forall \overline{v}{:}\overline{T}.\ e & \longrightarrow & \forall v_1{:}T_1, \ldots, v_n{:}T_n.\ e \\
\exists_T & \longrightarrow & \lambda\psi{:}T \to \mathsf{Bool}.\ \neg\,(\forall\gamma{:}T.\ \neg\,(\psi\ \gamma)) \\
\exists v{:}T.\ e & \longrightarrow & \exists_T\ (\lambda v{:}T.\ e) \\
\exists v_1{:}T_1, \ldots, v_n{:}T_n.\ e & \longrightarrow & \exists v_1{:}T_1.\ \ldots \exists v_n{:}T_n.\ e \\
\exists \overline{v}{:}\overline{T}.\ e & \longrightarrow & \exists v_1{:}T_1, \ldots, v_n{:}T_n.\ e \\
\exists!_T & \longrightarrow & \lambda\psi{:}T \to \mathsf{Bool}.\ (\exists\gamma{:}T.\ (\psi\ \gamma \wedge \forall\gamma'{:}T.\ (\psi\ \gamma' \Rightarrow \gamma' \equiv \gamma))) \\
\exists!\, v{:}T.\ e & \longrightarrow & \exists!_T\ (\lambda v{:}T.\ e) \\
e.f & \longrightarrow & \mathsf{proj}\ f\ e
\end{array}
$$

where γ, γ', and ψ are fixed but unspecified names in $\mathcal{N}$ that are all distinct.

Explanation:

- The abbreviations true and false stand for logical truth and falsehood, respectively.
- Negation is defined as equality with false.
- The logical connectives for conjunction, disjunction, and implication are defined in terms of conditionals.
- Coimplication is a synonym for equality, but only for booleans.
- Inequality is negation of equality.
- Stand-alone quantifiers are higher-order functions, but they can be written in binder form. Also the description operator can be written in binder form. Note that both $\forall\epsilon.\ e$ and $\exists\epsilon.\ e$ stand for e.
- A dotted projection $e.f$ abbreviates an applied projection whose record type is left implicit.

The function $\mathcal{FV} : \mathit{Exp} \to \mathcal{P}_\omega(\mathcal{N})$ returns the free variables of an expression

$$
\begin{array}{ll}
\mathcal{FV}(v) & = \{v\} \\
\mathcal{FV}(o[\overline{T}]) & = \emptyset \\
\mathcal{FV}(e_1\ e_2) & = \mathcal{FV}(e_1) \cup \mathcal{FV}(e_2) \\
\mathcal{FV}(\lambda v{:}T.\ e) & = \mathcal{FV}(e) - \{v\} \\
\mathcal{FV}(e_1 \equiv e_2) & = \mathcal{FV}(e_1) \cup \mathcal{FV}(e_2) \\
\mathcal{FV}(\mathsf{if}\ e_0\ e_1\ e_2) & = \mathcal{FV}(e_0) \cup \mathcal{FV}(e_1) \cup \mathcal{FV}(e_2) \\
\mathcal{FV}(\iota_T) & = \emptyset \\
\mathcal{FV}(\mathsf{proj}\ f) & = \emptyset
\end{array}
$$

Note that we do not consider the free variables in expressions contained in types contained in expressions (e.g. we do not consider the free variables in r as part of the free variables of $o[T|r]$). The reason is that, as will be apparent later, all the expressions contained in well-formed types have no free variables.

2.4 Contexts

We define the set of context elements as

$$\begin{aligned} \mathit{CxElem} = {} & \{\mathsf{ty}\ \tau{:}n \mid \tau \in \mathcal{N}\ \wedge\ n \in \mathbf{N}\} \\ & + \{\mathsf{tvar}\ \beta \mid \beta \in \mathcal{N}\} \\ & + \{\mathsf{op}\ o{:}\{\overline{\beta}\}\ T \mid o \in \mathcal{N}\ \wedge\ \overline{\beta} \in \mathcal{N}^{(*)}\ \wedge\ T \in \mathit{Type}\} \\ & + \{\mathsf{var}\ v{:}T \mid v \in \mathcal{N}\ \wedge\ T \in \mathit{Type}\} \\ & + \{\mathsf{ax}\ \{\overline{\beta}\}\ e \mid \overline{\beta} \in \mathcal{N}^{(*)}\ \wedge\ e \in \mathit{Exp}\} \end{aligned}$$

Explanation:

- A type declaration $\mathsf{ty}\ \tau{:}n$ introduces a type name with an associated arity.
- A type variable declaration $\mathsf{tvar}\ \beta$ introduces a type variable. We may write $\mathsf{tvar}\ \beta_1, \ldots, \mathsf{tvar}\ \beta_n$ as just $\mathsf{tvar}\ \beta_1, \ldots, \beta_n$.
- An op(eration) declaration $\mathsf{op}\ o{:}\{\overline{\beta}\}\ T$ introduces an op name with an associated type, polymorphic in the explicit type variables.
- A variable declaration $\mathsf{var}\ v{:}T$ introduces a variable with a type.
- An axiom $\mathsf{ax}\ \{\overline{\beta}\}\ e$ introduces an expression (with type Bool, as defined later), polymorphic in the explicit type variables. We may write $\mathsf{ax}\ \{\epsilon\}\ e$ as just $\mathsf{ax}\ e$.

We define the set of contexts as

$$\mathit{Cx} = \mathit{CxElem}^*$$

In other words, a context is a finite sequence of context elements.

The function $\mathcal{TN} : \mathit{Cx} \to \mathcal{P}_\omega(\mathcal{N})$ returns the type names declared in a context

$$\begin{aligned} \mathcal{TN}(\epsilon) &= \emptyset \\ \mathcal{TN}(\mathit{cxel}, \mathit{cx}) &= \begin{cases} \mathcal{TN}(\mathit{cx}) \cup \{\tau\} & \text{if} \quad \mathit{cxel} = \mathsf{ty}\ \tau{:}n \\ \mathcal{TN}(\mathit{cx}) & \text{otherwise} \end{cases} \end{aligned}$$

The function $\mathcal{TV} : \mathit{Cx} \to \mathcal{P}_\omega(\mathcal{N})$ returns the type variables declared in a context

$$\begin{aligned} \mathcal{TV}(\epsilon) &= \emptyset \\ \mathcal{TV}(\mathit{cxel}, \mathit{cx}) &= \begin{cases} \mathcal{TV}(\mathit{cx}) \cup \{\beta\} & \text{if} \quad \mathit{cxel} = \mathsf{tvar}\ \beta \\ \mathcal{TV}(\mathit{cx}) & \text{otherwise} \end{cases} \end{aligned}$$

The function $\mathcal{ON} : \mathit{Cx} \to \mathcal{P}_\omega(\mathcal{N})$ returns the op names declared in a context

$$\begin{aligned} \mathcal{ON}(\epsilon) &= \emptyset \\ \mathcal{ON}(\mathit{cxel}, \mathit{cx}) &= \begin{cases} \mathcal{ON}(\mathit{cx}) \cup \{o\} & \text{if} \quad \mathit{cxel} = \mathsf{op}\ o{:}\{\overline{\beta}\}\ T \\ \mathcal{ON}(\mathit{cx}) & \text{otherwise} \end{cases} \end{aligned}$$

The function $\mathcal{V} : \mathit{Cx} \to \mathcal{P}_\omega(\mathcal{N})$ returns the variables declared in a context

$$\begin{aligned} \mathcal{V}(\epsilon) &= \emptyset \\ \mathcal{V}(\mathit{cxel}, \mathit{cx}) &= \begin{cases} \mathcal{V}(\mathit{cx}) \cup \{v\} & \text{if} \quad \mathit{cxel} = \mathsf{var}\ v{:}T \\ \mathcal{V}(\mathit{cx}) & \text{otherwise} \end{cases} \end{aligned}$$

2.5 Substitutions

2.5.1 Type substitutions

The function $\_[\_] : (\mathit{Type} + \mathit{Exp}) \times (\mathcal{N} \xrightarrow{\text{p}} \mathit{Type}) \to \mathit{Type} + \mathit{Exp}$ substitutes each type variable $\beta \in \mathcal{D}(\sigma)$, where $\sigma : \mathcal{N} \xrightarrow{\text{p}} \mathit{Type}$, with the type $\sigma(\beta)$ in a type or expression x (written $x[\sigma]$)

$$\begin{aligned}
\mathsf{Bool}[\sigma] &= \mathsf{Bool} \\
\beta[\sigma] &= \begin{cases} \sigma(\beta) & \text{if} \quad \beta \in \mathcal{D}(\sigma) \\ \beta & \text{otherwise} \end{cases} \\
\tau[\overline{T}][\sigma] &= \tau[\overline{T}[\sigma]] \\
(T_1 \to T_2)[\sigma] &= T_1[\sigma] \to T_2[\sigma] \\
(\textstyle\prod_i f_i\ T_i)[\sigma] &= \textstyle\prod_i f_i\ T_i[\sigma] \\
(T|r)[\sigma] &= T[\sigma]|r[\sigma]
\end{aligned}$$

$$\begin{aligned}
v[\sigma] &= v \\
o[\overline{T}][\sigma] &= o[\overline{T}[\sigma]] \\
(e_1\ e_2)[\sigma] &= e_1[\sigma]\ e_2[\sigma] \\
(\lambda v{:}T.\ e)[\sigma] &= \lambda v{:}T[\sigma].\ e[\sigma] \\
(e_1 \equiv e_2)[\sigma] &= e_1[\sigma] \equiv e_2[\sigma] \\
(\mathsf{if}\ e_0\ e_1\ e_2)[\sigma] &= \mathsf{if}\ e_0[\sigma]\ e_1[\sigma]\ e_2[\sigma] \\
\iota_T[\sigma] &= \iota_{T[\sigma]} \\
\left(\mathsf{proj}_{\prod_i f_i\ T_i}\ f_j\right)[\sigma] &= \mathsf{proj}_{(\prod_i f_i\ T_i)[\sigma]}\ f_j
\end{aligned}$$

where of course $(T_1, \ldots, T_n)[\sigma] = T_1[\sigma], \ldots, T_n[\sigma]$. Given $\overline{\beta} \in \mathcal{N}^{(*)}$ and $\overline{T} \in \mathit{Type}^*$ such that $|\overline{\beta}| = |\overline{T}|$, we may write $T[\{\langle \beta_i, T_i \rangle \mid 1 \le i \le n\}]$ as just $T[\overline{\beta}/\overline{T}]$.

2.5.2 Expression substitutions

The function $\_[\_/\_] : \mathit{Exp} \times \mathcal{N} \times \mathit{Exp} \to \mathit{Exp}$ substitutes a variable u with an expression d in an expression e (written $e[u/d]$)

$$\begin{aligned}
v[u/d] &= \begin{cases} d & \text{if} \quad u = v \\ v & \text{otherwise} \end{cases} \\
o[\overline{T}][u/d] &= o[\overline{T}] \\
(e_1\ e_2)[u/d] &= e_1[u/d]\ e_2[u/d] \\
(\lambda v{:}T.\ e)[u/d] &= \begin{cases} \lambda v{:}T.\ e & \text{if} \quad u = v \\ \lambda v{:}T.\ e[u/d] & \text{otherwise} \end{cases} \\
(e_1 \equiv e_2)[u/d] &= e_1[u/d] \equiv e_2[u/d] \\
(\mathsf{if}\ e_0\ e_1\ e_2)[u/d] &= \mathsf{if}\ e_0[u/d]\ e_1[u/d]\ e_2[u/d] \\
\iota_T[u/d] &= \iota_T \\
(\mathsf{proj}\ f)[u/d] &= \mathsf{proj}\ f
\end{aligned}$$

No substitution is performed in the expressions contained in types contained in expressions because, as already mentioned, such inner expressions have no free variables in well-formed types.

The function $\mathcal{CV} : \mathit{Exp} \times \mathcal{N} \to \mathcal{P}_\omega(\mathcal{N})$ returns the variables that would be captured if a variable u were substituted with those variables in an expression e (i.e. all the variables bound in e at the free occurrences of u in e)

$$\begin{aligned}
\mathcal{CV}(v, u) &= \emptyset \\
\mathcal{CV}(o[\overline{T}], u) &= \emptyset \\
\mathcal{CV}(e_1\ e_2, u) &= \mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u) \\
\mathcal{CV}(\lambda v{:}T.\ e, u) &= \begin{cases} \{v\} \cup \mathcal{CV}(e, u) & \text{if} \quad u \in \mathcal{FV}(e) - \{v\} \\ \emptyset & \text{otherwise} \end{cases} \\
\mathcal{CV}(e_1 \equiv e_2, u) &= \mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u) \\
\mathcal{CV}(\mathsf{if}\ e_0\ e_1\ e_2, u) &= \mathcal{CV}(e_0, u) \cup \mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u) \\
\mathcal{CV}(\iota_T, u) &= \emptyset \\
\mathcal{CV}(\mathsf{proj}\ f, u) &= \emptyset
\end{aligned}$$

If we view an expression e as a tree, each free occurrence of u in e is reachable via a path in the tree that starts from the root. That path contains zero or more lambda binders, each with its own variable. Then $\mathcal{CV}(e, u)$ is the set of all lambda-bound variables along all paths of all free occurrences of u in e.

The relation $\mathit{OKsbs} \subseteq \mathit{Exp} \times \mathcal{N} \times \mathit{Exp}$ captures the condition that the substitution $e[u/d]$ causes no free variables in d to be captured

$$\mathit{OKsbs}(e, u, d) \;\Leftrightarrow\; \mathcal{FV}(d) \cap \mathcal{CV}(e, u) = \emptyset$$

3 Proof theory

The proof theory of the Metaslang logic includes not only rules to derive formulas (theorems), but also rules to derive typing and other judgements. The rules to derive such judgements are mutually recursive; even though they are presented separately in the following subsections, the rules are inductively defined all together.

3.1 Well-formed contexts

We define a unary relation $\vdash \_ : \textsc{context} \subseteq \mathit{Cx}$ to capture well-formed contexts as

$$\frac{}{\vdash \epsilon : \textsc{context}} \quad (\textsc{cxMT})$$

$$\frac{\vdash cx : \textsc{context} \qquad \tau \notin \mathcal{TN}(cx)}{\vdash cx, \mathsf{ty}\ \tau : n : \textsc{context}} \quad (\textsc{cxTDec})$$

$$\frac{\vdash cx : \textsc{context} \qquad \beta \notin \mathcal{TV}(cx)}{\vdash cx, \mathsf{tvar}\ \beta : \textsc{context}} \quad (\textsc{cxTVDec})$$

$$\frac{cx, \mathsf{tvar}\ \overline{\beta} \vdash T : \textsc{type} \qquad o \notin \mathcal{ON}(cx)}{\vdash cx, \mathsf{op}\ o : \{\overline{\beta}\}\ T : \textsc{context}} \quad (\textsc{cxODec})$$

$$\frac{cx \vdash T : \textsc{type} \qquad v \notin \mathcal{V}(cx)}{\vdash cx, \mathsf{var}\ v : T : \textsc{context}} \quad (\textsc{cxVDec})$$

$$\frac{cx, \mathsf{tvar}\ \overline{\beta} \vdash e : \mathsf{Bool}}{\vdash cx, \mathsf{ax}\ \{\overline{\beta}\}\ e : \textsc{context}} \quad (\textsc{cxAx})$$

Eplanation:

- The empty context ϵ is well-formed. All the other rules add context elements to well-formed contexts (not all rules include $\vdash cx : \textsc{context}$ as an explicit premise, but $\vdash cx : \textsc{context}$ is derivable from their explicit premises, as proved by Theorem 4.37). Thus, a well-formed context is constructed incrementally starting with the empty one and adding elements.
- A type declaration $\mathsf{ty}\ \tau : n$ can be added to cx if τ is not already declared in cx.
- A type variable declaration $\mathsf{tvar}\ \beta$ can be added to cx if β is not already declared in cx.

- An op declaration $\mathsf{op}\ o : \{\overline{\beta}\}\ T$ can be added to cx if o is not already declared in cx. The op's type T must be well-formed (defined later) in cx extended with the type variables $\overline{\beta}$, which must be distinct.
- A variable declaration $\mathsf{var}\ v : T$ can be added to cx if v is not already declared in cx and T is well-formed in cx.
- A formula e can be added to cx as an axiom if e has type Bool (defined later). In general, the axiom may be polymorphic over type variables $\overline{\beta}$.

3.2 Well-formed types

We define a binary relation $\_ \vdash \_ : \textsc{type} \subseteq \mathit{Cx} \times \mathit{Type}$ to capture well-formed types as

$$\frac{\vdash cx : \textsc{context}}{cx \vdash \mathsf{Bool} : \textsc{type}} \quad (\textsc{tyBool})$$

$$\frac{\begin{array}{c}\vdash cx : \textsc{context} \\ \beta \in \mathcal{TV}(cx)\end{array}}{cx \vdash \beta : \textsc{type}} \quad (\textsc{tyVar})$$

$$\frac{\begin{array}{c}\vdash cx : \textsc{context} \\ \mathsf{ty}\ \tau : n \in cx \\ |\overline{T}| = n \\ \forall i.\ \ cx \vdash T_i : \textsc{type}\end{array}}{cx \vdash \tau[\overline{T}] : \textsc{type}} \quad (\textsc{tyInst})$$

$$\frac{cx, \mathsf{var}\ v : T_1 \vdash T_2 : \textsc{type}}{cx \vdash T_1 \to T_2 : \textsc{type}} \quad (\textsc{tyArr})$$

$$\frac{\begin{array}{c}\vdash cx : \textsc{context} \\ \forall i.\ \ cx \vdash T_i : \textsc{type}\end{array}}{cx \vdash \prod_i f_i\ T_i : \textsc{type}} \quad (\textsc{tyRec})$$

$$\frac{\begin{array}{c}cx \vdash r : T \to \mathsf{Bool} \\ \mathcal{FV}(r) = \emptyset\end{array}}{cx \vdash T|r : \textsc{type}} \quad (\textsc{tyRestr})$$

Explanation:

- The type Bool is well-formed in any well-formed context.
- A type variable is a well-formed type in any well-formed context that declares it.
- A type instance $\tau[\overline{T}]$ is well-formed in any well-formed context cx that declares τ if the argument types are well-formed in cx and their number matches the arity of τ.
- An arrow type $T_1 \to T_2$ is well-formed in a context cx if its domain type T_1 is well-formed in cx and its range type T_2 is well-formed in cx extended with a variable v of type T_1. The judgement $cx \vdash T_1 : \textsc{type}$ is not an explicit premise of the rule because, as proved later, it is a consequence of the existing premise $cx, \mathsf{var}\ v : T_1 \vdash T_2 : \textsc{type}$. Allowing cx to be extended with $\mathsf{var}\ v : T_1$ for the well-formedness of T_2 makes sense because if T_1 is empty then $T_1 \to T_2$ contains just the empty

function (i.e. the function with empty domain) and the well-formedness of T_2 is irrelevant.[4] Even though T_2 cannot contain v (because well-formed types have no free variables), if Metaslang were extended with dependent arrow types[5] then T_2 would be allowed to contain v and the form of the rule would seem more natural than it does now. If we used a more restrictive rule with premises $cx \vdash T_1 : \text{TYPE}$ and $cx \vdash T_2 : \text{TYPE}$, then Theorem 4.90 would not hold (see explanation near that theorem).

- The rule for record types is straightforward.
- For restriction types, we require the predicate to have type $T \to \mathsf{Bool}$, which implies that T is a well-formed type (as proved later). We also require that r has no free variables.

3.3 Subtyping

We define a quaternary relation $\_ \vdash \_ \prec_\_ \_ \subseteq \mathit{Cx} \times \mathit{Type} \times \mathit{Exp} \times \mathit{Type}$ to capture subtyping as

$$\frac{cx \vdash T|r : \text{TYPE}}{cx \vdash T|r \prec_r T} \quad (\text{stRestr})$$

$$\frac{\begin{array}{c} cx \vdash T : \text{TYPE} \\ r = \lambda v{:}T.\ \mathsf{true} \end{array}}{cx \vdash T \prec_r T} \quad (\text{stRefl})$$

$$\frac{\begin{array}{c} cx, \mathsf{var}\ v{:}T \vdash T_1 \prec_r T_2 \\ v' \neq v'' \\ r' = \lambda v'{:}T \to T_2.\ \forall v''{:}T.\ r\ (v'\ v'') \end{array}}{cx \vdash T \to T_1 \prec_{r'} T \to T_2} \quad (\text{stArr})$$

$$\frac{\begin{array}{c} cx \vdash \prod_i f_i\ T_i : \text{TYPE} \\ \forall i.\ \ cx \vdash T_i \prec_{r_i} T'_i \\ P : \{1,\ldots,n\} \hookrightarrow \{1,\ldots,n\} \\ r = \lambda v{:}\prod_i f_{P(i)}\ T'_{P(i)}.\ \bigwedge_i r_i\ v.f_i \end{array}}{cx \vdash \prod_i f_i\ T_i \prec_r \prod_i f_{P(i)}\ T'_{P(i)}} \quad (\text{stRec})$$

Explanation:

- Unsurprisingly, a restriction type $T|r$ is a subtype of T, with r being the predicate over the supertype T that identifies the values that are also in the subtype $T|r$.
- Subtyping is reflexive, i.e. a (well-formed) type T is a subtype of itself and the associated subtype predicate is always true.
- Arrow types are monotonic in their range types with respect to subtyping. The associated predicate holds when all the values of the function satisfy the predicate associated to the range subtype. Note that the domain must be the same; while domain contravariance is used in some type systems with subtypes, it would violate extensionality (see explanation in [5]). The extension of cx with $\mathsf{var}\ v{:}T$ mirrors rule tyArr: if T is empty, then both $T \to T_1$ and $T \to T_2$ contain just the empty function and the former is trivially a subtype of the latter.

[4] The declaration $\mathsf{var}\ v : T_1$ enables the derivation of the formula $\exists v : T_1.\ \mathsf{true}$, which is not otherwise derivable in general, because T_1 may be empty. Thus, T_2 may be well-formed in $cx, \mathsf{var}\ v{:}T_1$ without necessarily being well-formed in cx. An example is a type T_2 that contains an expression $\mathsf{if}\ (\exists v{:}T_1.\ \mathsf{true})\ e_1\ e_2$ where e_2 is not well-typed in a consistent context because some theorem does not hold (e.g. that the divisor of a division is not 0); if $\exists v : T_1.\ \mathsf{true}$ is a theorem, then the context of the else branch is inconsistent (see rule exIf later) and therefore e_2 is well-typed in that inconsistent context.

[5] Such an extension is indeed planned.

- Record types are monotonic in their component types with respect to subtyping. The record subtype predicate holds when all the component subtype predicates hold on the record components. The components can be re-ordered, i.e. the order of record components is irrelevant. In the rule, the permutation is captured by a bijective function P on the field indices $\{1, \ldots, n\}$ (the rule explicitly says that P is injective only, but since domain and codomain are finite and equal, it follows that P is also surjective, hence bijective).

We do not need a transitivity rule for subtyping. As defined later, subtyping judgements are only used to assign types to expressions, e.g. to assign a supertype to an expression of a subtype. So, instead of using transitivity of subtyping, rules for well-typed expressions can be applied multiple times, achieving the same effect.

3.4 Well-typed expressions

We define a ternary relation $\_ \vdash \_ : \_ \subseteq \mathit{Cx} \times \mathit{Exp} \times \mathit{Type}$ to capture well-typed expressions as

$$\frac{\vdash cx : \textsc{context} \quad \mathsf{var}\ v{:}T \in cx}{cx \vdash v : T} \quad (\textsc{exVar})$$

$$\frac{\vdash cx : \textsc{context} \quad \mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx \quad \forall i.\ \ cx \vdash T_i : \textsc{type}}{cx \vdash o[\overline{T}] : T[\overline{\beta}/\overline{T}]} \quad (\textsc{exOp})$$

$$\frac{cx \vdash e_1 : T_1 \to T_2 \quad cx \vdash e_2 : T_1}{cx \vdash e_1\ e_2 : T_2} \quad (\textsc{exApp})$$

$$\frac{cx, \mathsf{var}\ v{:}T \vdash e : T'}{cx \vdash \lambda v{:}T.\ e : T \to T'} \quad (\textsc{exAbs})$$

$$\frac{cx \vdash e_1 : T \quad cx \vdash e_2 : T}{cx \vdash e_1 \equiv e_2 : \mathsf{Bool}} \quad (\textsc{exEq})$$

$$\frac{cx \vdash e_0 : \mathsf{Bool} \quad cx, \mathsf{ax}\ e_0 \vdash e_1 : T \quad cx, \mathsf{ax}\ \neg\ e_0 \vdash e_2 : T \quad cx \vdash T : \textsc{type}}{cx \vdash \mathsf{if}\ e_0\ e_1\ e_2 : T} \quad (\textsc{exIf})$$

$$\frac{cx \vdash T : \textsc{type}}{cx \vdash \iota_T : (T \to \mathsf{Bool}) | \exists!_T \to T} \quad (\textsc{exThe})$$

$$\frac{cx \vdash \prod_i f_i\ T_i : \textsc{type}}{cx \vdash \mathsf{proj}\ f_j : \prod_i f_i\ T_i \to T_j} \quad (\textsc{exProj})$$

$$\frac{cx \vdash e : T \quad cx \vdash T \prec_r T'}{cx \vdash e : T'} \quad (\textsc{exSuper})$$

$$\frac{cx \vdash e : T' \quad cx \vdash T \prec_r T' \quad cx \vdash r\ e}{cx \vdash e : T} \quad (\textsc{exSub})$$

$$\frac{cx \vdash \lambda v{:}T.\ e : T' \quad v' \notin \mathcal{FV}(e) \cup \mathcal{CV}(e, v)}{cx \vdash \lambda v'{:}T.\ e[v/v'] : T'} \quad (\textsc{exAbsAlpha})$$

Explanation:

- A variable v declared in a well-formed context is a well-typed expression with the type T given in its declaration.
- An op o declared in a well-formed context can be instantiated via well-formed types $\overline{T}$ whose number matches the number of type variables $\overline{\beta}$. The result is a well-formed expression whose type is obtained by substituting $\overline{\beta}$ with $\overline{T}$ in the declared type T of o.
- An application is well-typed if the function has an arrow type and the argument has the domain type of the arrow type. The type of the application is the range type of the arrow type.
- An abstraction is well-typed if the body is well-typed in the context extended with the declaration of the variable bound by the abstraction. The type of the abstraction is the arrow type that has the type of the variable as domain and the type of the body as range.
- An equality is well-typed with type Bool if the left- and right-hand sides are both well-typed with a common type T.
- In the rule exIf for conditionals, the two branches must be well-typed, with a common type, in the context where the condition holds and does not hold, respectively. The additional assumption about the condition holding or not is realized by adding an axiom to the context. The premise $cx \vdash T : \textsc{type}$ in exIf may be redundant, i.e. derivable from the others, but this fact has not been established yet and so for now we have it as an explicit premise, which is needed to prove Theorem 4.90.
- The description operator for a well-formed type is well-typed and denotes a function from the predicates over the type that are satisfied by a unique value to the type itself.
- A projector for a well-formed record type is well-typed and denotes a function from the record type to the corresponding component type.
- Rules exSuper and exSub link the notion of well-typed expressions to the notion of subtyping. If an expression e has a subtype T, it also has any supertype T'. If an expression e has a supertype T', it also has any subtype T such that the associated predicate holds on e.
- The last rule amounts to treating expressions up to alpha equivalence, allowing to rename bound variables maintaining well-typedness. Without this rule, the Metaslang logic would be non-monotonic, because extending a context with variable declarations may invalidate conclusions about the well-typedness of expressions that bind those variables (e.g. if $cx \vdash \lambda v{:}T.\ e : T'$ were provable then $cx, \mathsf{var}\ v{:}T \vdash \lambda v{:}T.\ e : T'$ would not be provable).

3.5 Theorems

We define a binary relation $\_ \vdash \_ \subseteq Cx \times Exp$ to capture theorems as

$$\frac{\begin{array}{c}\vdash cx : \textsc{context} \\ \mathsf{ax}\ \{\overline{\beta}\}\ e \in cx \\ \forall i.\ \ cx \vdash T_i : \textsc{type}\end{array}}{cx \vdash e[\overline{\beta}/\overline{T}]}\quad (\textsc{thAx})$$

$$\frac{cx \vdash e : \ldots}{cx \vdash e \equiv e}\quad (\textsc{thRefl})$$

$$\frac{cx \vdash e_1 \equiv e_2}{cx \vdash e_2 \equiv e_1}\quad (\textsc{thSymm})$$

$$\frac{\begin{array}{c}cx \vdash e_1 \equiv e_2 \\ cx \vdash e_2 \equiv e_3\end{array}}{cx \vdash e_1 \equiv e_3}\quad (\textsc{thTrans})$$

$$\frac{\begin{array}{c}cx \vdash e_1\ e_2 : \ldots \\ cx \vdash e_1 \equiv e_1' \\ cx \vdash e_2 \equiv e_2'\end{array}}{cx \vdash e_1\ e_2 \equiv e_1'\ e_2'}\quad (\textsc{thAppSubst})$$

$$\frac{\begin{array}{c}cx \vdash e_1 \equiv e_2 : \ldots \\ cx \vdash e_1 \equiv e_1' \\ cx \vdash e_2 \equiv e_2'\end{array}}{cx \vdash (e_1 \equiv e_2) \equiv (e_1' \equiv e_2')}\quad (\textsc{thEqSubst})$$

$$\frac{\begin{array}{c}cx \vdash \mathsf{if}\ e_0\ e_1\ e_2 : \ldots \\ cx \vdash e_0 \equiv e_0' \\ cx, \mathsf{ax}\ e_0 \vdash e_1 \equiv e_1' \\ cx, \mathsf{ax}\ \neg\ e_0 \vdash e_2 \equiv e_2'\end{array}}{cx \vdash \mathsf{if}\ e_0\ e_1\ e_2 \equiv \mathsf{if}\ e_0'\ e_1'\ e_2'}\quad (\textsc{thIfSubst})$$

$$\frac{\begin{array}{c}cx \vdash e \\ cx \vdash e \equiv e'\end{array}}{cx \vdash e'}\quad (\textsc{thSubst})$$

$$\frac{\begin{array}{c}\vdash cx : \textsc{context} \\ v \neq v'\end{array}}{cx \vdash \forall v : \mathsf{Bool} \to \mathsf{Bool}.\ (v\ \mathsf{true} \wedge v\ \mathsf{false} \Leftrightarrow (\forall v' : \mathsf{Bool}.\ v\ v'))}\quad (\textsc{thBool})$$

$$\frac{\begin{array}{c}cx \vdash T \to T' : \textsc{type} \\ v \neq v' \ \wedge\ v' \neq v'' \ \wedge\ v'' \neq v\end{array}}{cx \vdash \forall v : T \to T', v' : T \to T'.\ (v \equiv v' \Leftrightarrow (\forall v'' : T.\ v\ v'' \equiv v'\ v''))}\quad (\textsc{thExt})$$

$$\frac{\begin{array}{c}cx \vdash (\lambda v : T.\ e)\ e' : \ldots \\ OKsbs(e, v, e')\end{array}}{cx \vdash (\lambda v : T.\ e)\ e' \equiv e[v/e']}\quad (\textsc{thAbs})$$

$$\frac{\begin{array}{c} cx \vdash \mathsf{if}\ e_0\ e_1\ e_2 : \ldots \\ cx, \mathsf{ax}\ e_0 \vdash e_1 \equiv e \\ cx, \mathsf{ax}\ \neg\ e_0 \vdash e_2 \equiv e \end{array}}{cx \vdash \mathsf{if}\ e_0\ e_1\ e_2 \equiv e} \quad (\textsc{thIf})$$

$$\frac{cx \vdash \iota_T\ e : T}{cx \vdash e\ (\iota_T\ e)} \quad (\textsc{thThe})$$

$$\frac{\begin{array}{c} cx \vdash \prod_i f_i\ T_i : \textsc{type} \\ v \neq v' \end{array}}{cx \vdash \forall v{:}\prod_i f_i\ T_i, v'{:}\prod_i f_i\ T_i.\ ((\bigwedge_i v.f_i \equiv v'.f_i) \Rightarrow v \equiv v')} \quad (\textsc{thRec})$$

$$\frac{cx \vdash \prod_i f_i\ T_i \prec_r \prod_i f_i\ T'_i}{cx \vdash \forall v{:}\prod_i f_i\ T_i.\ \mathsf{proj}_{\prod_i f_i\ T_i}\ f_j\ v \equiv \mathsf{proj}_{\prod_i f_i\ T'_i}\ f_j\ v} \quad (\textsc{thProjSub})$$

$$\frac{\begin{array}{c} cx \vdash T \prec_r T' \\ cx \vdash e : T \end{array}}{cx \vdash r\ e} \quad (\textsc{thSub})$$

Explanation:

- Axioms in a well-formed context are readily instantiated into theorems, via rule thAx. More precisely, the type variables over which the axiom is polymorphic are replaced with well-formed types.
- Rules thRefl, thSymm, and thTrans say that equality is an equivalence, i.e. reflexive, symmetric, and transitive.
- Rules thAppSubst, thEqSubst, and thIfSubst state the substitutivity of equalities in expressions. Note that in rule thIfSubst the context is extended with an axiom saying that the condition is true or false.
- Rule thSubst says that anything equal to a theorem is itself a theorem.
- Rule thBool asserts that true and false are the only values of type Bool.
- Rule thExt says that a function is characterized by its values over all the values of its domain, i.e. by extensionality.
- Rule thAbs defines the semantics of lambda abstraction: the bound variable v is replaced with the argument e' in the body e. The premise of the rule says that the application is well-typed.
- Rule thIf defines the semantics of conditionals: a conditional equals an expression if both branches do, in the contexts extended with the assumption that the condition is true and false, respectively. Note the premise that requires the conditional to be well-typed.
- Rule thThe says that a description satisfies the predicate associated to the description.
- Rule thRec says that a record is characterized by the values of its components.
- Rule thProjSub says that projectors for record subtypes agree with projectors for record supertypes.
- Rule thSub says that every value of a subtype satisfies the predicate that characterizes the subtype with respect to a supertype.

3.6 Proofs

The previous subsections have defined judgements of the forms

$$\begin{array}{l} \vdash cx : \textsc{context} \\ cx \vdash T : \textsc{type} \\ cx \vdash T_1 \prec_r T_2 \\ cx \vdash e : T \\ cx \vdash e \end{array}$$

by means of a set of inductive rules.

A proof of a judgement is a finite sequence of judgements that ends with the proved judgement and where each judgement in the sequence is derived from preceding judgements using some rule.

[[[TO DO: Make sure that the rules for theorems are "sufficient", i.e. all truths "of interest" are indeed theorems derivable from the rules. Even though higher-order logic is notoriously incomplete, in practice theorem provers like PVS and HOL are sufficient to prove desired properties of formalized concepts without running into theoretical limitations. Perhaps the requirement boils down to prove completeness with respect to so-called "general models" (cf. [3]).]]]

4 Properties

4.1 Additional notions

In order to prove certain properties of the syntax and proof theory formalized in §2 and §3, we introduce some additional notions. Those notions have not been introduced earlier because they are not needed to define the Metaslang logic, but only to express and prove properties about it.

We extend expression substitution to context elements and contexts

$$\begin{array}{rl} (\mathsf{ty}\ \tau{:}n)[u/d] &= \mathsf{ty}\ \tau{:}n \\ (\mathsf{op}\ o{:}\{\overline{\beta}\}\ T)[u/d] &= \mathsf{op}\ o{:}\{\overline{\beta}\}\ T \\ (\mathsf{ax}\ \{\overline{\beta}\}\ e)[u/d] &= \mathsf{ax}\ \{\overline{\beta}\}\ e[u/d] \\ (\mathsf{tvar}\ \beta)[u/d] &= \mathsf{tvar}\ \beta \\ (\mathsf{var}\ v{:}T)[u/d] &= \mathsf{var}\ v{:}T \end{array}$$

$$\begin{array}{rl} \epsilon[u/d] &= \epsilon \\ (cx_1, cx_2)[u/d] &= cx_1[u/d], cx_2[u/d] \end{array}$$

Note that $(\mathsf{var}\ v{:}T)[u/d] = \mathsf{var}\ v{:}T$ even if $u = v$, because a variable declaration "works" more or less like a binder (and the v in $\lambda v{:}T.\ e$ is unchanged under substitution).

The function $\mathcal{FTV} : \mathit{Type} + \mathit{Exp} + \mathit{CxElem} + \mathit{Cx} \to \mathcal{P}_\omega(\mathcal{N})$ returns the free type variables of a type, expression, or context (element)

$$\begin{array}{ll} \mathcal{FTV}(\mathsf{Bool}) & = \emptyset \\ \mathcal{FTV}(\beta) & = \{\beta\} \\ \mathcal{FTV}(\tau[\overline{T}]) & = \bigcup_i \mathcal{FTV}(T_i) \\ \mathcal{FTV}(T_1 \to T_2) & = \mathcal{FTV}(T_1) \cup \mathcal{FTV}(T_2) \\ \mathcal{FTV}(\prod_i f_i\ T_i) & = \bigcup_i \mathcal{FTV}(T_i) \\ \mathcal{FTV}(T|r) & = \mathcal{FTV}(T) \cup \mathcal{FTV}(r) \end{array}$$

$$\begin{array}{ll} \mathcal{FTV}(v) & = \emptyset \\ \mathcal{FTV}(o[\overline{T}]) & = \bigcup_i \mathcal{FTV}(T_i) \\ \mathcal{FTV}(e_1\ e_2) & = \mathcal{FTV}(e_1) \cup \mathcal{FTV}(e_2) \\ \mathcal{FTV}(\lambda v{:}T.\ e) & = \mathcal{FTV}(T) \cup \mathcal{FTV}(e) \\ \mathcal{FTV}(e_1 \equiv e_2) & = \mathcal{FTV}(e_1) \cup \mathcal{FTV}(e_2) \\ \mathcal{FTV}(\mathsf{if}\ e_0\ e_1\ e_2) & = \mathcal{FTV}(e_0) \cup \mathcal{FTV}(e_1) \cup \mathcal{FTV}(e_2) \\ \mathcal{FTV}(\iota_T) & = \mathcal{FTV}(T) \\ \mathcal{FTV}(\mathsf{proj}_{\prod_i f_i\ T_i}\ f_j) & = \mathcal{FTV}(\prod_i f_i\ T_i) \end{array}$$

$$
\begin{array}{lll}
\mathcal{FTV}(\mathsf{ty}\ \tau{:}n) & = \emptyset \\
\mathcal{FTV}(\mathsf{op}\ o{:}\{\overline{\beta}\}\ T) & = \mathcal{FTV}(T) - \overline{\beta} \\
\mathcal{FTV}(\mathsf{ax}\ \{\overline{\beta}\}\ e) & = \mathcal{FTV}(e) - \overline{\beta} \\
\mathcal{FTV}(\mathsf{tvar}\ \beta) & = \emptyset \\
\mathcal{FTV}(\mathsf{var}\ v{:}T) & = \mathcal{FTV}(T)
\end{array}
$$

$$
\begin{array}{ll}
\mathcal{FTV}(\epsilon) & = \emptyset \\
\mathcal{FTV}(cx_1, cx_2) & = \mathcal{FTV}(cx_1) \cup \mathcal{FTV}(cx_2)
\end{array}
$$

Note that the type variables that occur in a type or expression are always free; the only bound type variables are the $\overline{\beta}$ in context elements.

We extend type substitution to context elements, contexts, and type substitutions themselves

$$
\begin{array}{rl}
(\mathsf{ty}\ \tau{:}n)[\sigma] & = \mathsf{ty}\ \tau{:}n \\
(\mathsf{op}\ o{:}\{\overline{\beta}\}\ T)[\sigma] & = \mathsf{op}\ o{:}\{\overline{\beta}\}\ T[\sigma\!\uparrow_{\overline{\beta}}] \\
(\mathsf{ax}\ \{\overline{\beta}\}\ e)[\sigma] & = \mathsf{ax}\ \{\overline{\beta}\}\ e[\sigma\!\uparrow_{\overline{\beta}}] \\
(\mathsf{tvar}\ \beta)[\sigma] & = \mathsf{tvar}\ \beta \\
(\mathsf{var}\ v{:}T)[\sigma] & = \mathsf{var}\ v{:}T[\sigma]
\end{array}
$$

$$
\begin{array}{rl}
\epsilon[\sigma] & = \epsilon \\
(cx_1, cx_2)[\sigma] & = cx_1[\sigma], cx_2[\sigma]
\end{array}
$$

$$
\sigma[\sigma'] = \{\langle \beta, \sigma(\beta)[\sigma'] \rangle \mid \beta \in \mathcal{D}(\sigma)\}
$$

The function $\mathcal{CTV} : (\mathit{CxElem} + \mathit{Cx}) \times \mathcal{N} \to \mathcal{P}_\omega(\mathcal{N})$ returns the type variables that would be captured if a type variable α were substituted with those type variables in a context (element), i.e. all the variables bound in the (context) element at the free occurrences of α in the context (element)

$$
\begin{array}{ll}
\mathcal{CTV}(\mathsf{ty}\ \tau{:}n, \alpha) & = \emptyset \\
\mathcal{CTV}(\mathsf{op}\ o{:}\{\overline{\beta}\}\ T, \alpha) & = \begin{cases} \overline{\beta} & \text{if} \quad \alpha \in \mathcal{FTV}(T) - \overline{\beta} \\ \emptyset & \text{otherwise} \end{cases} \\
\mathcal{CTV}(\mathsf{ax}\ \{\overline{\beta}\}\ e, \alpha) & = \begin{cases} \overline{\beta} & \text{if} \quad \alpha \in \mathcal{FTV}(e) - \overline{\beta} \\ \emptyset & \text{otherwise} \end{cases} \\
\mathcal{CTV}(\mathsf{tvar}\ \beta, \alpha) & = \emptyset \\
\mathcal{CTV}(\mathsf{var}\ v{:}T, \alpha) & = \emptyset
\end{array}
$$

$$
\begin{array}{ll}
\mathcal{CTV}(\epsilon, \alpha) & = \emptyset \\
\mathcal{CTV}((cx_1, cx_2), \alpha) & = \mathcal{CTV}(cx_1, \alpha) \cup \mathcal{CTV}(cx_2, \alpha)
\end{array}
$$

The relation $\mathit{OKsbs} \subseteq \mathit{Cx} \times (\mathcal{N} \xrightarrow{\mathrm{p}} \mathit{Type})$ captures the condition that the substitution $cx[\sigma]$ causes no free type variables in cx to be captured and does not substitute any type variable declared in cx

$$
\mathit{OKsbs}(cx, \sigma) \ \Leftrightarrow\ (\forall \beta \in \mathcal{D}(\sigma).\ \ \mathcal{FTV}(\sigma(\beta)) \cap \mathcal{CTV}(cx, \beta) = \emptyset) \ \wedge\ \mathcal{D}(\sigma) \cap \mathcal{TV}(cx) = \emptyset
$$

Given $\overline{\beta} \in \mathcal{N}^{(*)}$ and $\overline{T} \in \mathit{Type}^*$ such that $|\overline{\beta}| = |\overline{T}|$, we may write $\mathit{OKsbs}(cx, \{\langle \beta_i, T_i \rangle \mid 1 \le i \le n\})$ as just $\mathit{OKsbs}(cx, \overline{\beta}, \overline{T})$.

4.2 Syntactic properties

Expression substitution only takes place at the free occurrences of the variable that is substituted. Thus, if the variable is not free in the expression, substitution causes no change and no variable capture:

Theorem 4.1

$$
u \notin \mathcal{FV}(e) \ \Rightarrow\ \mathcal{CV}(e, u) = \emptyset \ \wedge\ \mathit{OKsbs}(e, u, d) \ \wedge\ e[u/d] = e
$$

Since expression substitution only substitutes the free occurrences of the variable, the variable itself is not captured at those occurrences:

Theorem 4.2

$$u \notin \mathcal{CV}(e, u)$$

It is always possible to substitute a variable with itself and the substitution causes no change:

Theorem 4.3

$$OKsbs(e, u, u) \ \wedge \ e[u/u] = e$$

The following four theorems say that if a substitution causes no capture in a compound expression, it does not cause capture in the component expressions either:

Theorem 4.4

$$OKsbs(e_1\ e_2, u, d) \ \Rightarrow \ OKsbs(e_1, u, d) \ \wedge \ OKsbs(e_2, u, d)$$

Theorem 4.5

$$OKsbs(e_1 \equiv e_2, u, d) \ \Rightarrow \ OKsbs(e_1, u, d) \ \wedge \ OKsbs(e_2, u, d)$$

Theorem 4.6

$$OKsbs(\mathsf{if}\ e_0\ e_1\ e_2, u, d) \ \Rightarrow \ OKsbs(e_0, u, d) \ \wedge \ OKsbs(e_1, u, d) \ \wedge \ OKsbs(e_2, u, d)$$

Theorem 4.7

$$OKsbs(\lambda v{:}T.\ e, u, d) \ \wedge \ u \neq v \ \Rightarrow \ OKsbs(e, u, d)$$

In the last theorem, the condition $u \neq v$ is used in the proof and is in fact necessary, because in general $OKsbs(\lambda v{:}T.\ e, u, d) \not\Rightarrow OKsbs(e, u, d)$, as shown by the counter-example $u = v$, $e = \lambda w{:}T.\ v$, and $d = w$.

If we substitute u with d in e, we remove u from the free variables of e and add the free variables of d. This is actually true only if u is free in e (otherwise the substitution causes no change) and no variable is captured (otherwise the captured variables of d would not contribute to the free variables of the result of substitution):

Theorem 4.8

$$u \in \mathcal{FV}(e) \ \wedge \ OKsbs(e, u, d) \ \Rightarrow \ \mathcal{FV}(e[u/d]) = (\mathcal{FV}(e) - \{u\}) \cup \mathcal{FV}(d)$$

If we substitute u with an expression d that does not have u among its free variables, the resulting expression does not have u among its free variables (provided that the substitution captured no variables):

Theorem 4.9

$$OKsbs(e, u, d) \ \wedge \ u \notin \mathcal{FV}(d) \ \Rightarrow \ u \notin \mathcal{FV}(e[u/d])$$

Two expression substitutions commute if their variables do not "interact":

Theorem 4.10

$$u \neq u' \ \wedge \ u \notin \mathcal{FV}(d') \ \wedge \ u' \notin \mathcal{FV}(d) \ \Rightarrow \ e[u/d][u'/d'] = e[u'/d'][u/d]$$

If we substitute first u with d and then u with d', we obtain the same result as directly substituting u with $d[u/d']$:

Theorem 4.11

$$e[u/d][u/d'] = e[u/d[u/d']]$$

Variable renaming is idempotent:

Theorem 4.12

$$e[u/u'][u/u'] = e[u/u']$$

Substituting w with d (such that no capture takes place) and then renaming u to u' is equivalent to substituting w with the renaming applied to d (which renames the free occurrences of u in d) and then applying the renaming to the result (which renames the free occurrences of u in the original expression e):

Theorem 4.13

$$OKsbs(e, w, d) \;\Rightarrow\; e[w/d][u/u'] = e[w/d[u/u']][u/u']$$

In an expression e, renaming u to u' and then back u' to u leaves e unchanged, provided that u' is not captured (otherwise the captured occurrences would not be renamed back to u) and does not already occur free in e (otherwise we do not obtain e at the end, because all free occurrences of u' disappear when we rename u' to u):

Theorem 4.14

$$OKsbs(e, u, u') \;\wedge\; u' \notin \mathcal{FV}(e) \;\Rightarrow\; e[u/u'][u'/u] = e$$

If we rename a variable u to u' in an expression then the variables captured at the free occurrences of u in the original expression coincide with those captured at the free occurrences of u' in the transformed expression (provided that u' is not captured in the renaming and that u' was not already free in the original expression):

Theorem 4.15

$$OKsbs(e, u, u') \;\wedge\; u' \notin \mathcal{FV}(e) \;\Rightarrow\; \mathcal{CV}(e, u) = \mathcal{CV}(e[u/u'], u')$$

The following theorem says how captured variables change under expression substitution, assuming that the substituted variable is free in the expression (otherwise no change takes place) and that no variable is captured. The relationship should be quite intuitive when thinking of expressions as trees and captured variables as the lambda-bound variables encountered along the paths in the trees. The substitution replaces each free occurrence of u in e with d. Thus, if w is free in d we have to add the variables in the paths to free occurrences of w in d to those in the paths to free occurrences of u in e. If w is not free in d, there are no such paths. In addition, if $w \neq u$, we add the variables in the paths to the free occurrences of w in e; if instead $w = u$, only the free occurrences of w in d count.

Theorem 4.16

$$\begin{array}{l} u \in \mathcal{FV}(e) \;\wedge\; OKsbs(e, u, d) \;\Rightarrow \\ \mathcal{CV}(e[u/d], w) = \left\{\begin{array}{ll} \mathcal{CV}(e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{array}\right. \cup \left\{\begin{array}{ll} \mathcal{CV}(e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{array}\right. \end{array}$$

If we rename a variable u to u' in an expression then the variables captured at the free occurrences of a third variable w in the original expression coincide with those captured at the free occurrences of w in the transformed expression (provided that u' is not captured in the renaming):

Theorem 4.17

$$OKsbs(e, u, u') \;\wedge\; w \neq u \;\wedge\; w \neq u' \;\Rightarrow\; \mathcal{CV}(e, w) = \mathcal{CV}(e[u/u'], w)$$

Renaming bound variables does not change free variables, provided that the renaming causes no capture (not only the substitution $e[v/v']$ must cause no capture, but also v' must not occur free in e, otherwise it would be captured by the top-level $\lambda v' : T.\ \ldots$):

Theorem 4.18

$$v' \notin \mathcal{FV}(e) \cup \mathcal{CV}(e, v) \;\Rightarrow\; \mathcal{FV}(\lambda v{:}T.\ e) = \mathcal{FV}(\lambda v'{:}T.\ e[v/v'])$$

Type substitution does not change free and captured variables:

Theorem 4.19

$$\mathcal{FV}(e[\sigma]) = \mathcal{FV}(e)$$

Theorem 4.20

$$\mathcal{CV}(e[\sigma], u) = \mathcal{CV}(e, u)$$

If we substitute u with d in an e, we add the free type variables of d to those of e if u is free in e (otherwise we add no type variables):

Theorem 4.21

$$\mathcal{FTV}(e[u/d]) = \mathcal{FTV}(e) \cup \begin{cases} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(e) \\ \emptyset & \text{otherwise} \end{cases}$$

Variable renaming does not change free type variables:

Theorem 4.22

$$\mathcal{FTV}(e[u/u']) = \mathcal{FTV}(e)$$

Applying a type substitution σ to the result of a variable substitution $e[u/d]$ is like applying σ to e and d and then performing the variable substitution:

Theorem 4.23

$$e[u/d][\sigma] = e[\sigma][u/d[\sigma]]$$

Variable renaming commutes with type substitution:

Theorem 4.24

$$e[u/u'][\sigma] = e[\sigma][u/u']$$

Expression substitution does not change the type, op, and (type) variable declarations in a context, does not change the type definitions in a context, and applies to the axioms in a context:

Theorem 4.25

$$\begin{aligned}
\mathcal{TN}(cx[u/d]) &= \mathcal{TN}(cx) \\
\mathcal{ON}(cx[u/d]) &= \mathcal{ON}(cx) \\
\mathcal{TV}(cx[u/d]) &= \mathcal{TV}(cx) \\
\mathcal{V}(cx[u/d]) &= \mathcal{V}(cx) \\
\mathsf{ty}\ \tau{:}n \in cx &\Leftrightarrow \mathsf{ty}\ \tau{:}n \in cx[u/d] \\
\mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx &\Leftrightarrow \mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx[u/d] \\
\mathsf{ax}\ \{\overline{\beta}\}\ e \in cx &\Rightarrow \mathsf{ax}\ \{\overline{\beta}\}\ e[u/d] \in cx[u/d] \\
\mathsf{tvar}\ \beta \in cx &\Leftrightarrow \mathsf{tvar}\ \beta \in cx[u/d] \\
\mathsf{var}\ v{:}T \in cx &\Leftrightarrow \mathsf{var}\ v{:}T \in cx[u/d]
\end{aligned}$$

An empty type substitution causes no change:

Theorem 4.26

$$x \in \mathit{Type} + \mathit{Exp} + \mathit{CxElem} + \mathit{Cx} \quad \Rightarrow \quad x[\emptyset] = x$$

In a type substitution, only the type variables that are free in the type, expression, or context (element) matter (i.e. we can eliminate the other type variables from the domain of the substitution):

Theorem 4.27

$$x \in \mathit{Type} + \mathit{Exp} + \mathit{CxElem} + \mathit{Cx} \quad \Rightarrow \quad x[\sigma] = x[\sigma \downarrow_{\mathcal{FTV}(x)}]$$

If none of the type variables substituted by a type substitution appears in the type, expression, or context (element) to which the type substitution is applied, then the type, expression, or context (element) is unchanged:

Theorem 4.28

$$x \in \mathit{Type} + \mathit{Exp} + \mathit{CxElem} + \mathit{Cx} \quad \wedge \quad \mathcal{D}(\sigma) \cap \mathcal{FTV}(x) = \emptyset \quad \Rightarrow \quad x[\sigma] = x$$

If we apply a type substitution σ to a type or expression x, the free type variables that are substituted are replaced by the free type variables of the types that replace those type variables:

Theorem 4.29

$$x \in \mathit{Type} + \mathit{Exp} \quad \Rightarrow \quad \mathcal{FTV}(x[\sigma]) \quad = \quad (\mathcal{FTV}(x) - \mathcal{D}(\sigma)) \quad \cup \bigcup_{\alpha \in \mathcal{FTV}(x) \cap \mathcal{D}(\sigma)} \mathcal{FTV}(\sigma(\alpha))$$

Applying two type substitutions σ and σ' one after the other to a type or expression is like applying $\sigma[\sigma']$, provided that σ' does not substitute type variables that are free in the type or expression but are not in the domain of σ:

Theorem 4.30

$$x \in \mathit{Type} + \mathit{Exp} \quad \wedge \quad (\mathcal{FTV}(x) - \mathcal{D}(\sigma)) \cap \mathcal{D}(\sigma') = \emptyset \quad \Rightarrow \quad x[\sigma][\sigma'] = x[\sigma[\sigma']]$$

Applying a type substitution σ followed by another type substitution σ' is like applying σ' followed by $\sigma[\sigma']$, under certain conditions:

Theorem 4.31

$$\begin{array}{l} x \in \mathit{Type} + \mathit{Exp} \\ \mathcal{D}(\sigma) \cap \mathcal{D}(\sigma') = \emptyset \\ (\forall \alpha \in \mathcal{FTV}(x) \cap \mathcal{D}(\sigma').\ \ \mathcal{FTV}(\sigma'(\alpha)) \cap \mathcal{D}(\sigma) = \emptyset) \\ \quad \Rightarrow \\ x[\sigma][\sigma'] = x[\sigma'][\sigma[\sigma']] \end{array}$$

4.3 Properties of abbreviations

The following five theorems show that $\mathcal{FV}$, $\_[\_/\_]$, $\mathcal{CV}$, $\_[\_]$, and $\mathcal{FTV}$ "extend" to the abbreviations defined in §2 as expected.

Theorem 4.32

$$
\begin{array}{ll}
\mathcal{FV}(\mathsf{true}) & = \emptyset \\
\mathcal{FV}(\mathsf{false}) & = \emptyset \\
\mathcal{FV}(\neg) & = \emptyset \\
\mathcal{FV}(e_1 \wedge e_2) & = \mathcal{FV}(e_1) \cup \mathcal{FV}(e_2) \\
\mathcal{FV}(e_1 \vee e_2) & = \mathcal{FV}(e_1) \cup \mathcal{FV}(e_2) \\
\mathcal{FV}(e_1 \Rightarrow e_2) & = \mathcal{FV}(e_1) \cup \mathcal{FV}(e_2) \\
\mathcal{FV}(\Leftrightarrow) & = \emptyset \\
\mathcal{FV}(e_1 \Leftrightarrow e_2) & = \mathcal{FV}(e_1) \cup \mathcal{FV}(e_2) \\
\mathcal{FV}(e_1 \not\equiv e_2) & = \mathcal{FV}(e_1) \cup \mathcal{FV}(e_2) \\
\mathcal{FV}(\iota v{:}T.e) & = \mathcal{FV}(e) - \{v\} \\
\mathcal{FV}(\forall_T) & = \emptyset \\
\mathcal{FV}(\forall v{:}T.\ e) & = \mathcal{FV}(e) - \{v\} \\
\mathcal{FV}(\forall v_1{:}T_1, \ldots, v_n{:}T_n.\ e) & = \mathcal{FV}(e) - \overline{v} \\
\mathcal{FV}(\forall \overline{v}{:}\overline{T}.\ e) & = \mathcal{FV}(e) - \overline{v} \\
\mathcal{FV}(\exists_T) & = \emptyset \\
\mathcal{FV}(\exists v{:}T.\ e) & = \mathcal{FV}(e) - \{v\} \\
\mathcal{FV}(\exists v_1{:}T_1, \ldots, v_n{:}T_n.\ e) & = \mathcal{FV}(e) - \overline{v} \\
\mathcal{FV}(\exists \overline{v}{:}\overline{T}.\ e) & = \mathcal{FV}(e) - \overline{v} \\
\mathcal{FV}(\exists!_T) & = \emptyset \\
\mathcal{FV}(\exists!\, v{:}T.\ e) & = \mathcal{FV}(e) - \{v\} \\
\mathcal{FV}(e.f) & = \mathcal{FV}(e)
\end{array}
$$

Theorem 4.33

$$
\begin{array}{rl}
\mathsf{true}[u/d] & = \mathsf{true} \\
\mathsf{false}[u/d] & = \mathsf{false} \\
\neg[u/d] & = \neg \\
(e_1 \wedge e_2)[u/d] & = e_1[u/d] \wedge e_2[u/d] \\
(e_1 \vee e_2)[u/d] & = e_1[u/d] \vee e_2[u/d] \\
(e_1 \Rightarrow e_2)[u/d] & = e_1[u/d] \Rightarrow e_2[u/d] \\
\Leftrightarrow [u/d] & = \Leftrightarrow \\
(e_1 \Leftrightarrow e_2)[u/d] & = e_1[u/d] \Leftrightarrow e_2[u/d] \\
(e_1 \not\equiv e_2)[u/d] & = e_1[u/d] \not\equiv e_2[u/d] \\
(\iota v{:}T.e)[u/d] & = \begin{cases} \iota v{:}T.e & \textit{if} \quad u = v \\ \iota v{:}T.e[u/d] & \textit{otherwise} \end{cases} \\
\forall_T[u/d] & = \forall_T \\
(\forall v{:}T.\ e)[u/d] & = \begin{cases} \forall v{:}T.\ e & \textit{if} \quad u = v \\ \forall v{:}T.\ e[u/d] & \textit{otherwise} \end{cases} \\
(\forall v_1{:}T_1, \ldots, v_n{:}T_n.\ e)[u/d] & = \begin{cases} \forall v_1{:}T_1, \ldots, v_n{:}T_n.\ e & \textit{if} \quad u \in \overline{v} \\ \forall v_1{:}T_1, \ldots, v_n{:}T_n.\ e[u/d] & \textit{otherwise} \end{cases} \\
(\forall \overline{v}{:}\overline{T}.\ e)[u/d] & = \begin{cases} \forall \overline{v}{:}\overline{T}.\ e & \textit{if} \quad u \in \overline{v} \\ \forall \overline{v}{:}\overline{T}.\ e[u/d] & \textit{otherwise} \end{cases} \\
\exists_T[u/d] & = \exists_T \\
(\exists v{:}T.\ e)[u/d] & = \begin{cases} \exists v{:}T.\ e & \textit{if} \quad u = v \\ \exists v{:}T.\ e[u/d] & \textit{otherwise} \end{cases} \\
(\exists v_1{:}T_1, \ldots, v_n{:}T_n.\ e)[u/d] & = \begin{cases} \exists v_1{:}T_1, \ldots, v_n{:}T_n.\ e & \textit{if} \quad u \in \overline{v} \\ \exists v_1{:}T_1, \ldots, v_n{:}T_n.\ e[u/d] & \textit{otherwise} \end{cases} \\
(\exists \overline{v}{:}\overline{T}.\ e)[u/d] & = \begin{cases} \exists \overline{v}{:}\overline{T}.\ e & \textit{if} \quad u \in \overline{v} \\ \exists \overline{v}{:}\overline{T}.\ e[u/d] & \textit{otherwise} \end{cases} \\
\exists!_T[u/d] & = \exists!_T \\
(\exists!\, v{:}T.\ e)[u/d] & = \begin{cases} \exists!\, v{:}T.\ e & \textit{if} \quad u = v \\ \exists!\, v{:}T.\ e[u/d] & \textit{otherwise} \end{cases} \\
(e.f)[u/d] & = e[u/d].f
\end{array}
$$

Theorem 4.34

$$
\begin{array}{ll}
\mathcal{CV}(\mathsf{true}, u) & = \emptyset \\
\mathcal{CV}(\mathsf{false}, u) & = \emptyset \\
\mathcal{CV}(\neg, u) & = \emptyset \\
\mathcal{CV}(e_1 \wedge e_2, u) & = \mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u) \\
\mathcal{CV}(e_1 \vee e_2, u) & = \mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u) \\
\mathcal{CV}(e_1 \Rightarrow e_2, u) & = \mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u) \\
\mathcal{CV}(\Leftrightarrow, u) & = \emptyset \\
\mathcal{CV}(e_1 \Leftrightarrow e_2, u) & = \mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u) \\
\mathcal{CV}(e_1 \not\equiv e_2, u) & = \mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u) \\
\mathcal{CV}(\iota v\!:\!T.e, u) & = \begin{cases} \{v\} \cup \mathcal{CV}(e, u) & \textit{if} \quad u \in \mathcal{FV}(e) - \{v\} \\ \emptyset & \textit{otherwise} \end{cases} \\
\mathcal{CV}(\forall_T, u) & = \emptyset \\
\mathcal{CV}(\forall v\!:\!T.\ e, u) & = \begin{cases} \{v\} \cup \mathcal{CV}(e, u) & \textit{if} \quad u \in \mathcal{FV}(e) - \{v\} \\ \emptyset & \textit{otherwise} \end{cases} \\
\mathcal{CV}(\forall v_1\!:\!T_1, \ldots, v_n\!:\!T_n.\ e, u) & = \begin{cases} \overline{v} \cup \mathcal{CV}(e, u) & \textit{if} \quad u \in \mathcal{FV}(e) - \overline{v} \\ \emptyset & \textit{otherwise} \end{cases} \\
\mathcal{CV}(\forall \overline{v}\!:\!\overline{T}.\ e, u) & = \begin{cases} \overline{v} \cup \mathcal{CV}(e, u) & \textit{if} \quad u \in \mathcal{FV}(e) - \overline{v} \\ \emptyset & \textit{otherwise} \end{cases} \\
\mathcal{CV}(\exists_T, u) & = \emptyset \\
\mathcal{CV}(\exists v\!:\!T.\ e, u) & = \begin{cases} \{v\} \cup \mathcal{CV}(e, u) & \textit{if} \quad u \in \mathcal{FV}(e) - \{v\} \\ \emptyset & \textit{otherwise} \end{cases} \\
\mathcal{CV}(\exists v_1\!:\!T_1, \ldots, v_n\!:\!T_n.\ e, u) & = \begin{cases} \overline{v} \cup \mathcal{CV}(e, u) & \textit{if} \quad u \in \mathcal{FV}(e) - \overline{v} \\ \emptyset & \textit{otherwise} \end{cases} \\
\mathcal{CV}(\exists \overline{v}\!:\!\overline{T}.\ e, u) & = \begin{cases} \overline{v} \cup \mathcal{CV}(e, u) & \textit{if} \quad u \in \mathcal{FV}(e) - \overline{v} \\ \emptyset & \textit{otherwise} \end{cases} \\
\mathcal{CV}(\exists!_T, u) & = \emptyset \\
\mathcal{CV}(\exists!\, v\!:\!T.\ e, u) & = \begin{cases} \{v\} \cup \mathcal{CV}(e, u) & \textit{if} \quad u \in \mathcal{FV}(e) - \{v\} \\ \emptyset & \textit{otherwise} \end{cases} \\
\mathcal{CV}(e.f, u) & = \mathcal{CV}(e, u)
\end{array}
$$

Theorem 4.35

$$
\begin{array}{rl}
\mathsf{true}[\sigma] & = \mathsf{true} \\
\mathsf{false}[\sigma] & = \mathsf{false} \\
\neg[\sigma] & = \neg \\
(e_1 \wedge e_2)[\sigma] & = e_1[\sigma] \wedge e_2[\sigma] \\
(e_1 \vee e_2)[\sigma] & = e_1[\sigma] \vee e_2[\sigma] \\
(e_1 \Rightarrow e_2)[\sigma] & = e_1[\sigma] \Rightarrow e_2[\sigma] \\
\Leftrightarrow [\sigma] & = \emptyset \\
(e_1 \Leftrightarrow e_2)[\sigma] & = e_1[\sigma] \Leftrightarrow e_2[\sigma] \\
(e_1 \not\equiv e_2)[\sigma] & = e_1[\sigma] \not\equiv e_2[\sigma] \\
(\iota v\!:\!T.e)[\sigma] & = \iota v\!:\!T[\sigma].e[\sigma] \\
\forall_T[\sigma] & = \forall_{T[\sigma]} \\
(\forall v\!:\!T.\ e)[\sigma] & = \forall v\!:\!T[\sigma].\ e[\sigma] \\
(\forall v_1\!:\!T_1, \ldots, v_n\!:\!T_n.\ e)[\sigma] & = \forall v_1\!:\!T_1[\sigma], \ldots, v_n\!:\!T_n[\sigma].\ e[\sigma] \\
(\forall \overline{v}\!:\!\overline{T}.\ e)[\sigma] & = \forall \overline{v}\!:\!\overline{T}[\sigma].\ e[\sigma] \\
\exists_T[\sigma] & = \exists_{T[\sigma]} \\
(\exists v\!:\!T.\ e)[\sigma] & = \exists v\!:\!T[\sigma].\ e[\sigma] \\
(\exists v_1\!:\!T_1, \ldots, v_n\!:\!T_n.\ e)[\sigma] & = \exists v_1\!:\!T_1[\sigma], \ldots, v_n\!:\!T_n[\sigma].\ e[\sigma] \\
(\exists \overline{v}\!:\!\overline{T}.\ e)[\sigma] & = \exists \overline{v}\!:\!\overline{T}[\sigma].\ e[\sigma] \\
\exists!_T[\sigma] & = \exists!_{T[\sigma]} \\
(\exists!\, v\!:\!T.\ e)[\sigma] & = \exists!\, v\!:\!T[\sigma].\ e[\sigma] \\
(e.f)[\sigma] & = e[\sigma].f
\end{array}
$$

Theorem 4.36

$$
\begin{array}{ll}
\mathcal{FTV}(\mathsf{true}) & = \emptyset \\
\mathcal{FTV}(\mathsf{false}) & = \emptyset \\
\mathcal{FTV}(\neg) & = \emptyset \\
\mathcal{FTV}(e_1 \wedge e_2) & = \mathcal{FTV}(e_1) \cup \mathcal{FTV}(e_2) \\
\mathcal{FTV}(e_1 \vee e_2) & = \mathcal{FTV}(e_1) \cup \mathcal{FTV}(e_2) \\
\mathcal{FTV}(e_1 \Rightarrow e_2) & = \mathcal{FTV}(e_1) \cup \mathcal{FTV}(e_2) \\
\mathcal{FTV}(\Leftrightarrow) & = \emptyset \\
\mathcal{FTV}(e_1 \Leftrightarrow e_2) & = \mathcal{FTV}(e_1) \cup \mathcal{FTV}(e_2) \\
\mathcal{FTV}(e_1 \not\equiv e_2) & = \mathcal{FTV}(e_1) \cup \mathcal{FTV}(e_2) \\
\mathcal{FTV}(\iota v{:}T.e) & = \mathcal{FTV}(T) \cup \mathcal{FTV}(e) \\
\mathcal{FTV}(\forall_T) & = \mathcal{FTV}(T) \\
\mathcal{FTV}(\forall v{:}T.\ e) & = \mathcal{FTV}(T) \cup \mathcal{FTV}(e) \\
\mathcal{FTV}(\forall v_1{:}T_1, \ldots, v_n{:}T_n.\ e) & = \mathcal{FTV}(e) \cup \bigcup_i \mathcal{FTV}(T_i) \\
\mathcal{FTV}(\forall \overline{v}{:}\overline{T}.\ e) & = \mathcal{FTV}(e) \cup \bigcup_i \mathcal{FTV}(T_i) \\
\mathcal{FTV}(\exists_T) & = \mathcal{FTV}(T) \\
\mathcal{FTV}(\exists v{:}T.\ e) & = \mathcal{FTV}(T) \cup \mathcal{FTV}(e) \\
\mathcal{FTV}(\exists v_1{:}T_1, \ldots, v_n{:}T_n.\ e) & = \mathcal{FTV}(e) \cup \bigcup_i \mathcal{FTV}(T_i) \\
\mathcal{FTV}(\exists \overline{v}{:}\overline{T}.\ e) & = \mathcal{FTV}(e) \cup \bigcup_i \mathcal{FTV}(T_i) \\
\mathcal{FTV}(\exists!_T) & = \mathcal{FTV}(T) \\
\mathcal{FTV}(\exists!\, v{:}T.\ e) & = \mathcal{FTV}(T) \cup \mathcal{FTV}(e) \\
\mathcal{FTV}(e.f) & = \mathcal{FTV}(e) \cup \bigcup_i \mathcal{FTV}(T_i)
\end{array}
$$

4.4 Proof-theoretical properties

Any prefix of a well-formed context is itself well-formed and if we derive a judgement with some context, (any prefix of) that context is well-formed:

Theorem 4.37

$$
\begin{array}{lcl}
\vdash cx_1, cx_2 : \textsc{context} & \Rightarrow & \vdash cx_1 : \textsc{context} \\
cx_1, cx_2 \vdash \ldots & \Rightarrow & \vdash cx_1 : \textsc{context}
\end{array}
$$

The following four theorems say that well-formed contexts have no duplicate types, ops, and (type) variables:

Theorem 4.38

$$\vdash cx_1, \mathsf{ty}\ \tau{:}n, cx_2 : \textsc{context} \ \Rightarrow\ \tau \notin \mathcal{TN}(cx_1, cx_2)$$

Theorem 4.39

$$\vdash cx_1, \mathsf{op}\ o{:}\{\overline{\beta}\}\ T, cx_2 : \textsc{context} \ \Rightarrow\ o \notin \mathcal{ON}(cx_1, cx_2)$$

Theorem 4.40

$$\vdash cx_1, \mathsf{tvar}\ \beta, cx_2 : \textsc{context} \ \Rightarrow\ \beta \notin \mathcal{TV}(cx_1, cx_2)$$

Theorem 4.41

$$\vdash cx_1, \mathsf{var}\ v{:}T, cx_2 : \textsc{context} \ \Rightarrow\ v \notin \mathcal{V}(cx_1, cx_2)$$

The type of an op declared in a well-formed context is well-formed in the prefix of the context that precedes the op, extended with the type variables over which the op is polymorphic:

Theorem 4.42

$$\vdash cx_1, \mathsf{op}\ o{:}\{\overline{\beta}\}\ T, cx_2 : \textsc{context} \quad \Rightarrow \quad cx_1, \mathsf{tvar}\ \overline{\beta} \vdash T : \textsc{type}$$

An axiom declared in a well-formed context is well-typed (with type Bool) in the prefix of the context that precedes the axiom, extended with the type variables over which the axiom is polymorphic:

Theorem 4.43

$$\vdash cx_1, \mathsf{ax}\ \{\overline{\beta}\}\ e, cx_2 : \textsc{context} \quad \Rightarrow \quad cx_1, \mathsf{tvar}\ \overline{\beta} \vdash e : \mathsf{Bool}$$

The type of a variable declared in a well-formed context is well-formed in the prefix of the context that precedes the variable declaration:

Theorem 4.44

$$\vdash cx_1, \mathsf{var}\ v{:}T, cx_2 : \textsc{context} \quad \Rightarrow \quad cx_1 \vdash T : \textsc{type}$$

A well-formed type variable is declared in the context:

Theorem 4.45

$$cx \vdash \beta : \textsc{type} \quad \Rightarrow \quad \beta \in \mathcal{TV}(cx)$$

The type name of a well-formed type instance is declared in the context, with an arity that matches the type arguments, and each type argument is well-formed:

Theorem 4.46

$$cx \vdash \tau[\overline{T}] : \textsc{type} \quad \Rightarrow \quad \mathsf{ty}\ \tau{:}|\overline{T}| \in cx \quad \wedge \quad (\forall i.\ \ cx \vdash T_i : \textsc{type})$$

The domain and range types of a well-formed arrow type are well-formed (for the range, the context is extended with a variable of the domain type):

Theorem 4.47

$$cx \vdash T_1 \to T_2 : \textsc{type} \quad \Rightarrow \quad cx \vdash T_1 : \textsc{type} \quad \wedge \quad (\exists v.\ \ cx, \mathsf{var}\ v{:}T_1 \vdash T_2 : \textsc{type})$$

The component types of a well-formed record type are well-formed:

Theorem 4.48

$$cx \vdash \prod_i f_i\ T_i : \textsc{type} \quad \Rightarrow \quad (\forall i.\ \ cx \vdash T_i : \textsc{type})$$

The predicate of a restriction type is well-typed, with the expected type:

Theorem 4.49

$$cx \vdash T|r : \textsc{type} \quad \Rightarrow \quad cx \vdash r : T \to \mathsf{Bool}$$

The predicate of a well-formed restriction type has no free variables:

Theorem 4.50

$$cx \vdash T|r : \textsc{type} \quad \Rightarrow \quad \mathcal{FV}(r) = \emptyset$$

For each well-typed application, the first expression (function) has an arrow type and the second expression (argument) has the domain type:

Theorem 4.51

$$cx \vdash e_1\ e_2 : \ldots \quad \Rightarrow \quad \exists T_1, T_2.\ \ cx \vdash e_1 : T_1 \to T_2 \quad \wedge \quad cx \vdash e_2 : T_1$$

A well-typed abstraction has an arrow type whose domain is the type of its bound variable:

Theorem 4.52

$$cx \vdash \lambda v{:}T.\ e : \ldots \quad \Rightarrow \quad \exists T'.\ \ cx \vdash \lambda v{:}T.\ e : T \to T'$$

The subtype predicates that appear in (provable) subtype judgements have no free variables:

Theorem 4.53

$$cx \vdash T_1 \prec_r T_2 \ \Rightarrow\ \mathcal{FV}(r) = \emptyset$$

All the free variables in a well-typed expression or a theorem are declared in the context, and all the free variables in an axiom of a well-formed context are declared before the axiom:

Theorem 4.54

$$cx \vdash e : T \quad \Rightarrow \quad \mathcal{FV}(e) \subseteq \mathcal{V}(cx)$$

Theorem 4.55

$$\vdash cx_1, \mathsf{ax}\ \{\overline{\beta}\}\ e, cx_2 : \textsc{context} \quad \Rightarrow \quad \mathcal{FV}(e) \subseteq \mathcal{V}(cx_1)$$

Theorem 4.56

$$cx \vdash e \quad \Rightarrow \quad \mathcal{FV}(e) \subseteq \mathcal{V}(cx)$$

All the free type variables in the types and expressions to the right of a provable judgement $cx \vdash \ldots$ are declared in cx, and all the free type variables in a context element of a well-formed context are declared before the context element:

Theorem 4.57

$$\begin{array}{lcl}
cx \vdash T : \textsc{type} & \Rightarrow & \mathcal{TV}(cx) \supseteq \mathcal{FTV}(T) \\
cx \vdash T_1 \prec_r T_2 & \Rightarrow & \mathcal{TV}(cx) \supseteq \mathcal{FTV}(T_1) \cup \mathcal{FTV}(r) \cup \mathcal{FTV}(T_2) \\
cx \vdash e : T & \Rightarrow & \mathcal{TV}(cx) \supseteq \mathcal{FTV}(e) \cup \mathcal{FTV}(T) \\
cx \vdash e & \Rightarrow & \mathcal{TV}(cx) \supseteq \mathcal{FTV}(e) \\
\vdash cx_1, cxel, cx_2 : \textsc{context} & \Rightarrow & \mathcal{TV}(cx_1) \supseteq \mathcal{FTV}(cxel)
\end{array}$$

The derivation of a judgement does not depend on the choice of the names for the variables in the derivation. So, if we replace a variable declared in the context of the judgement with a fresh one[6] and perform the substitution in the rest of the judgement accordingly, the judgement is still provable:

Theorem 4.58

$$\begin{array}{lcl}
u' \textit{ fresh} \ \wedge\ \vdash cx_1, \mathsf{var}\ u{:}S, cx_2 : \textsc{context} & \Rightarrow & \vdash cx_1, \mathsf{var}\ u'{:}S, cx_2[u/u'] : \textsc{context} \\
u' \textit{ fresh} \ \wedge\ cx_1, \mathsf{var}\ u{:}S, cx_2 \vdash T : \textsc{type} & \Rightarrow & cx_1, \mathsf{var}\ u'{:}S, cx_2[u/u'] \vdash T : \textsc{type} \\
u' \textit{ fresh} \ \wedge\ cx_1, \mathsf{var}\ u{:}S, cx_2 \vdash T_1 \prec_r T_2 & \Rightarrow & cx_1, \mathsf{var}\ u'{:}S, cx_2[u/u'] \vdash T_1 \prec_r T_2 \\
u' \textit{ fresh} \ \wedge\ cx_1, \mathsf{var}\ u{:}S, cx_2 \vdash e : T & \Rightarrow & cx_1, \mathsf{var}\ u'{:}S, cx_2[u/u'] \vdash e[u/u'] : T \\
u' \textit{ fresh} \ \wedge\ cx_1, \mathsf{var}\ u{:}S, cx_2 \vdash e & \Rightarrow & cx_1, \mathsf{var}\ u'{:}S, cx_2[u/u'] \vdash e[u/u']
\end{array}$$

The Metaslang logic is monotonic, in the sense that anything can be derived in a bigger context that can be derived in a smaller context:

Theorem 4.59

$$\vdash cx' : \textsc{context} \quad \wedge \quad cx \subseteq cx' \quad \Rightarrow \quad \left(\begin{array}{lcl}
cx \vdash T : \textsc{type} & \Rightarrow & cx' \vdash T : \textsc{type} \\
cx \vdash T_1 \prec_r T_2 & \Rightarrow & cx' \vdash T_1 \prec_r T_2 \\
cx \vdash e : T & \Rightarrow & cx' \vdash e : T \\
cx \vdash e & \Rightarrow & cx' \vdash e
\end{array} \right)$$

[6]The adjective "fresh" means that the variable does not occur anywhere in (the judgements that comprise) the derivation that is under discussion. This notion could be made more formal, but it should be sufficiently clear as stated.

Note that the monotonicity theorem above also applies to the case in which cx' is a (well-formed) permutation of cx (it is still the case that $cx \subseteq cx'$). In other words, anything that can be derived in a context can be also derived in any well-formed permutation of that context.

The following theorem says that if we derive a judgement $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash \ldots$, then we can substitute the $\overline{\alpha}$ with well-formed types $\overline{S}$, under certain conditions:

Theorem 4.60

$$\begin{array}{l} \forall i.\quad cx_1, cx_2 \vdash S_i : \textsc{type} \\ \mathit{OKsbs}(cx_3, \overline{\alpha}, \overline{S}) \\ \vdash cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] : \textsc{context} \\ \quad \Rightarrow \\ \left(\begin{array}{lcl} cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash T : \textsc{type} & \Rightarrow & cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash T[\overline{\alpha}/\overline{S}] : \textsc{type} \\ cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash T_1 \prec_r T_2 & \Rightarrow & cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash T_1[\overline{\alpha}/\overline{S}] \prec_{r[\overline{\alpha}/\overline{S}]} T_2[\overline{\alpha}/\overline{S}] \\ cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash e : T & \Rightarrow & cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash e[\overline{\alpha}/\overline{S}] : T[\overline{\alpha}/\overline{S}] \\ cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash e & \Rightarrow & cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash e[\overline{\alpha}/\overline{S}] \end{array} \right) \end{array}$$

The following theorems prove derived well-formedness rules for contexts:

Theorem 4.61

$$\frac{\begin{array}{c} cx \vdash T : \textsc{type} \\ \vdash cx, cx' : \textsc{context} \\ v \notin \mathcal{V}(cx, cx') \end{array}}{\vdash cx, cx', \mathsf{var}\ v{:}T : \textsc{context}} \quad (\textsc{cxVdec'})$$

Theorem 4.62

$$\frac{\begin{array}{c} \vdash cx : \textsc{context} \\ v \notin \mathcal{V}(cx) \end{array}}{\vdash cx, \mathsf{var}\ v{:}\mathsf{Bool} : \textsc{context}} \quad (\textsc{cxVdecBool})$$

Theorem 4.63

$$\frac{cx \vdash e : \mathsf{Bool}}{\vdash cx, \mathsf{ax}\ e : \textsc{context}} \quad (\textsc{cxAx0})$$

The following theorem proves a derived well-formedness rule for arrow types:

Theorem 4.64

$$\frac{\begin{array}{c} cx \vdash T_1 : \textsc{type} \\ cx \vdash T_2 : \textsc{type} \end{array}}{cx \vdash T_1 \to T_2 : \textsc{type}} \quad (\textsc{tyArr'})$$

The following theorems prove derived well-typedness rules for expressions, including the abbreviations defined in §2:

Theorem 4.65

$$\frac{cx \vdash T : \textsc{type}}{cx \vdash \lambda v{:}T.\ v : T \to T} \quad (\textsc{exId})$$

Theorem 4.66

$$\frac{\vdash cx : \textsc{context}}{cx \vdash \lambda v{:}\mathsf{Bool}.\ v : \mathsf{Bool} \to \mathsf{Bool}} \quad (\textsc{exIdBool})$$

Theorem 4.67

$$\frac{\vdash cx : \textsc{context}}{cx \vdash \mathsf{true} : \mathsf{Bool}} \quad (\textsc{exTrue})$$

Theorem 4.68

$$\frac{cx \vdash T : \textsc{type}}{cx \vdash \lambda v{:}T.\ \mathsf{true} : T \to \mathsf{Bool}} \quad (\textsc{exConstTrue})$$

Theorem 4.69

$$\frac{\vdash cx : \textsc{context}}{cx \vdash \mathsf{false} : \mathsf{Bool}} \quad (\textsc{exFalse})$$

Theorem 4.70

$$\frac{\vdash cx : \textsc{context}}{cx \vdash \neg : \mathsf{Bool} \to \mathsf{Bool}} \quad (\textsc{exNot})$$

Theorem 4.71

$$\frac{cx \vdash e : \mathsf{Bool}}{cx \vdash \neg\ e : \mathsf{Bool}} \quad (\textsc{exNeg})$$

Theorem 4.72

$$\frac{\begin{array}{c} cx \vdash e_1 : \mathsf{Bool} \\ cx, \mathsf{ax}\ e_1 \vdash e_2 : \mathsf{Bool} \end{array}}{cx \vdash e_1 \wedge e_2 : \mathsf{Bool}} \quad (\textsc{exConj})$$

Theorem 4.73

$$\frac{\begin{array}{c} cx \vdash e_1 : \mathsf{Bool} \\ cx \vdash e_2 : \mathsf{Bool} \end{array}}{cx \vdash e_1 \wedge e_2 : \mathsf{Bool}} \quad (\textsc{exConj0})$$

Theorem 4.74

$$\frac{\begin{array}{c} cx \vdash e_1 : \mathsf{Bool} \\ cx, \mathsf{ax}\ \neg\ e_1 \vdash e_2 : \mathsf{Bool} \end{array}}{cx \vdash e_1 \vee e_2 : \mathsf{Bool}} \quad (\textsc{exDisj})$$

Theorem 4.75

$$\frac{\begin{array}{c} cx \vdash e_1 : \mathsf{Bool} \\ cx \vdash e_2 : \mathsf{Bool} \end{array}}{cx \vdash e_1 \vee e_2 : \mathsf{Bool}} \quad (\textsc{exDisj0})$$

Theorem 4.76

$$\frac{\begin{array}{c} cx \vdash e_1 : \mathsf{Bool} \\ cx, \mathsf{ax}\ e_1 \vdash e_2 : \mathsf{Bool} \end{array}}{cx \vdash e_1 \Rightarrow e_2 : \mathsf{Bool}} \quad (\textsc{exImpl})$$

Theorem 4.77

$$\frac{\begin{array}{c} cx \vdash e_1 : \mathsf{Bool} \\ cx \vdash e_2 : \mathsf{Bool} \end{array}}{cx \vdash e_1 \Rightarrow e_2 : \mathsf{Bool}} \quad (\textsc{exImpl0})$$

Theorem 4.78

$$\frac{\vdash cx : \textsc{context}}{cx \vdash \Leftrightarrow\!: \mathsf{Bool} \to \mathsf{Bool} \to \mathsf{Bool}} \quad (\textsc{exIff})$$

Theorem 4.79

$$\frac{cx \vdash e_1 : \mathsf{Bool} \quad cx \vdash e_2 : \mathsf{Bool}}{cx \vdash e_1 \Leftrightarrow e_2 : \mathsf{Bool}} \quad (\textsc{exCoImpl})$$

Theorem 4.80

$$\frac{cx \vdash e_1 : T \quad cx \vdash e_2 : T}{cx \vdash e_1 \not\equiv e_2 : \mathsf{Bool}} \quad (\textsc{exNeq})$$

Theorem 4.81

$$\frac{cx \vdash T : \textsc{type}}{cx \vdash \forall_T : (T \to \mathsf{Bool}) \to \mathsf{Bool}} \quad (\textsc{exFA})$$

Theorem 4.82

$$\frac{cx, \mathsf{var}\ v{:}T \vdash e : \mathsf{Bool}}{cx \vdash \forall v{:}T.\ e : \mathsf{Bool}} \quad (\textsc{exForAll})$$

Theorem 4.83

$$\frac{cx \vdash T : \textsc{type}}{cx \vdash \exists_T : (T \to \mathsf{Bool}) \to \mathsf{Bool}} \quad (\textsc{exEX})$$

Theorem 4.84

$$\frac{cx, \mathsf{var}\ v{:}T \vdash e : \mathsf{Bool}}{cx \vdash \exists v{:}T.\ e : \mathsf{Bool}} \quad (\textsc{exExists})$$

Theorem 4.85

$$\frac{cx \vdash T : \textsc{type}}{cx \vdash \exists!_T : (T \to \mathsf{Bool}) \to \mathsf{Bool}} \quad (\textsc{exEX1})$$

Theorem 4.86

$$\frac{cx, \mathsf{var}\ v{:}T \vdash e : \mathsf{Bool}}{cx \vdash \exists!\, v{:}T.\ e : \mathsf{Bool}} \quad (\textsc{exExists1})$$

Theorem 4.87

$$\frac{cx, \mathsf{var}\ v{:}T \vdash e : \mathsf{Bool} \quad v' \notin \mathcal{FV}(e) \cup \mathcal{CV}(e, v)}{cx \vdash \forall v'{:}T.\ e[v/v'] : \mathsf{Bool}} \quad (\textsc{exForAllAlpha})$$

Theorem 4.88

$$\frac{cx \vdash \prod_i f_i\ T_i : \textsc{type} \quad cx \vdash e : \prod_i f_i\ T_i}{cx \vdash e.f_j : T_j} \quad (\textsc{exDotProj0})$$

If a type is a subtype of another one, both types are well-formed; in addition, the predicate that defines the subtype is well-typed and is indeed a predicate over the supertype:

Theorem 4.89

$$cx \vdash T_1 \prec_r T_2 \quad \Rightarrow \quad cx \vdash T_1 : \textsc{type} \quad \wedge \quad cx \vdash T_2 : \textsc{type} \quad \wedge \quad cx \vdash r : T_2 \to \mathsf{Bool}$$

The type of a well-typed expression is well-formed:

Theorem 4.90

$$cx \vdash e : T \quad \Rightarrow \quad cx \vdash T : \textsc{type}$$

Using the previous theorem, we can eliminate the first premise from rule exDotProj0 (however, note that exDotProj0 is used in the proof of Theorem 4.89, which is proved by induction together with Theorem 4.90):

Theorem 4.91

$$\frac{cx \vdash e : \prod_i f_i\ T_i}{cx \vdash e.f_j : T_j} \quad (\textsc{exDotProj})$$

The expression true is a theorem:

Theorem 4.92

$$\frac{\vdash cx : \textsc{context}}{cx \vdash \mathsf{true}} \quad (\textsc{thTrue})$$

Proving that an expression equals true implies that the expression is a theorem:

Theorem 4.93

$$\frac{cx \vdash e \equiv \mathsf{true}}{cx \vdash e} \quad (\textsc{thEqTrue})$$

A universally quantified theorem can be instantiated with any well-typed expression whose type matches the universally quantified variable's and whose substitution does not cause capture:

Theorem 4.94

$$\frac{\begin{array}{c} cx \vdash \forall v{:}T.\ e : \mathsf{Bool} \\ cx \vdash \forall v{:}T.\ e \\ cx \vdash e' : T \\ OKsbs(e, v, e') \end{array}}{cx \vdash e[v/e']} \quad (\textsc{thForAllElim})$$

5 Models

[[[TO DO]]]

This section defines the notion of model of a context.

A model of a context is a mapping from names declared in the context to suitable set-theoretic entities. For instance, a type name τ of arity n is mapped to an n-ary function over sets (if $n = 0$, the model maps the type name simply to a set). The mapping is extended to all well-formed types, which are mapped to sets, and to all well-typed expressions, which are mapped to elements of the sets that their types map to. The model must satisfy all the axioms of the context.

It should be possible to prove the soundness of the rules to derive judgements with respect to models.

Since higher-order logic is notoriously incomplete, it is not possible to prove completeness of the rules to derive assertions. However, it should be possible to prove completeness with respect to general (a.k.a. Henkin) models. A general model is one in which the type $T_1 \to T_2$ is a subset of all functions from T_1 to T_2, and not necessarily the set of all such functions (as in standard models). Since there are more general models than standard models (a standard model is also a general model but not all general models are standard models), fewer formulas are true in all general models than in all standard models.

Perhaps this section should also contain a proof of the consistency of the Metaslang logic, analogously to the proof of the consistency of the higher-order logic defined in [3].

A Proofs

Proof of Theorem 4.1

Since

$$\begin{array}{l}\mathcal{CV}(e,u) = \emptyset \\ \quad \Rightarrow \\ \mathcal{FV}(d) \cap \mathcal{CV}(e,u) = \emptyset \\ \quad \Rightarrow \quad \text{[definition of } OKsbs\text{]} \\ OKsbs(e,u,d)\end{array}$$

we are left to prove

$$u \notin \mathcal{FV}(e) \;\Rightarrow\; \mathcal{CV}(e,u) = \emptyset \;\wedge\; e[u/d] = e$$

which we do by induction on e:

$o[\overline{T}]$, ι_T, $\mathsf{proj}\ f$)

$\mathcal{CV}(e,u) = \emptyset$ [definition of $\mathcal{CV}$]

$e[u/d] = e$ [definition of $\_[\_/\_]$]

v)

1. $\mathcal{CV}(v,u) = \emptyset$ [definition of $\mathcal{CV}$]
2. $u \notin \mathcal{FV}(v)$ [hypothesis]
3. $u \neq v$ [2, $\mathcal{FV}(v) = \{v\}$]
4. $v[u/d] = v$ [definition of $\_[\_/\_]$, 3]

$e_1\ e_2$)

1. $u \notin \mathcal{FV}(e_1\ e_2)$ [hypothesis]
2. $u \notin \mathcal{FV}(e_1) \;\wedge\; u \notin \mathcal{FV}(e_2)$ [1, $\mathcal{FV}(e_1\ e_2) = \mathcal{FV}(e_1) \cup \mathcal{FV}(e_2)$]
3. $\mathcal{CV}(e_1\ e_2, u)$
   $=$ [definition of $\mathcal{CV}$]
   $\mathcal{CV}(e_1,u) \cup \mathcal{CV}(e_2,u)$
   $=$ [induction hypothesis, 2]
   $\emptyset$
4. $(e_1\ e_2)[u/d]$
   $=$ [definition of $\_[\_/\_]$]
   $e_1[u/d]\ e_2[u/d]$
   $=$ [induction hypothesis, 2]
   $e_1\ e_2$

$e_1 \equiv e_2$, $\mathsf{if}\ e_0\ e_1\ e_2$)

Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)

1. $u \notin \mathcal{FV}(\lambda v{:}T.\ e)$ [hypothesis]
2. $u \notin \mathcal{FV}(e) - \{v\}$ [1, $\mathcal{FV}(\lambda v{:}T.\ e) = \mathcal{FV}(e) - \{v\}$]
3. $\mathcal{CV}(\lambda v{:}T.\ e, u) = \emptyset$ [definition of $\mathcal{CV}$, 2]

If $u = v$:

4. $(\lambda v{:}T.\ e)[u/d] = \lambda v{:}T.\ e$ [definition of $\_[\_/\_]$]

If $u \neq v$:

4. $u \notin \mathcal{FV}(e)$ [2]
5. $(\lambda v{:}T.\ e)[u/d]$
   $=$ [definition of $\_[\_/\_]$]
   $\lambda v{:}T.\ e[u/d]$
   $=$ [induction hypothesis, 4]
   $\lambda v{:}T.\ e$

Proof of Theorem 4.2

By induction on e:

v, $o[\overline{T}]$, ι_T, proj f)

1. $\mathcal{CV}(e, u) = \emptyset$ [definition of $\mathcal{CV}$]
2. $u \notin \mathcal{CV}(e, u)$ [1]

$e_1\ e_2$)

1. $\mathcal{CV}(e_1\ e_2, u) = \mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u)$ [definition of $\mathcal{CV}$]
2. $u \notin \mathcal{CV}(e_1, u)\ \wedge\ u \notin \mathcal{CV}(e_2, u)$ [induction hypothesis]
3. $u \notin \mathcal{CV}(e_1\ e_2, u)$ [1, 2]

$e_1 \equiv e_2$, if $e_0\ e_1\ e_2$)
Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)
If $u \notin \mathcal{FV}(e) - \{v\}$:

1. $\mathcal{CV}(\lambda v{:}T.\ e, u) = \emptyset$ [definition of $\mathcal{CV}$]
2. $u \notin \mathcal{CV}(\lambda v{:}T.\ e, u)$ [1]

If $u \in \mathcal{FV}(e) - \{v\}$:

1. $\mathcal{CV}(\lambda v{:}T.\ e, u) = \{v\} \cup \mathcal{CV}(e, u)$ [definition of $\mathcal{CV}$]
2. $u \notin \mathcal{CV}(e, u)$ [induction hypothesis]
3. $u \neq v$ [$u \in \mathcal{FV}(e) - \{v\}$]
4. $u \notin \mathcal{CV}(\lambda v{:}T.\ e, u)$ [1, 2, 3]

Proof of Theorem 4.3

We prove $OKsbs(e, u, u)$ as follows:

1. $u \notin \mathcal{CV}(e, u)$ [Theorem 4.2]
2. $\{u\} \cap \mathcal{CV}(e, u) = \emptyset$ [1]
3. $OKsbs(e, u, u)$ [definition of $OKsbs$, 2]

We prove $e[u/u] = e$ by induction on e:

$o[\overline{T}]$, ι_T, proj f)

$e[u/u] = e$ [definition of $\_[\_/\_]$]

v)
If $v \neq u$:

$v[u/u] = v$ [definition of $\_[\_/\_]$]

If $v = u$:

$v[u/u]$
$\quad =$ [definition of $\_[\_/\_]$]
u
$\quad =$ [$v = u$]
v

$e_1\ e_2$)

$(e_1\ e_2)[u/u]$
$\quad =$ [definition of $\_[\_/\_]$]
$e_1[u/u]\ e_2[u/u]$
$\quad =$ [induction hypothesis]
$e_1\ e_2$

$e_1 \equiv e_2$, $\mathsf{if}\ e_0\ e_1\ e_2$)
Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)
If $u = v$:

$(\lambda v{:}T.\ e)[u/u] = \lambda v{:}T.\ e$ [definition of $\_[\_/\_]$]

If $u \neq v$:

$(\lambda v{:}T.\ e)[u/u]$
$\quad =$ [definition of $\_[\_/\_]$]
$\lambda v{:}T.\ e[u/u]$
$\quad =$ [induction hypothesis]
$\lambda v{:}T.\ e$

Proof of Theorem 4.4

1. $OKsbs(e_1\ e_2, u, d)$ [hypothesis]
2. $\mathcal{FV}(d) \cap \mathcal{CV}(e_1\ e_2, u) = \emptyset$ [1, definition of $OKsbs$]
3. $\mathcal{FV}(d) \cap \mathcal{CV}(e_1, u) = \emptyset\ \wedge\ \mathcal{FV}(d) \cap \mathcal{CV}(e_2, u) = \emptyset$ [2, $\mathcal{CV}(e_1\ e_2, u) = \mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u)$]
4. $OKsbs(e_1, u, d)\ \wedge\ OKsbs(e_2, u, d)$ [3, definition of $OKsbs$]

Proof of Theorem 4.5

Analogous to Theorem 4.4.

Proof of Theorem 4.6

Analogous to Theorem 4.4.

Proof of Theorem 4.7

If $u \notin \mathcal{FV}(e)$:

$OKsbs(e, u, d)$ [Theorem 4.1]

If $u \in \mathcal{FV}(e)$:

1. $OKsbs(\lambda v{:}T.\ e, u, d)$ [hypothesis]
2. $\mathcal{FV}(d) \cap \mathcal{CV}(\lambda v{:}T.\ e, u) = \emptyset$ [1, definition of $OKsbs$]
3. $u \in \mathcal{FV}(e) - \{v\}$ [$u \neq v$, $u \in \mathcal{FV}(e)$]
4. $\mathcal{FV}(d) \cap (\{v\} \cup \mathcal{CV}(e, u)) = \emptyset$ [definition of $\mathcal{CV}$, 3, 2]
5. $\mathcal{FV}(e) \cap \mathcal{CV}(e, u) = \emptyset$ [4]
6. $OKsbs(e, u, d)$ [definition of $OKsbs$, 5]

Proof of Theorem 4.8

By induction on e:

$o[\overline{T}], \iota_T, \mathsf{proj}\ f)$
$\mathcal{FV}(e) = \emptyset \ \Rightarrow\ u \notin \mathcal{FV}(e)$

$v)$

$\mathcal{FV}(v[u/d])$
$=$ [definition of $\_[\_/\_]$, $u \in \mathcal{FV}(v) \ \Rightarrow\ u = v$]
$\mathcal{FV}(d)$
$=$
$\emptyset \cup \mathcal{FV}(d)$
$=$
$(\{u\} - \{u\}) \cup \mathcal{FV}(d)$
$=$ [$u \in \mathcal{FV}(v) \ \Rightarrow\ \mathcal{FV}(v) = \{u\}$]
$(\mathcal{FV}(v) - \{u\}) \cup \mathcal{FV}(d)$

$e_1\ e_2)$
If $u \in \mathcal{FV}(e_1) \ \wedge\ u \in \mathcal{FV}(e_2)$:

$\mathcal{FV}((e_1\ e_2)[u/d])$
$=$ [definition of $\_[\_/\_]$, definition of $\mathcal{FV}$]
$\mathcal{FV}(e_1[u/d]) \cup \mathcal{FV}(e_2[u/d])$
$=$ [Theorem 4.4, induction hypothesis]
$(\mathcal{FV}(e_1) - \{u\}) \cup \mathcal{FV}(d) \cup (\mathcal{FV}(e_2) - \{u\}) \cup \mathcal{FV}(d)$
$=$
$(\mathcal{FV}(e_1) - \{u\}) \cup (\mathcal{FV}(e_2) - \{u\}) \cup \mathcal{FV}(d)$
$=$
$((\mathcal{FV}(e_1) \cup \mathcal{FV}(e_2)) - \{u\}) \cup \mathcal{FV}(d)$
$=$
$(\mathcal{FV}(e_1\ e_2) - \{u\}) \cup \mathcal{FV}(d)$

If $u \in \mathcal{FV}(e_1) \ \wedge\ u \notin \mathcal{FV}(e_2)$:

$\mathcal{FV}((e_1\ e_2)[u/d])$
$=$ [definition of $\_[\_/\_]$, definition of $\mathcal{FV}$]
$\mathcal{FV}(e_1[u/d]) \cup \mathcal{FV}(e_2[u/d])$
$=$ [Theorem 4.4, induction hypothesis, Theorem 4.1]

$(\mathcal{FV}(e_1) - \{u\}) \cup \mathcal{FV}(d) \cup \mathcal{FV}(e_2)$
$\quad = [u \notin \mathcal{FV}(e_2) \;\Rightarrow\; \mathcal{FV}(e_2) = \mathcal{FV}(e_2) - \{u\}]$
$(\mathcal{FV}(e_1) - \{u\}) \cup (\mathcal{FV}(e_2) - \{u\}) \cup \mathcal{FV}(d)$
$\quad =$ [as above]
$(\mathcal{FV}(e_1\ e_2) - \{u\}) \cup \mathcal{FV}(d)$

If $u \notin \mathcal{FV}(e_1) \wedge u \in \mathcal{FV}(e_2)$, the proof is analogous to the previous one, with e_1 and e_2 swapped.

Finally, $u \notin \mathcal{FV}(e_1) \;\wedge\; u \notin \mathcal{FV}(e_2) \;\Rightarrow\; u \notin \mathcal{FV}(e_1\ e_2)$.

$e_1 \equiv e_2$, if $e_0\ e_1\ e_2$)

Analogous to $e_1\ e_2$, using Theorem 4.5 and Theorem 4.6 instead of Theorem 4.4.

$\lambda v{:}T.\ e$)

1. $u \in \mathcal{FV}(\lambda v{:}T.\ e)$ [hypothesis]
2. $OKsbs(\lambda v{:}T.\ e, u, d)$ [hypothesis]
3. $u \neq v$ [1, $\mathcal{FV}(\lambda v{:}T.\ e) = \mathcal{FV}(e) - \{v\}$]
4. $OKsbs(e, u, d)$ [Theorem 4.7, 2, 3]
5. $\mathcal{FV}(d) \cap (\{v\} \cup \mathcal{CV}(e, u)) = \emptyset$ [1, 2]
6. $\mathcal{FV}((\lambda v{:}T.\ e)[u/d])$
   $\quad =$ [3]
   $\mathcal{FV}(\lambda v{:}T.\ e[u/d])$
   $\quad =$
   $\mathcal{FV}(e[u/d]) - \{v\}$
   $\quad =$ [induction hypothesis, 4, $(1 \Rightarrow\ u \in \mathcal{FV}(e))$]
   $(\mathcal{FV}(e) - \{u\}) \cup \mathcal{FV}(d)) - \{v\}$
   $\quad =$
   $((\mathcal{FV}(e) - \{u\}) - \{v\}) \cup (\mathcal{FV}(d) - \{v\})$
   $\quad =$
   $((\mathcal{FV}(e) - \{v\}) - \{u\}) \cup (\mathcal{FV}(d) - \{v\})$
   $\quad =$ [definition of $\mathcal{FV}$]
   $(\mathcal{FV}(\lambda v{:}T.\ e) - \{u\}) \cup (\mathcal{FV}(d) - \{v\})$
   $\quad =$ $[5 \Rightarrow\ v \notin \mathcal{FV}(d)]$
   $(\mathcal{FV}(\lambda v{:}T.\ e) - \{u\}) \cup \mathcal{FV}(d)$

Proof of Theorem 4.9

If $u \notin \mathcal{FV}(e)$:

1. $e[u/d] = e$ [Theorem 4.1]
2. $u \notin \mathcal{FV}(e[u/d])$ [1, hypothesis $u \notin \mathcal{FV}(e)$]

If $u \in \mathcal{FV}(e)$:

1. $\mathcal{FV}(e[u/d]) = (\mathcal{FV}(e) - \{u\}) \cup \mathcal{FV}(d)$ [Theorem 4.8, hypothesis $OKsbs(e, u, d)$]
2. $u \notin \mathcal{FV}(e) - \{u\}$
3. $u \notin \mathcal{FV}(d)$ [hypothesis]
4. $u \notin \mathcal{FV}(e[u/d])$ [1, 2, 3]

Proof of Theorem 4.10

By induction on e:

$o[\overline{T}]$, ι_T, $\mathsf{proj}\ f$)

$$e[u/d][u'/d'] = e[u'/d'] = e = e[u/d] = e[u'/d'][u/d] \qquad \text{[definition of } \_[\_/\_]\text{]}$$

u)

$$\begin{array}{l} u[u/d][u'/d'] \\ \quad = \text{ [definition of } \_[\_/\_]\text{]} \\ d[u'/d'] \\ \quad = \text{ [Theorem 4.1, hypothesis } u' \notin \mathcal{FV}(d)\text{]} \\ d \\ \quad = \text{ [definition of } \_[\_/\_]\text{]} \\ u[u/d] \\ \quad = \text{ [definition of } \_[\_/\_]\text{, hypothesis } u \neq u'\text{]} \\ u[u'/d'][u/d] \end{array}$$

u')

Symmetric to previous proof.

$v \notin \{u, u'\}$)

$$v[u/d][u'/d'] = v[u'/d'] = v = v[u/d] = v[u'/d'][u/d] \qquad \text{[definition of } \_[\_/\_]\text{]}$$

$e_1\ e_2$)

$$\begin{array}{l} (e_1\ e_2)[u/d][u'/d'] \\ \quad = \\ e_1[u/d][u'/d']\ e_2[u/d][u'/d'] \\ \quad = \text{ [induction hypothesis]} \\ e_1[u'/d'][u/d]\ e_2[u'/d'][u/d] \\ \quad = \\ (e_1\ e_2)[u'/d'][u/d] \end{array}$$

$e_1 \equiv e_2$, $\mathsf{if}\ e_0\ e_1\ e_2$)

Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)

If $v = u$:

1. $v \neq u'$ [hypothesis $u \neq u'$]
2. $$\begin{array}{l} (\lambda v{:}T.\ e)[u/d][u'/d'] \\ \quad = \text{ [} v = u\text{]} \\ (\lambda v{:}T.\ e)[u'/d'] \\ \quad = \text{ [1]} \\ \lambda v{:}T.\ e[u'/d'] \\ \quad = \text{ [} v = u\text{]} \\ (\lambda v{:}T.\ e[u'/d'])[u/d] \\ \quad = \text{ [1]} \\ (\lambda v{:}T.\ e)[u'/d'][u/d] \end{array}$$

If $v = u'$, the proof is symmetric to the previous one.

If $v \notin \{u, u'\}$:

$(\lambda v{:}T.\ e)[u/d][u'/d']$
$\quad =$
$\lambda v{:}T.\ e[u/d][u'/d']$
$\quad =$ [induction hypothesis]
$\lambda v{:}T.\ e[u'/d'][u/d]$
$\quad =$
$(\lambda v{:}T.\ e)[u'/d'][u/d]$

Proof of Theorem 4.11

By induction on e:

u)

$u[u/d][u/d'] = d[u/d'] = u[u/d[u/d']]$ [definition of $\_[\_/\_]$]

$v \neq u$)

$v[u/d][u/d'] = v = v[u/d[u/d']]$ [definition of $\_[\_/\_]$]

$o[\overline{T}]$, ι_T, $\mathsf{proj}\ f$) :
Similar to case $v \neq u$ above.

$e_1\ e_2$)

$(e_1\ e_2)[u/d][u/d']$
$\quad =$ [definition of $\_[\_/\_]$]
$e_1[u/d][u/d']\ e_2[u/d][u/d']$
$\quad =$ [induction hypothesis]
$e_1[u/d[u/d']]\ e_2[u/d[u/d']]$
$\quad =$ [definition of $\_[\_/\_]$]
$(e_1\ e_2)[u/d[u/d']]$

$e_1 \equiv e_2$, $\mathsf{if}\ e_0\ e_1\ e_2$)
Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)
If $v = u$:

$(\lambda v{:}T.\ e)[u/d][u/d'] = \lambda v{:}T.\ e = (\lambda v{:}T.\ e)[u/d[u/d']]$ [definition of $\_[\_/\_]$]

If $v \neq u$:

$(\lambda v{:}T.\ e)[u/d][u/d']$
$\quad =$ [definition of $\_[\_/\_]$]
$\lambda v{:}T.\ e[u/d][u/d']$
$\quad =$ [induction hypothesis]
$\lambda v{:}T.\ e[u/d[u/d']]$
$\quad =$ [definition of $\_[\_/\_]$]
$(\lambda v{:}T.\ e)[u/d[u/d']]$

Proof of Theorem 4.12

If $u = u'$:

$e[u/u'][u/u'] = e = e[u/u']$ [Theorem 4.3]

If $u \neq u'$, the proof is by induction on e:

u)

$u[u/u'][u/u'] = u' = u[u/u']$ [definition of $\_[\_/\_]$]

$v \neq u$)

$v[u/u'][u/u'] = v = v[u/u']$ [definition of $\_[\_/\_]$]

$o[\overline{T}]$, ι_T, proj f) :

$e[u/u'][u/u'] = e = e[u/u']$ [Theorem 4.1]

$e_1\ e_2$)

$(e_1\ e_2)[u/u'][u/u'] = e_1[u/u'][u/u']\ e_2[u/u'][u/u'] = e_1[u/u']\ e_2[u/u'] = (e_1\ e_2)[u/u']$
[definition of $\_[\_/\_]$, induction hypothesis]

$e_1 \equiv e_2$, if $e_0\ e_1\ e_2$)
Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)
If $u = v$:

$(\lambda v{:}T.\ e)[u/u'][u/u'] = \lambda v{:}T.\ e = (\lambda v{:}T.\ e)[u/u']$ [definition of $\_[\_/\_]$]

If $u \neq v$:

$(\lambda v{:}T.\ e)[u/u'][u/u'] = \lambda v{:}T.\ e[u/u'][u/u'] = \lambda v{:}T.\ e[u/u'] = (\lambda v{:}T.\ e)[u/u']$
[definition of $\_[\_/\_]$, induction hypothesis]

Proof of Theorem 4.13

By induction on e:

w)

$w[w/d][u/u']$
$\quad =$ [definition of $\_[\_/\_]$]
$d[u/u']$
$\quad =$ [Theorem 4.12]
$d[u/u'][u/u']$
$\quad =$ [definition of $\_[\_/\_]$]
$w[w/d[u/u']][u/u']$

$v \neq w$)

$v[w/d][u/u'] = v[u/u'] = v[w/d[u/u']][u/u']$ [definition of $\_[\_/\_]$]

$o[\overline{T}]$, ι_T, proj f) :

$e[w/d][u/u'] = e = e[w/d[u/u']][u/u']$ [Theorem 4.1]

$e_1\ e_2$)

1. $OKsbs(e_1, w, d)\ \wedge\ OKsbs(e_2, w, d)$ [Theorem 4.4, hypothesis]
2. $(e_1\ e_2)[w/d][u/u']$
$\quad =$ [definition of $\_[\_/\_]$]
$e_1[w/d][u/u']\ e_2[w/d][u/u']$
$\quad =$ [induction hypothesis, 1]
$e_1[w/d[u/u']][u/u']\ e_2[w/d[u/u']][u/u']$
$\quad =$ [definition of $\_[\_/\_]$]
$(e_1\ e_2)[w/d[u/u']][u/u']$

$e_1 \equiv e_2$, $\mathsf{if}\ e_0\ e_1\ e_2$)
Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)
If $w \notin \mathcal{FV}(\lambda v{:}T.\ e)$:

$$(\lambda v{:}T.\ e)[w/d][u/u'] = (\lambda v{:}T.\ e)[u/u'] = (\lambda v{:}T.\ e)[w/d[u/u']][u/u']$$
[definition of $\_[\_/\_]$, Theorem 4.1]

If $w \in \mathcal{FV}(\lambda v{:}T.\ e)$:

1. $w \in \mathcal{FV}(e) - \{v\}$ [definition of $\_[\_/\_]$]
2. $w \neq v$ [1]

If $u \notin \mathcal{FV}(d)$:

3. $(\lambda v{:}T.\ e)[w/d][u/u']$
   $=$ [definition of $\_[\_/\_]$, 2]
   $(\lambda v{:}T.\ e[w/d])[u/u']$
   $=$ [hypothesis $u \notin \mathcal{FV}(d)$, Theorem 4.1]
   $(\lambda v{:}T.\ e[w/d[u/u']])[u/u']$
   $=$ [definition of $\_[\_/\_]$, 2]
   $(\lambda v{:}T.\ e)[w/d[u/u']][u/u']$

If $u \in \mathcal{FV}(d)$:

3. $\mathcal{FV}(d) \cap \mathcal{CV}(\lambda v{:}T.\ e, w) = \emptyset$ [hypothesis $OKsbs(\lambda v{:}T.\ e, w, d)$]
4. $\mathcal{CV}(\lambda v{:}T.\ e, w) = \{v\} \cup \mathcal{CV}(e, w)$ [definition of $\mathcal{CV}$, 1]
5. $v \notin \mathcal{FV}(d)$ [3, 4]
6. $v \neq u$ [5, hypothesis $u \in \mathcal{FV}(d)$]
7. $OKsbs(e, w, d)$ [Theorem 4.7, hypothesis $OKsbs(\lambda v{:}T.\ e, w, d)$, 2]
8. $(\lambda v{:}T.\ e)[w/d][u/u']$
   $=$ [definition of $\_[\_/\_]$, 2, 6]
   $\lambda v{:}T.\ e[w/d][u/u']$
   $=$ [induction hypothesis, 7]
   $\lambda v{:}T.\ e[w/d[u/u']][u/u']$
   $=$ [definition of $\_[\_/\_]$, 2, 6]
   $(\lambda v{:}T.\ e)[w/d[u/u']][u/u']$

Proof of Theorem 4.14

By induction on e:

$o[\overline{T}]$, ι_T, $\mathsf{proj}\ f$)
$e[u/u'][u'/u] = e[u'/u] = e$

u)
$u[u/u'][u'/u] = u'[u'/u] = u$

u')
$u' \in \mathcal{FV}(u')$

$v \notin \{u, u'\}$)
$v[u/u'][u'/u] = v[u'/u] = v$

$e_1\ e_2$)

1. $OKsbs(e_1\ e_2, u, u')$ [hypothesis]
2. $u' \notin \mathcal{FV}(e_1\ e_2)$ [hypothesis]
3. $OKsbs(e_1, u, u')\ \wedge\ OKsbs(e_1, u, u')$ [Theorem 4.4, 1]
4. $u' \notin \mathcal{FV}(e_1)\ \wedge\ u' \notin \mathcal{FV}(e_2)$ [2]
5. $(e_1\ e_2)[u/u'][u'/u]$
   $=$
   $e_1[u/u'][u'/u]\ e_2[u/u'][u'/u]$
   $=$ [induction hypothesis, 3, 4]
   $e_1\ e_2$

$e_1 \equiv e_2$, $\mathsf{if}\ e_0\ e_1\ e_2$)
Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)
If $u \notin \mathcal{FV}(\lambda v{:}T.\ e)$:

$(\lambda v{:}T.\ e)[u/u'][u'/u]$
$=$ [Theorem 4.1, hypothesis $u \notin \mathcal{FV}(\lambda v{:}T.\ e)$]
$(\lambda v{:}T.\ e)[u'/u]$
$=$ [Theorem 4.1, hypothesis $u' \notin \mathcal{FV}(\lambda v{:}T.\ e)$]
$\lambda v{:}T.\ e$

If $u \in \mathcal{FV}(\lambda v{:}T.\ e)$:

1. $u \in \mathcal{FV}(e) - \{v\}$ [$\mathcal{FV}(\lambda v{:}T.\ e) = \mathcal{FV}(e) - \{v\}$]
2. $u \neq v$ [1]
3. $\{u'\} \cap (\{v\} \cup \mathcal{CV}(e, u)) = \emptyset$ [hypothesis $OKsbs(\lambda v{:}T.\ e, u, u')$, 1]
4. $u' \neq v$ [3]
5. $OKsbs(e, u, u')$ [Theorem 4.7, hypothesis $OKsbs(\lambda v{:}T.\ e, u, u')$, 2]
6. $u' \notin \mathcal{FV}(e) - \{v\}$ [hypothesis $u' \notin \mathcal{FV}(\lambda v{:}T.\ e)$]
7. $u' \notin \mathcal{FV}(e)$ [6, 4]
8. $(\lambda v{:}T.\ e)[u/u'][u'/u]$
   $=$ [2]
   $(\lambda v{:}T.\ e[u/u'])[u'/u]$
   $=$ [4]
   $\lambda v{:}T.\ e[u/u'][u'/u]$
   $=$ [induction hypothesis, 5, 7]
   $\lambda v{:}T.\ e$

Proof of Theorem 4.15

By induction on e:

$o[\overline{T}]$, ι_T, $\mathsf{proj}\ f$)
$\mathcal{CV}(e, u) = \emptyset = \mathcal{CV}(e, u') = \mathcal{CV}(e[u/u'], u')$

u)
$\mathcal{CV}(u, u) = \emptyset = \mathcal{CV}(u', u') = \mathcal{CV}(u[u/u'], u')$

$v \neq u$)
$\mathcal{CV}(v, u) = \emptyset = \mathcal{CV}(v, u') = \mathcal{CV}(v[u/u'], u')$

$e_1\ e_2$)

1. $OKsbs(e_1, u, u') \ \wedge \ OKsbs(e_2, u, u')$ [Theorem 4.4, hypothesis $OKsbs(e_1\ e_2, u, u')$]
2. $u' \notin \mathcal{FV}(e_1) \ \wedge \ u' \notin \mathcal{FV}(e_2)$ [hypothesis $u' \notin \mathcal{FV}(e_1\ e_2)$]
3. $\mathcal{CV}(e_1\ e_2, u)$
   $=$
   $\mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u)$
   $=$ [induction hypothesis, 1, 2]
   $\mathcal{CV}(e_1[u/u'], u') \cup \mathcal{CV}(e_2[u/u'], u')$
   $=$
   $\mathcal{CV}(e_1[u/u']\ e_2[u/u'], u')$
   $=$
   $\mathcal{CV}((e_1\ e_2)[u/u'], u')$

$e_1 \equiv e_2$, $\mathsf{if}\ e_0\ e_1\ e_2$)
Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)
If $u \notin \mathcal{FV}(\lambda v{:}T.\ e)$:

$\mathcal{CV}(\lambda v{:}T.\ e, u)$
$=$ [hypothesis $u \notin \mathcal{FV}(\lambda v{:}T.\ e)$]
$\emptyset$
$=$ [hypothesis $u' \notin \mathcal{FV}(\lambda v{:}T.\ e)$]
$\mathcal{CV}(\lambda v{:}T.\ e, u')$
$=$ [Theorem 4.1, hypothesis $u \notin \mathcal{FV}(\lambda v{:}T.\ e)$]
$\mathcal{CV}((\lambda v{:}T.\ e)[u/u'], u)$

If $u \in \mathcal{FV}(\lambda v{:}T.\ e)$:

1. $u \in \mathcal{FV}(e) - \{v\}$ [$\mathcal{FV}(\lambda v{:}T.\ e) = \mathcal{FV}(e) - \{v\}$]
2. $u \neq v$ [1]
3. $\{u'\} \cap (\{v\} \cup \mathcal{CV}(e, u)) = \emptyset$ [hypothesis $OKsbs(\lambda v{:}T.\ e, u, u')$, 1]
4. $u' \neq v$ [3]
5. $OKsbs(e, u, u')$ [Theorem 4.7, hypothesis $OKsbs(\lambda v{:}T.\ e, u, u')$, 2]
6. $u' \notin \mathcal{FV}(e) - \{v\}$ [hypothesis $u' \notin \mathcal{FV}(\lambda v{:}T.\ e)$]
7. $u' \notin \mathcal{FV}(e)$ [6, 4]
8. $u \in \mathcal{FV}(e)$ [1]
9. $\mathcal{FV}(e[u/u']) = (\mathcal{FV}(e) - \{u\}) \cup \{u'\}$ [Theorem 4.8, 5, 8]
10. $u' \in \mathcal{FV}(e[u/u'])$ [9]
11. $u' \in \mathcal{FV}(e[u/u']) - \{v\}$ [10, 4]
12. $\mathcal{CV}(\lambda v{:}T.\ e, u)$
    $=$ [1]
    $\{v\} \cup \mathcal{CV}(e, u)$
    $=$ [induction hypothesis, 5, 7]
    $\{v\} \cup \mathcal{CV}(e[u/u'], u')$
    $=$ [11]
    $\mathcal{CV}(\lambda v{:}T.\ e[u/u'], u')$
    $=$ [2]
    $\mathcal{CV}((\lambda v{:}T.\ e)[u/u'], u')$

Proof of Theorem 4.16

By induction on e:

v)

1. $v = u$ [hypothesis $u \in \mathcal{FV}(v)$]
2. $w \notin \mathcal{FV}(d) \;\Rightarrow\; \mathcal{CV}(d, w) = \emptyset$ [Theorem 4.1]
3. $\mathcal{CV}(v[u/d], w)$
   $=$ [1]
   $\mathcal{CV}(d, w)$
   $=$
   $\emptyset \cup \begin{cases} \mathcal{CV}(d,w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \mathcal{CV}(d,w) & \text{otherwise} \end{cases}$
   $=$ [2]
   $\begin{cases} \emptyset & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(d,w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$
   $=$ [$\mathcal{CV}(v, w) = \mathcal{CV}(v, u) = \emptyset$]
   $\begin{cases} \mathcal{CV}(v,w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(v,u) \cup \mathcal{CV}(d,w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$

$o[\overline{T}]$, ι_T, $\mathsf{proj}\ f$)
These cases are impossible because $\mathcal{FV}(e) = \emptyset$.

$e_1\ e_2$)

1. $u \in \mathcal{FV}(e_1\ e_2)$ [hypothesis]
2. $u \in \mathcal{FV}(e_1) \;\vee\; u \in \mathcal{FV}(e_2)$ [1]
3. $OKsbs(e_1\ e_2, u, d)$ [hypothesis]
4. $OKsbs(e_1, u, d) \;\wedge\; OKsbs(e_2, u, d)$ [Theorem 4.4, 3]

If $u \in \mathcal{FV}(e_1) \;\wedge\; u \in \mathcal{FV}(e_2)$:

$\mathcal{CV}((e_1\ e_2)[u/d], w)$
$=$
$\mathcal{CV}(e_1[u/d], w) \cup \mathcal{CV}(e_2[u/d], w)$
$=$ [induction hypothesis, 4]
$\begin{cases} \mathcal{CV}(e_1,w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(e_1,w) \cup \mathcal{CV}(d,w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases} \cup$
$\begin{cases} \mathcal{CV}(e_2,w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(e_2,w) \cup \mathcal{CV}(d,w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$
$=$
$\begin{cases} \mathcal{CV}(e_1,w) \cup \mathcal{CV}(e_2,w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup$
$\begin{cases} \mathcal{CV}(e_1,w) \cup \mathcal{CV}(e_2,w) \cup \mathcal{CV}(d,w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$
$=$
$\begin{cases} \mathcal{CV}(e_1\ e_2,w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(e_1\ e_2,w) \cup \mathcal{CV}(d,w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$

If $u \in \mathcal{FV}(e_1) \;\wedge\; u \notin \mathcal{FV}(e_2)$:

5. $\mathcal{CV}(e_2, u) = \emptyset$ [Theorem 4.1]
6. $w = u \;\Rightarrow\; \mathcal{CV}(e_2, w) = \emptyset$ [5]

7. $\mathcal{CV}((e_1\ e_2)[u/d], w)$
   $=$
   $\mathcal{CV}(e_1[u/d], w) \cup \mathcal{CV}(e_2[u/d], w)$
   $=$ [Theorem 4.1]
   $\mathcal{CV}(e_1[u/d], w) \cup \mathcal{CV}(e_2, w)$
   $=$ [induction hypothesis, 4]
   $\begin{cases} \mathcal{CV}(e_1, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(e_1, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases} \cup \mathcal{CV}(e_2, w)$
   $=$
   $\begin{cases} \mathcal{CV}(e_1, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(e_1, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases} \cup$
   $\begin{cases} \mathcal{CV}(e_2, w) & \text{if} \quad w \neq u \\ \mathcal{CV}(e_2, w) & \text{otherwise} \end{cases}$
   $=$ [6]
   $\begin{cases} \mathcal{CV}(e_1, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(e_1, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases} \cup$
   $\begin{cases} \mathcal{CV}(e_2, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases}$
   $=$
   $\begin{cases} \mathcal{CV}(e_1, w) \cup \mathcal{CV}(e_2, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(e_1, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$
   $=$
   $\begin{cases} \mathcal{CV}(e_1\ e_2, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(e_1, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$
   $=$ [5]
   $\begin{cases} \mathcal{CV}(e_1\ e_2, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$
   $=$
   $\begin{cases} \mathcal{CV}(e_1\ e_2, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(e_1\ e_2, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$

If $u \notin \mathcal{FV}(e_1) \wedge u \in \mathcal{FV}(e_2)$, the proof is analogous to the previous one, with e_1 and e_2 swapped.

$\lambda v{:}T.\ e$)

1. $u \in \mathcal{FV}(\lambda v{:}T.\ e)$ [hypothesis]
2. $u \in \mathcal{FV}(e)$ [1]
3. $u \neq v$ [1]
4. $OKsbs(\lambda v{:}T.\ e, u, d)$ [hypothesis]
5. $OKsbs(e, u, d)$ [Theorem 4.7, 4, 3]
6. $\mathcal{CV}((\lambda v{:}T.\ e)[u/d], w) = \mathcal{CV}(\lambda v{:}T.\ e[u/d], w)$ [3]
7. $\mathcal{FV}(e[u/d]) = (\mathcal{FV}(e) - \{u\}) \cup \mathcal{FV}(d)$ [Theorem 4.8, 2, 5]

If $w \notin \mathcal{FV}(e[u/d]) - \{v\}$:

8. $\mathcal{CV}(\lambda v{:}T.\ e[u/d], w) = \emptyset$

If $w = v$:

9. $\mathcal{CV}(\lambda v{:}T.\ e, w) = \emptyset$
10. $\mathcal{FV}(d) \cap \mathcal{CV}(\lambda v{:}T.\ e, u) = \emptyset$ [4]
11. $\mathcal{CV}(\lambda v{:}T.\ e, u) = \{v\} \cup \mathcal{CV}(e, u)$ [2, 3]
12. $v \notin \mathcal{FV}(d)$ [10, 11]
13. $w \notin \mathcal{FV}(d)$ [12, hypothesis $w = v$]

14. $\mathcal{CV}((\lambda v{:}T.\ e)[u/d], w)$
$$\overset{=}{\begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases}} \cup \begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$$
[6, 8, 9, 13]

If $w \neq v$:

9. $w \notin (\mathcal{FV}(e) - \{u\}) \cup \mathcal{FV}(d)$ [7, hypothesis $w \notin \mathcal{FV}(e[u/d]) - \{v\}$]
10. $w \notin \mathcal{FV}(e) - \{u\}$ [9]
11. $w \neq u \ \Rightarrow \ w \notin \mathcal{FV}(e)$ [10]
12. $w \notin \mathcal{FV}(e) \ \Rightarrow \ \mathcal{CV}(\lambda v{:}T.\ e, w) = \emptyset$
13. $w \notin \mathcal{FV}(d)$ [9]
14. $\mathcal{CV}((\lambda v{:}T.\ e)[u/d], w)$
$$\overset{=}{\begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases}} \cup \begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$$
[6, 8, 11, 12, 13]

If $w \in \mathcal{FV}(e[u/d]) - \{v\}$:

8. $\mathcal{CV}(\lambda v{:}T.\ e[u/d], w)$
$$\overset{=}{\{v\} \cup \mathcal{CV}(e[u/d], w)}$$
$$= \text{ [induction hypothesis, 2, 5]}$$
$$\{v\} \cup \begin{cases} \mathcal{CV}(e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$$
9. $w \in \mathcal{FV}(e[u/d]) \ \wedge \ w \neq v$ [hypothesis $w \in \mathcal{FV}(e[u/d]) - \{v\}$]
10. $w \in (\mathcal{FV}(e) - \{u\}) \cup \mathcal{FV}(d)$ [9, 7]
11. $w \neq v$ [9]
12. $\mathcal{CV}(\lambda v{:}T.\ e, u) = \{v\} \cup \mathcal{CV}(e, u)$ [2, 3]

If $w \in \mathcal{FV}(d) \ \wedge \ w \neq u$:

13. $\mathcal{CV}(\lambda v{:}T.\ e[u/d], w) = \{v\} \cup \mathcal{CV}(e, w) \cup \mathcal{CV}(e, u) \cup \mathcal{CV}(d, w)$ [8]

If $w \in \mathcal{FV}(e) - \{v\}$:

14. $$\begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$$
$$\overset{=}{\{v\} \cup \mathcal{CV}(e, w) \cup \{v\} \cup \mathcal{CV}(e, u) \cup \mathcal{CV}(d, w)}$$ [12]
15. $\mathcal{CV}((\lambda v{:}T.\ e)[u/d], w)$
$$\overset{=}{\begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases}} \cup \begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$$
[6, 13, 14]

If $w \notin \mathcal{FV}(e) - \{v\}$:

14. $$\begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$$
$$\overset{=}{\emptyset \cup \{v\} \cup \mathcal{CV}(e, u) \cup \mathcal{CV}(d, w)}$$ [12]
15. $w \notin \mathcal{FV}(e)$ [11, hypothesis $w \notin \mathcal{FV}(e) - \{v\}$]

16. $\mathcal{CV}(e, w) = \emptyset$ [Theorem 4.1, 15]
17. $\mathcal{CV}(\lambda v{:}T.\ e[u/d], w) = \{v\} \cup \mathcal{CV}(e, u) \cup \mathcal{CV}(d, w)$ [16, 13]
18. $\mathcal{CV}((\lambda v{:}T.\ e)[u/d], w)$
$\overset{=}{\begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases}} \cup \begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$
[6, 17, 14]

If $w \in \mathcal{FV}(d) \ \wedge \ w = u$:

13. $\mathcal{CV}(\lambda v{:}T.\ e[u/d], w) = \{v\} \cup \mathcal{CV}(e, u) \cup \mathcal{CV}(d, w)$ [8]
14. $\begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$
$\overset{=}{\emptyset \cup \{v\} \cup \mathcal{CV}(e, u) \cup \mathcal{CV}(d, w)}$ [12]
15. $\mathcal{CV}((\lambda v{:}T.\ e)[u/d], w)$
$\overset{=}{\begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases}} \cup \begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$
[6, 13, 14]

If $w \notin \mathcal{FV}(d)$:

13. $w \in \mathcal{FV}(e) - \{u\}$ [10]
14. $w \in \mathcal{FV}(e)$ [13]
15. $w \neq u$ [13]
16. $\mathcal{CV}(\lambda v{:}T.\ e[u/d], w) = \{v\} \cup \mathcal{CV}(e, w) \cup \emptyset$ [8, 15, hypothesis $w \notin \mathcal{FV}(d)$]
17. $\mathcal{CV}(\lambda v{:}T.\ e, w) = \{v\} \cup \mathcal{CV}(e, w)$ [14, 11]
18. $\begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases} \cup \begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$
$\overset{=}{\{v\} \cup \mathcal{CV}(e, w) \cup \emptyset}$ [15, 17, hypothesis $w \notin \mathcal{FV}(d)$]
19. $\mathcal{CV}((\lambda v{:}T.\ e)[, /w])$
$\overset{=}{\begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, w) & \text{if} \quad w \neq u \\ \emptyset & \text{otherwise} \end{cases}} \cup \begin{cases} \mathcal{CV}(\lambda v{:}T.\ e, u) \cup \mathcal{CV}(d, w) & \text{if} \quad w \in \mathcal{FV}(d) \\ \emptyset & \text{otherwise} \end{cases}$ [6, 16, 18]

Proof of Theorem 4.17

By induction on e:

$o[\overline{T}]$, ι_T, proj f)
$e[u/u'] = e \ \Rightarrow \ \mathcal{CV}(e, w) = \mathcal{CV}(e[u/u'], w)$

u)
$\mathcal{CV}(u, w) = \emptyset = \mathcal{CV}(u', w) = \mathcal{CV}(u[u/u'], w)$

$v \neq u$)
$v[u/u'] = v \ \Rightarrow \ \mathcal{CV}(v, w) = \mathcal{CV}(v[u/u'], w)$

$e_1\ e_2$)

1. $OKsbs(e_1, u, u') \ \wedge \ OKsbs(e_2, u, u')$ [Theorem 4.4, hypothesis $OKsbs(e_1\ e_2, u, u')$]

2. $\mathcal{CV}(e_1\ e_2, w)$
   $=$
   $\mathcal{CV}(e_1, w) \cup \mathcal{CV}(e_2, w)$
   $=$ [induction hypothesis, 1]
   $\mathcal{CV}(e_1[u/u'], w) \cup \mathcal{CV}(e_2[u/u'], w)$
   $=$
   $\mathcal{CV}(e_1[u/u']\ e_2[u/u'], w)$
   $=$
   $\mathcal{CV}((e_1\ e_2)[u/u'], w)$

$e_1 \equiv e_2$, if $e_0\ e_1\ e_2$)
Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)
If $u \notin \mathcal{FV}(\lambda v{:}T.\ e)$:

1. $(\lambda v{:}T.\ e)[u/u'] = \lambda v{:}T.\ e$ [Theorem 4.1, hypothesis $u \notin \mathcal{FV}(\lambda v{:}T.\ e)$]
2. $\mathcal{CV}(\lambda v{:}T.\ e, w) = \mathcal{CV}((\lambda v{:}T.\ e)[u/u'], w)$ [1]

If $u \in \mathcal{FV}(\lambda v{:}T.\ e)$:

1. $\mathcal{CV}(\lambda v{:}T.\ e, w) = \begin{cases} \{v\} \cup \mathcal{CV}(e, w) & \text{if} \quad w \in \mathcal{FV}(e) - \{v\} \\ \emptyset & \text{otherwise} \end{cases}$ [$\mathcal{CV}$]
2. $u \in \mathcal{FV}(\lambda v{:}T.\ e)$ [hypothesis]
3. $u \in \mathcal{FV}(e) - \{v\}$ [2, $\mathcal{FV}$]
4. $u \neq v$ [3]
5. $\mathcal{CV}((\lambda v{:}T.\ e)[u/u'], w) = \mathcal{CV}(\lambda v{:}T.\ e[u/u'], w)$ [4, $\_[\_/\_]$]
6. $\mathcal{CV}(\lambda v{:}T.\ e[u/u'], w) = \begin{cases} \{v\} \cup \mathcal{CV}(e[u/u'], w) & \text{if} \quad w \in \mathcal{FV}(e[u/u']) - \{v\} \\ \emptyset & \text{otherwise} \end{cases}$ [$\mathcal{CV}$]
7. $OKsbs(\lambda v{:}T.\ e, u, u')$ [hypothesis]
8. $OKsbs(e, u, u')$ [Theorem 4.7, 7, 4]
9. $w \neq u\ \wedge\ w \neq u'$ [hypothesis]
10. $\mathcal{CV}(e, w) = \mathcal{CV}(e[u/u'], w)$ [induction hypothesis, 8, 9]
11. $u \in \mathcal{FV}(e)$ [3]
12. $\mathcal{FV}(e[u/u']) = (\mathcal{FV}(e) - \{u\}) \cup \{u'\}$ [Theorem 4.8, 8, 11]
13. $w \in \mathcal{FV}(e)$
    $\Rightarrow$ [9]
    $w \in \mathcal{FV}(e) - \{u\}$
    $\Rightarrow$
    $w \in (\mathcal{FV}(e) - \{u\}) \cup \{u'\}$
    $\Rightarrow$ [12]
    $w \in \mathcal{FV}(e[u/u'])$
14. $w \in \mathcal{FV}(e[u/u'])$
    $\Rightarrow$ [12]
    $w \in (\mathcal{FV}(e) - \{u\}) \cup \{u'\}$
    $\Rightarrow$ [9]
    $w \in \mathcal{FV}(e) - \{u\}$
    $\Rightarrow$
    $w \in \mathcal{FV}(e)$
15. $w \in \mathcal{FV}(e)\ \Leftrightarrow\ w \in \mathcal{FV}(e[u/u'])$ [13, 14]
16. $\mathcal{CV}(\lambda v{:}T.\ e, w) = \mathcal{CV}((\lambda v{:}T.\ e)[u/u'], w)$ [1, 10, 15, 5]

Proof of Theorem 4.18

If $v \notin \mathcal{FV}(e)$:

$\mathcal{FV}(\lambda v'{:}T.\ e[v/v'])$
$\quad =$ [Theorem 4.1, hypothesis $v \notin \mathcal{FV}(e)$]
$\mathcal{FV}(\lambda v'{:}T.\ e)$
$\quad =$ [$\mathcal{FV}$]
$\mathcal{FV}(e) - \{v'\}$
$\quad =$ [hypothesis $v' \notin \mathcal{FV}(e)$]
$\mathcal{FV}(e)$
$\quad =$ [hypothesis $v \notin \mathcal{FV}(e)$]
$\mathcal{FV}(e) - \{v\}$
$\quad =$ [$\mathcal{FV}$]
$\mathcal{FV}(\lambda v{:}T.\ e)$

If $v \in \mathcal{FV}(e)$:

1. $v' \notin \mathcal{CV}(e, v)$ [hypothesis]
2. $\{v'\} \cap \mathcal{CV}(e, v) = \emptyset$ [1]
3. $OKsbs(e, v, v')$ [2, $OKsbs$]
4. $\mathcal{FV}(\lambda v'{:}T.\ e[v/v'])$
$\quad =$ [$\mathcal{FV}$]
$\mathcal{FV}(e[v/v']) - \{v'\}$
$\quad =$ [Theorem 4.8, hypothesis $v \in \mathcal{FV}(e)$, 3]
$((\mathcal{FV}(e) - \{v\}) \cup \{v'\}) - \{v'\}$
$\quad =$
$\mathcal{FV}(e) - \{v, v'\}$
$\quad =$ [hypothesis $v' \notin \mathcal{FV}(e)$]
$\mathcal{FV}(e) - \{v\}$
$\quad =$ [$\mathcal{FV}$]
$\mathcal{FV}(\lambda v{:}T.\ e)$

Proof of Theorem 4.19

By induction on e:

v)

$\mathcal{FV}(v[\sigma]) = \mathcal{FV}(v)$ [definition of $\_[\_]$]

$o[\overline{T}]$)

$\mathcal{FV}(o[\overline{T}][\sigma])$
$\quad =$ [definition of $\_[\_]$]
$\mathcal{FV}(o[\overline{T}[\sigma]])$
$\quad =$ [definition of $\mathcal{FV}$]
$\emptyset$
$\quad =$ [definition of $\mathcal{FV}$]
$\mathcal{FV}(o[\overline{T}])$

ι_T, proj f)
Analogous to $o[\overline{T}]$.

$e_1\ e_2$)

$$
\begin{array}{l}
\mathcal{FV}((e_1\ e_2)[\sigma]) \\
\quad = \text{ [definition of } \_[\_]] \\
\mathcal{FV}(e_1[\sigma]\ e_2[\sigma]) \\
\quad = \text{ [definition of } \mathcal{FV}] \\
\mathcal{FV}(e_1[\sigma]) \cup \mathcal{FV}(e_2[\sigma]) \\
\quad = \text{ [induction hypothesis]} \\
\mathcal{FV}(e_1) \cup \mathcal{FV}(e_1) \\
\quad = \text{ [definition of } \mathcal{FV}] \\
\mathcal{FV}(e_1\ e_2)
\end{array}
$$

$e_1 \equiv e_2$, $\mathsf{if}\ e_0\ e_1\ e_2$)
Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)

$$
\begin{array}{l}
\mathcal{FV}((\lambda v{:}T.\ e)[\sigma]) \\
\quad = \text{ [definition of } \_[\_]] \\
\mathcal{FV}(\lambda v{:}T[\sigma].\ e[\sigma]) \\
\quad = \text{ [definition of } \mathcal{FV}] \\
\mathcal{FV}(e[\sigma]) - \{v\} \\
\quad = \text{ [induction hypothesis]} \\
\mathcal{FV}(e) - \{v\} \\
\quad = \text{ [definition of } \mathcal{FV}] \\
\mathcal{FV}(\lambda v{:}T.\ e)
\end{array}
$$

Proof of Theorem 4.20

By induction on e:

v, $o[\overline{T}]$, ι_T, $\mathsf{proj}\ f$)

$$\mathcal{CV}(e[\sigma], u) = \emptyset = \mathcal{CV}(e, u)$$

$e_1\ e_2$)

$$
\begin{array}{l}
\mathcal{CV}((e_1\ e_2)[\sigma], u) \\
\quad = \\
\mathcal{CV}(e_1[\sigma], u) \cup \mathcal{CV}(e_2[\sigma], u) \\
\quad = \text{ [induction hypothesis]} \\
\mathcal{CV}(e_1, u) \cup \mathcal{CV}(e_2, u) \\
\quad = \\
\mathcal{CV}(e_1\ e_2, u)
\end{array}
$$

$e_1 \equiv e_2$, $\mathsf{if}\ e_0\ e_1\ e_2$)
Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)

$$
\begin{array}{l}
\mathcal{CV}((\lambda v{:}T.\ e)[\sigma], u) \\
\quad = \\
\mathcal{CV}(\lambda v{:}T[\sigma].\ e[\sigma], u) \\
\quad = \\
\begin{cases} \{v\} \cup \mathcal{CV}(e[\sigma], u) & \text{if} \quad u \in \mathcal{FV}(e[\sigma]) - \{v\} \\ \emptyset & \text{otherwise} \end{cases} \\
\quad = \text{ [Theorem 4.19]} \\
\begin{cases} \{v\} \cup \mathcal{CV}(e[\sigma], u) & \text{if} \quad u \in \mathcal{FV}(e) - \{v\} \\ \emptyset & \text{otherwise} \end{cases} \\
\quad = \text{ [induction hypothesis]}
\end{array}
$$

$$\begin{cases} \{v\} \cup \mathcal{CV}(e,u) & \text{if} \quad u \in \mathcal{FV}(e) - \{v\} \\ \emptyset & \text{otherwise} \end{cases}$$
$$=$$
$$\mathcal{CV}(\lambda v{:}T.\ e, u)$$

Proof of Theorem 4.21

By induction on e:

$o[\overline{T}]$, ι_T, proj f)

1. $\mathcal{FV}(e) = \emptyset$
2. $u \notin \mathcal{FV}(e)$ [1]
3. $e[u/d] = e$ [Theorem 4.1, 2]
4. $\mathcal{FTV}(e[u/d]) = \mathcal{FTV}(e) \cup \begin{cases} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(e) \\ \emptyset & \text{otherwise} \end{cases}$ [3, 2]

u)

$$\begin{array}{l} \mathcal{FTV}(u[u/d]) \\ \quad = \\ \mathcal{FTV}(d) \\ \quad = \ [\mathcal{FTV}(u) = \emptyset] \\ \mathcal{FTV}(u) \cup \mathcal{FTV}(d) \\ \quad = \ [u \in \{u\}] \\ \mathcal{FTV}(u) \cup \begin{cases} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(u) \\ \emptyset & \text{otherwise} \end{cases} \end{array}$$

$v \neq u$)

$$\begin{array}{l} \mathcal{FTV}(v[u/d]) \\ \quad = \\ \mathcal{FTV}(v) \\ \quad = \\ \emptyset \\ \quad = \ [\mathcal{FTV}(v) = \emptyset \ \wedge \ u \notin \{v\}] \\ \mathcal{FTV}(v) \cup \begin{cases} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(v) \\ \emptyset & \text{otherwise} \end{cases} \end{array}$$

$e_1\ e_2$)

$$\begin{array}{l} \mathcal{FTV}((e_1\ e_2)[u/d]) \\ \quad = \\ \mathcal{FTV}(e_1[u/d]) \cup \mathcal{FTV}(e_2[u/d]) \\ \quad = \ [\text{induction hypothesis}] \\ \mathcal{FTV}(e_1) \cup \begin{cases} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(e_1) \\ \emptyset & \text{otherwise} \end{cases} \cup \mathcal{FTV}(e_2) \cup \begin{cases} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(e_2) \\ \emptyset & \text{otherwise} \end{cases} \\ \quad = \\ \mathcal{FTV}(e_1\ e_2) \cup \begin{cases} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(e_1) \ \vee \ u \in \mathcal{FV}(e_2) \\ \emptyset & \text{otherwise} \end{cases} \\ \quad = \\ \mathcal{FTV}(e_1\ e_2) \cup \begin{cases} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(e_1\ e_2) \\ \emptyset & \text{otherwise} \end{cases} \end{array}$$

$e_1 \equiv e_2$, if $e_0\ e_1\ e_2$)

Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)

$$
\begin{array}{l}
\mathcal{FTV}((\lambda v{:}T.\ e)[u/d]) \\
\quad = \\
\mathcal{FTV}(\left\{\begin{array}{ll} \lambda v{:}T.\ e & \text{if} \quad u = v \\ \lambda v{:}T.\ e[u/d] & \text{otherwise} \end{array}\right.) \\
\quad = \\
\left\{\begin{array}{ll} \mathcal{FTV}(\lambda v{:}T.\ e) & \text{if} \quad u = v \\ \mathcal{FTV}(\lambda v{:}T.\ e[u/d]) & \text{otherwise} \end{array}\right. \\
\quad = \ [u = v \ \Rightarrow \ u \notin \mathcal{FV}(\lambda v{:}T.\ e)] \\
\left\{\begin{array}{ll} \mathcal{FTV}(\lambda v{:}T.\ e) \cup \left\{\begin{array}{ll} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(\lambda v{:}T.\ e) \\ \emptyset & \text{otherwise} \end{array}\right. & \text{if} \quad u = v \\ \mathcal{FTV}(T) \cup \mathcal{FTV}(e[u/d]) & \text{otherwise} \end{array}\right. \\
\quad = \ [\text{induction hypothesis}] \\
\left\{\begin{array}{ll} \mathcal{FTV}(\lambda v{:}T.\ e) \cup \left\{\begin{array}{ll} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(\lambda v{:}T.\ e) \\ \emptyset & \text{otherwise} \end{array}\right. & \text{if} \quad u = v \\ \mathcal{FTV}(T) \cup \mathcal{FTV}(e) \cup \left\{\begin{array}{ll} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(e) \\ \emptyset & \text{otherwise} \end{array}\right. & \text{otherwise} \end{array}\right. \\
\quad = \ [\mathcal{FTV}(\lambda v{:}T.\ e) = \mathcal{FTV}(T) \cup \mathcal{FTV}(e)] \\
\left\{\begin{array}{ll} \mathcal{FTV}(\lambda v{:}T.\ e) \cup \left\{\begin{array}{ll} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(\lambda v{:}T.\ e) \\ \emptyset & \text{otherwise} \end{array}\right. & \text{if} \quad u = v \\ \mathcal{FTV}(\lambda v{:}T.\ e) \cup \left\{\begin{array}{ll} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(e) \\ \emptyset & \text{otherwise} \end{array}\right. & \text{otherwise} \end{array}\right. \\
\quad = \ [u \neq v \ \wedge \ u \in \mathcal{FV}(e) \ \Rightarrow \ u \in \mathcal{FV}(\lambda v{:}T.\ e)] \\
\left\{\begin{array}{ll} \mathcal{FTV}(\lambda v{:}T.\ e) \cup \left\{\begin{array}{ll} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(\lambda v{:}T.\ e) \\ \emptyset & \text{otherwise} \end{array}\right. & \text{if} \quad u = v \\ \mathcal{FTV}(\lambda v{:}T.\ e) \cup \left\{\begin{array}{ll} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(\lambda v{:}T.\ e) \\ \emptyset & \text{otherwise} \end{array}\right. & \text{otherwise} \end{array}\right. \\
\quad = \\
\mathcal{FTV}(\lambda v{:}T.\ e) \cup \left\{\begin{array}{ll} \mathcal{FTV}(d) & \text{if} \quad u \in \mathcal{FV}(\lambda v{:}T.\ e) \\ \emptyset & \text{otherwise} \end{array}\right.
\end{array}
$$

Proof of Theorem 4.22

$$
\begin{array}{l}
\mathcal{FTV}(e[u/u']) \\
\quad = \ [\text{Theorem 4.21}] \\
\mathcal{FTV}(e) \cup \left\{\begin{array}{ll} \mathcal{FTV}(u') & \text{if} \quad u \in \mathcal{FV}(e) \\ \emptyset & \text{otherwise} \end{array}\right. \\
\quad = \ [\mathcal{FTV}(u') = \emptyset] \\
\mathcal{FTV}(e)
\end{array}
$$

Proof of Theorem 4.23

By induction on e:

u)

$$
\begin{array}{l}
u[u/d][\sigma] \\
\quad = \\
d[\sigma] \\
\quad = \\
u[\sigma][u/d[\sigma]]
\end{array}
$$

$v \neq u$)

$$
\begin{array}{l}
v[u/d][\sigma] \\
\quad = \\
v \\
\quad = \\
v[\sigma][u/d[\sigma]]
\end{array}
$$

$o[\overline{T}]$, ι_T, $\mathsf{proj}\ f$)

$$
\begin{array}{l}
e[u/d][\sigma] \\
\quad = \text{ [Theorem 4.1]} \\
e[\sigma] \\
\quad = \text{ [Theorem 4.1, Theorem 4.19]} \\
e[\sigma][u/d[\sigma]]
\end{array}
$$

$e_1\ e_2$)

$$
\begin{array}{l}
(e_1\ e_2)[u/d][\sigma] \\
\quad = \\
e_1[u/d][\sigma]\ e_2[u/d][\sigma] \\
\quad = \text{ [induction hypothesis]} \\
e_1[\sigma][u/d[\sigma]]\ e_2[\sigma][u/d[\sigma]] \\
\quad = \\
(e_1\ e_2)[\sigma][u/d[\sigma]]
\end{array}
$$

$e_1 \equiv e_2$, $\mathsf{if}\ e_0\ e_1\ e_2$)
Analogous to $e_1\ e_2$.

$\lambda v{:}T.\ e$)

$$
\begin{array}{l}
(\lambda v{:}T.\ e)[u/d][\sigma] \\
\quad = \\
\begin{cases} \lambda v{:}T[\sigma].\ e[\sigma] & \text{if} \quad u = v \\ \lambda v{:}T[\sigma].\ e[u/d][\sigma] & \text{otherwise} \end{cases} \\
\quad = \text{ [induction hypothesis]} \\
\begin{cases} \lambda v{:}T[\sigma].\ e[\sigma] & \text{if} \quad u = v \\ \lambda v{:}T[\sigma].\ e[\sigma][u/d[\sigma]] & \text{otherwise} \end{cases} \\
\quad = \\
(\lambda v{:}T.\ e)[\sigma][u/d[\sigma]]
\end{array}
$$

Proof of Theorem 4.24

$$
\begin{array}{l}
e[u/u'][\sigma] \\
\quad = \text{ [Theorem 4.23]} \\
e[\sigma][u/u'[\sigma]] \\
\quad = \\
e[\sigma][u/u']
\end{array}
$$

Proof of Theorem 4.25

Immediate by inspection of the definition of expression substitution in context elements and contexts.

Proof of Theorem 4.26

By induction on x:

β)

$$\beta[\emptyset] = \beta \qquad \text{[definition of } \_[\_],\ \beta \notin \mathcal{D}(\emptyset)]$$

Bool, v, $\mathsf{ty}\ \tau{:}n$, $\mathsf{tvar}\ \beta$, ϵ)

$$x[\emptyset] = x \qquad \text{[definition of } \_[\_]]$$

$T_1 \to T_2$)

$(T_1 \to T_2)[\emptyset]$
$\quad =$
$T_1[\emptyset] \to T_2[\emptyset]$
$\quad =$ [induction hypothesis]
$T_1 \to T_2$

$\mathsf{op}\ o{:}\{\overline{\beta}\}\ T)$

$(\mathsf{op}\ o{:}\{\overline{\beta}\}\ T)[\emptyset]$
$\quad =$
$\mathsf{op}\ o{:}\{\overline{\beta}\}\ T[\emptyset\!\uparrow_{\overline{\beta}}]$
$\quad =$ $[\emptyset\!\uparrow_{\overline{\beta}} = \emptyset]$
$\mathsf{op}\ o{:}\{\overline{\beta}\}\ T[\emptyset]$
$\quad =$ [induction hypothesis]
$\mathsf{op}\ o{:}\{\overline{\beta}\}\ T$

$\mathsf{ax}\ \{\overline{\beta}\}\ e)$
Analogous to $\mathsf{op}\ o{:}\{\overline{\beta}\}\ T$.

all other kinds)
Analogous to $T_1 \to T_2$.

Proof of Theorem 4.27

By induction on x:

$\beta)$
If $\beta \notin \mathcal{D}(\sigma)$:

$\beta[\sigma]$
$\quad =$
β
$\quad =$ $[\beta \notin \mathcal{D}(\emptyset)]$
$\beta[\emptyset]$
$\quad =$ [hypothesis $\beta \notin \mathcal{D}(\sigma)$]
$\beta[\sigma\!\downarrow_{\{\beta\}}]$
$\quad =$ $[\mathcal{FTV}(\beta) = \{\beta\}]$
$\beta[\sigma\!\downarrow_{\mathcal{FTV}(\beta)}]$

If $\beta \in \mathcal{D}(\sigma)$:

$\beta[\sigma]$
$\quad =$
$\sigma(\beta)$
$\quad =$
$\{\langle\beta, \sigma(\beta)\rangle\}(\beta)$
$\quad =$ [hypothesis $\beta \in \mathcal{D}(\sigma)$]
$\sigma\!\downarrow_{\{\beta\}}(\beta)$
$\quad =$ [hypothesis $\beta \in \mathcal{D}(\sigma)$]
$\beta[\sigma\!\downarrow_{\{\beta\}}]$
$\quad =$ $[\mathcal{FTV}(\beta) = \{\beta\}]$
$\beta[\sigma\!\downarrow_{\mathcal{FTV}(\beta)}]$

Bool, v, $\mathsf{ty}\ \tau{:}n$, $\mathsf{tvar}\ \beta$, $\epsilon)$

$x[\sigma]$
$\quad =$

x
$\quad =$
$x[\emptyset]$
$\quad = \ [\mathcal{FTV}(x) = \emptyset]$
$x[\sigma\downarrow_{\mathcal{FTV}(x)}]$

$T_1 \to T_2$)

$(T_1 \to T_2)[\sigma]$
$\quad =$
$T_1[\sigma] \to T_2[\sigma]$
$\quad =$ [induction hypothesis]
$T_1[\sigma\downarrow_{\mathcal{FTV}(T_1)}] \to T_2[\sigma\downarrow_{\mathcal{FTV}(T_2)}]$
$\quad = \ [\mathcal{FTV}(T_1) \subseteq \mathcal{FTV}(T_1 \to T_2) \ \wedge \ \mathcal{FTV}(T_2) \subseteq \mathcal{FTV}(T_1 \to T_2)]$
$T_1[\sigma\downarrow_{\mathcal{FTV}(T_1\to T_2)}\downarrow_{\mathcal{FTV}(T_1)}] \to T_2[\sigma\downarrow_{\mathcal{FTV}(T_1\to T_2)}\downarrow_{\mathcal{FTV}(T_2)}]$
$\quad =$ [induction hypothesis]
$T_1[\sigma\downarrow_{\mathcal{FTV}(T_1\to T_2)}] \to T_2[\sigma\downarrow_{\mathcal{FTV}(T_1\to T_2)}]$
$\quad =$
$(T_1 \to T_2)[\sigma\downarrow_{\mathcal{FTV}(T_1\to T_2)}]$

$\mathsf{op}\ o\colon\{\overline{\beta}\}\ T$)

$(\mathsf{op}\ o\colon\{\overline{\beta}\}\ T)[\sigma]$
$\quad =$
$\mathsf{op}\ o\colon\{\overline{\beta}\}\ T[\sigma\uparrow_{\overline{\beta}}]$
$\quad =$ [induction hypothesis]
$\mathsf{op}\ o\colon\{\overline{\beta}\}\ T[\sigma\uparrow_{\overline{\beta}}\downarrow_{\mathcal{FTV}(T)}]$
$\quad = \ [(\mathcal{D}(\sigma) - \overline{\beta}) \cap \mathcal{FTV}(T) = (\mathcal{D}(\sigma) \cap \mathcal{FTV}(T)) - \overline{\beta}]$
$\mathsf{op}\ o\colon\{\overline{\beta}\}\ T[\sigma\downarrow_{\mathcal{FTV}(T)}\uparrow_{\overline{\beta}}]$
$\quad = \ [(\mathcal{D}(\sigma) \cap \mathcal{FTV}(T)) - \overline{\beta} = (\mathcal{D}(\sigma) \cap (\mathcal{FTV}(T) - \overline{\beta})) - \overline{\beta}]$
$\mathsf{op}\ o\colon\{\overline{\beta}\}\ T[\sigma\downarrow_{\mathcal{FTV}(T)-\overline{\beta}}\uparrow_{\overline{\beta}}]$
$\quad = \ [\mathcal{FTV}(\mathsf{op}\ o\colon\{\overline{\beta}\}\ T) = \mathcal{FTV}(T) - \overline{\beta}]$
$(\mathsf{op}\ o\colon\{\overline{\beta}\}\ T)[\sigma\downarrow_{\mathcal{FTV}(\mathsf{op}\ \alpha\{\overline{\beta}\}\ T)}\uparrow_{\overline{\beta}}]$
$\quad =$
$(\mathsf{op}\ o\colon\{\overline{\beta}\}\ T)[\sigma\downarrow_{\mathcal{FTV}(\mathsf{op}\ \alpha\{\overline{\beta}\}\ T)}]$

$\mathsf{ax}\ \{\overline{\beta}\}\ e$)
Analogous to $\mathsf{op}\ o\colon\{\overline{\beta}\}\ T$.

all other kinds)
Analogous to $T_1 \to T_2$.

Proof of Theorem 4.28

$x[\sigma]$
$\quad =$ [Theorem 4.27]
$x[\sigma\downarrow_{\mathcal{FTV}(x)}]$
$\quad =$ [hypothesis $\mathcal{D}(\sigma) \cap \mathcal{FTV}(x) = \emptyset$]
$x[\emptyset]$
$\quad =$ [Theorem 4.26]
x

Proof of Theorem 4.29

By induction on x:

Bool, v)

$$\begin{aligned}
&\mathcal{FTV}(x[\sigma]) \\
&\quad = \\
&\mathcal{FTV}(x) \\
&\quad = \\
&\emptyset \\
&\quad = \\
&(\emptyset - \mathcal{D}(\sigma)) \cup \textstyle\bigcup_{\alpha \in \emptyset \cap \mathcal{D}(\sigma)} \mathcal{FTV}(\sigma(\alpha)) \\
&\quad = \ [\mathcal{FTV}(x) = \emptyset] \\
&(\mathcal{FTV}(x) - \mathcal{D}(\sigma)) \cup \textstyle\bigcup_{\alpha \in \mathcal{FTV}(x) \cap \mathcal{D}(\sigma)} \mathcal{FTV}(\sigma(\alpha))
\end{aligned}$$

β)

If $\beta \notin \mathcal{D}(\sigma)$:

$$\begin{aligned}
&\mathcal{FTV}(\beta[\sigma]) \\
&\quad = \\
&\mathcal{FTV}(\beta) \\
&\quad = \\
&\{\beta\} \\
&\quad = \ [\beta \notin \mathcal{D}(\sigma)] \\
&(\{\beta\} - \mathcal{D}(\sigma)) \cup \textstyle\bigcup_{\alpha \in \{\beta\} \cap \mathcal{D}(\sigma)} \mathcal{FTV}(\sigma(\alpha)) \\
&\quad = \ [\mathcal{FTV}(\beta) = \{\beta\}] \\
&(\mathcal{FTV}(\beta) - \mathcal{D}(\sigma)) \cup \textstyle\bigcup_{\alpha \in \mathcal{FTV}(\beta) \cap \mathcal{D}(\sigma)} \mathcal{FTV}(\sigma(\alpha))
\end{aligned}$$

If $\beta \in \mathcal{D}(\sigma)$:

$$\begin{aligned}
&\mathcal{FTV}(\beta[\sigma]) \\
&\quad = \\
&\mathcal{FTV}(\sigma(\beta)) \\
&\quad = \ [\beta \in \mathcal{D}(\sigma) \ \Rightarrow \ \mathcal{FTV}(\beta) - \mathcal{D}(\sigma) = \{\beta\} - \mathcal{D}(\sigma) = \emptyset] \\
&(\mathcal{FTV}(\beta) - \mathcal{D}(\sigma)) \cup \mathcal{FTV}(\sigma(\beta)) \\
&\quad = \\
&(\mathcal{FTV}(\beta) - \mathcal{D}(\sigma)) \cup \textstyle\bigcup_{\alpha \in \{\beta\}} \mathcal{FTV}(\sigma(\alpha)) \\
&\quad = \ [\mathcal{FTV}(\beta) = \{\beta\} \ \wedge \ \beta \in \mathcal{D}(\sigma)] \\
&(\mathcal{FTV}(\beta) - \mathcal{D}(\sigma)) \cup \textstyle\bigcup_{\alpha \in \mathcal{FTV}(\beta) \cap \mathcal{D}(\sigma)} \mathcal{FTV}(\sigma(\alpha))
\end{aligned}$$

$T_1 \to T_2$)

$$\begin{aligned}
&\mathcal{FTV}((T_1 \to T_2)[\sigma]) \\
&\quad = \\
&\mathcal{FTV}(T_1[\sigma]) \cup \mathcal{FTV}(T_2[\sigma]) \\
&\quad = \ [\text{induction hypothesis}] \\
&(\mathcal{FTV}(T_1) - \mathcal{D}(\sigma)) \cup \textstyle\bigcup_{\alpha \in \mathcal{FTV}(T_1) \cap \mathcal{D}(\sigma)} \mathcal{FTV}(\sigma(\alpha)) \cup \\
&(\mathcal{FTV}(T_2) - \mathcal{D}(\sigma)) \cup \textstyle\bigcup_{\alpha \in \mathcal{FTV}(T_2) \cap \mathcal{D}(\sigma)} \mathcal{FTV}(\sigma(\alpha)) \\
&\quad = \\
&((\mathcal{FTV}(T_1) \cup \mathcal{FTV}(T_2)) - \mathcal{D}(\sigma)) \cup \\
&\textstyle\bigcup_{\alpha \in (\mathcal{FTV}(T_1) \cap \mathcal{D}(\sigma)) \cup (\mathcal{FTV}(T_1) \cap \mathcal{D}(\sigma))} \mathcal{FTV}(\sigma(\alpha)) \\
&\quad = \\
&(\mathcal{FTV}(T_1 \to T_2) - \mathcal{D}(\sigma)) \cup \textstyle\bigcup_{\alpha \in (\mathcal{FTV}(T_1) \cup \mathcal{FTV}(T_2)) \cap \mathcal{D}(\sigma)} \mathcal{FTV}(\sigma(\alpha)) \\
&\quad = \\
&(\mathcal{FTV}(T_1 \to T_2) - \mathcal{D}(\sigma)) \cup \textstyle\bigcup_{\alpha \in \mathcal{FTV}(T_1 \to T_2) \cap \mathcal{D}(\sigma)} \sigma(\alpha)
\end{aligned}$$

all other kinds)

Analogous to $T_1 \to T_2$.

Proof of Theorem 4.30

By induction on x:

Bool, v)

$$x[\sigma][\sigma'] = x = x[\sigma[\sigma']]$$

β)

If $\beta \notin \mathcal{D}(\sigma)$:

1. $\beta[\sigma][\sigma'] = \beta[\sigma']$
2. $(\mathcal{FTV}(\beta) - \mathcal{D}(\sigma)) \cap \mathcal{D}(\sigma') = \emptyset$ [hypothesis]
3. $\beta \notin \mathcal{D}(\sigma')$ [2, hypothesis $\beta \notin \mathcal{D}(\sigma)$]
4. $\beta[\sigma'] = \beta$ [3]
5. $\mathcal{D}(\sigma[\sigma']) = \mathcal{D}(\sigma)$ [definition of $\_[\_]$]
6. $\beta \notin \mathcal{D}(\sigma[\sigma'])$ [hypothesis $\beta \notin \mathcal{D}(\sigma)$, 5]
7. $\beta[\sigma[\sigma']] = \beta$ [6]
8. $\beta[\sigma][\sigma'] = \beta[\sigma[\sigma']]$ [1, 4, 7]

If $\beta \in \mathcal{D}(\sigma)$:

$\beta[\sigma][\sigma']$
$\quad =$
$\sigma(\beta)[\sigma']$
$\quad =$ [definition of $\sigma[\sigma']$]
$\sigma[\sigma'](\beta)$
$\quad =$ [$\mathcal{D}(\sigma[\sigma']) = \mathcal{D}(\sigma)$]
$\beta[\sigma[\sigma']]$

$T_1 \to T_2$)

1. $\mathcal{FTV}(T_1 \to T_2) = \mathcal{FTV}(T_1) \cup \mathcal{FTV}(T_2)$
2. $(\mathcal{FTV}(T_1 \to T_2) - \mathcal{D}(\sigma)) \cap \mathcal{D}(\sigma') = \emptyset$ [hypothesis]
3. $(\mathcal{FTV}(T_1) - \mathcal{D}(\sigma)) \cap \mathcal{D}(\sigma') = (\mathcal{FTV}(T_2) - \mathcal{D}(\sigma)) \cap \mathcal{D}(\sigma') = \emptyset$ [1, 2]
4. $(T_1 \to T_2)[\sigma][\sigma']$
   $\quad =$
   $T_1[\sigma][\sigma'] \to T_2[\sigma][\sigma']$
   $\quad =$ [induction hypothesis, 3]
   $T_1[\sigma[\sigma']] \to T_2[\sigma[\sigma']]$
   $\quad =$
   $(T_1 \to T_2)[\sigma[\sigma']]$

all other kinds)

Analogous to $T_1 \to T_2$.

Proof of Theorem 4.31

By induction on x:

Bool, v)

$$x[\sigma][\sigma'] = x[\sigma'][\sigma[\sigma']]$$

β)

If $\beta \notin \mathcal{D}(\sigma) \ \wedge \ \beta \notin \mathcal{D}(\sigma')$:

$\beta[\sigma][\sigma']$
$\quad =$
β
$\quad =$ [$\mathcal{D}(\sigma[\sigma']) = \mathcal{D}(\sigma)$]
$\beta[\sigma'][\sigma[\sigma']]$

If $\beta \in \mathcal{D}(\sigma)$:

1. $\beta \notin \mathcal{D}(\sigma')$ [hypothesis $\mathcal{D}(\sigma) \cap \mathcal{D}(\sigma') = \emptyset$]
2. $\beta[\sigma][\sigma']$
   $\quad =$ [hypothesis $\beta \in \mathcal{D}(\sigma)$]
   $\sigma(\beta)[\sigma']$
   $\quad =$ [definition of $\sigma[\sigma']$]
   $\sigma[\sigma'](\beta)$
   $\quad =$ [$\mathcal{D}(\sigma[\sigma']) = \mathcal{D}(\sigma)$]
   $\beta[\sigma[\sigma']]$
   $\quad =$ [1]
   $\beta[\sigma'][\sigma[\sigma']]$

If $\beta \in \mathcal{D}(\sigma')$:

1. $\beta \notin \mathcal{D}(\sigma)$ [hypothesis $\mathcal{D}(\sigma) \cap \mathcal{D}(\sigma') = \emptyset$]
2. $\forall \alpha \in \mathcal{FTV}(\beta) \cap \mathcal{D}(\sigma').\ \ \mathcal{FTV}(\sigma'(\alpha)) \cap \mathcal{D}(\sigma) = \emptyset$ [hypothesis]
3. $\mathcal{FTV}(\sigma'(\beta)) \cap \mathcal{D}(\sigma) = \emptyset$ [hypothesis $\beta \in \mathcal{D}(\sigma')$, 2]
4. $\beta[\sigma][\sigma']$
   $\quad =$ [1]
   $\beta[\sigma']$
   $\quad =$ [hypothesis $\beta \in \mathcal{D}(\sigma')$]
   $\sigma'(\beta)$
   $\quad =$ [3, $\mathcal{D}(\sigma[\sigma']) = \mathcal{D}(\sigma)$, Theorem 4.28]
   $\sigma'(\beta)[\sigma[\sigma']]$
   $\quad =$ [hypothesis $\beta \in \mathcal{D}(\sigma')$]
   $\beta[\sigma'][\sigma[\sigma']]$

$T_1 \to T_2$)

1. $\mathcal{FTV}(T_1 \to T_2) = \mathcal{FTV}(T_1) \cup \mathcal{FTV}(T_2)$
2. $\forall \alpha \in \mathcal{FTV}(T_1 \to T_2) \cap \mathcal{D}(\sigma').\ \ \mathcal{FTV}(\sigma'(\alpha)) \cap \mathcal{D}(\sigma) = \emptyset$ [hypothesis]
3. $\forall \alpha \in \mathcal{FTV}(T_1) \cap \mathcal{D}(\sigma').\ \ \mathcal{FTV}(\sigma'(\alpha)) \cap \mathcal{D}(\sigma) = \emptyset$ [1, 2]
4. $\forall \alpha \in \mathcal{FTV}(T_2) \cap \mathcal{D}(\sigma').\ \ \mathcal{FTV}(\sigma'(\alpha)) \cap \mathcal{D}(\sigma) = \emptyset$ [1, 2]
5. $(T_1 \to T_2)[\sigma][\sigma']$
   $\quad =$
   $T_1[\sigma][\sigma'] \to T_2[\sigma][\sigma']$
   $\quad =$ [induction hypothesis, 3, 4]
   $T_1[\sigma'][\sigma[\sigma']] \to T_2[\sigma'][\sigma[\sigma']]$
   $\quad =$
   $(T_1 \to T_2)[\sigma'][\sigma[\sigma']]$

all other kinds)
Analogous to $T_1 \to T_2$.

Proof of Theorem 4.32

By straightforward calculation, and by induction on n for $\forall v_1 : T_1, \ldots, v_n : T_n.\ e$ and for $\exists v_1 : T_1, \ldots, v_n : T_n.\ e$.

Proof of Theorem 4.33

By straightforward calculation, and by induction on n for $\forall v_1{:}T_1, \ldots, v_n{:}T_n.\ e$ and for $\exists v_1{:}T_1, \ldots, v_n{:}T_n.\ e$. Theorem 4.1 can be readily used for the expressions that do not have free variables.

Proof of Theorem 4.34

By straightforward calculation, and by induction on n for $\forall v_1{:}T_1, \ldots, v_n{:}T_n.\ e$ and for $\exists v_1{:}T_1, \ldots, v_n{:}T_n.\ e$ (using Theorem 4.1 for the base cases $\forall \epsilon.\ e$ and $\exists \epsilon.\ e$, which both stand for e, when $u \notin \mathcal{FV}(e)$). Theorem 4.1 can be readily used for the expressions that do not have free variables.

Proof of Theorem 4.35

By straightforward calculation, and by induction on n for $\forall v_1{:}T_1, \ldots, v_n{:}T_n.\ e$ and for $\exists v_1{:}T_1, \ldots, v_n{:}T_n.\ e$. Recall that a projection $e.f$ is implicitly decorated by a record type $\prod_i f_i\ T_i$.

Proof of Theorem 4.36

By straightforward calculation, and by induction on n for $\forall v_1{:}T_1, \ldots, v_n{:}T_n.\ e$ and for $\exists v_1{:}T_1, \ldots, v_n{:}T_n.\ e$. Recall that a projection $e.f$ is implicitly decorated by a record type $\prod_i f_i\ T_i$.

Proof of Theorem 4.37

By induction on the derivation of the judgements in the antecedents of the implications:

cxMT)
: We have $cx_1, cx_2 = cx_1 = cx_2 = \epsilon$ and $\vdash \epsilon : \textsc{context}$ by cxMT.

cxTdec, cxTVdec, cxOdec, cxVdec, cxAx)
: If $cx_2 = \epsilon$, the (first) implication is trivially proved. If instead $cx_2 \neq \epsilon$, we have $cx_2 = cx_2', cxel$ for some cx_2' and $cxel$ and the rule has either a premise of the form $\vdash cx_1, cx_2' : \textsc{context}$ or a premise of the form $cx_1, cx_2', cx_3 \vdash \ldots$ for some cx_3 (possibly $cx_3 = \epsilon$); thus, by induction hypothesis we have $\vdash cx_1 : \textsc{context}$.

all other rules)
: The rule has either a premise of the form $\vdash cx_1, cx_2 : \textsc{context}$ or a premise of the form $cx_1, cx_2, cx_3 \vdash \ldots$ for some cx_3 (possibly $cx_3 = \epsilon$). Thus, by induction hypothesis we have $\vdash cx_1 : \textsc{context}$.

Proof of Theorem 4.38

1. $\vdash cx_1, \mathsf{ty}\ \tau{:}n, cx_2 : \textsc{context}$ [hypothesis]
2. $\vdash cx_1, \mathsf{ty}\ \tau{:}n : \textsc{context}$ [Theorem 4.37, 1]
3. $\tau \notin \mathcal{TN}(cx_1)$ [premise of cxTdec, only rule that can derive 2]

Consider $\tau' \in \mathcal{TN}(cx_2)$:

4. $\exists cx_2', cx_2'', n'.\ \ cx_2 = cx_2', \mathsf{ty}\ \tau'{:}n', cx_2''$
5. $\vdash cx_1, \mathsf{ty}\ \tau{:}n, cx_2', \mathsf{ty}\ \tau'{:}n' : \textsc{context}$ [1, 4, Theorem 4.37]
6. $\tau' \notin \mathcal{TN}(cx_1, \mathsf{ty}\ \tau{:}n, cx_2')$ [premise of cxTdec, only rule that can derive 5]
7. $\mathcal{TN}(cx_1, \mathsf{ty}\ \tau{:}n, cx_2') = \mathcal{TN}(cx_1) \cup \{\tau\} \cup \mathcal{TN}(cx_2')$
8. $\tau \neq \tau'$ [6, 7]

Since τ' is arbitrary:

9. $\forall \tau' \in \mathcal{TN}(cx_2).\ \ \tau \neq \tau'$
10. $\tau \notin \mathcal{TN}(cx_2)$ [9]
11. $\tau \notin \mathcal{TN}(cx_1) \cup \mathcal{TN}(cx_2)$ [3, 10]

Proof of Theorem 4.39

Analogous to Theorem 4.38, using cxOdec and $\mathcal{ON}$ instead of cxTdec and $\mathcal{TN}$.

Proof of Theorem 4.40

Analogous to Theorem 4.38, using cxTVdec and $\mathcal{TV}$ instead of cxTdec and $\mathcal{TN}$.

Proof of Theorem 4.41

Analogous to Theorem 4.38, using cxVdec and $\mathcal{V}$ instead of cxTdec and $\mathcal{TN}$.

Proof of Theorem 4.42

1. $\vdash cx_1, \mathsf{op}\ o{:}\{\overline{\beta}\}\ T, cx_2 : \textsc{context}$ [hypothesis]
2. $\vdash cx_1, \mathsf{op}\ o{:}\{\overline{\beta}\}\ T : \textsc{context}$ [Theorem 4.37, 1]
3. $cx_1, \mathsf{tvar}\ \overline{\beta} \vdash T : \textsc{type}$ [premise of cxOdec, only rule that can derive 2]

Proof of Theorem 4.43

Analogous to Theorem 4.42, using cxAx instead of cxOdec.

Proof of Theorem 4.44

Analogous to Theorem 4.42, using cxVdec instead of cxOdec.

Proof of Theorem 4.45

The only rule that can derive a judgement of the form $cx \vdash \beta : \textsc{type}$ is tyVar, which has $\beta \in \mathcal{TV}(cx)$ as its premise.

Proof of Theorem 4.46

The only rule that can derive a judgement of the form $cx \vdash \tau[\overline{T}] : \textsc{type}$ is tyInst, which has $\mathsf{ty}\ \tau{:}n \in cx$, with $n = |\overline{T}|$, as well as $\forall i.\ \ cx \vdash T_i : \textsc{type}$ as two of its premises.

Proof of Theorem 4.47

The only rule that can derive a judgement of the form $cx \vdash T_1 \to T_2 : \textsc{type}$ is tyArr, which has $cx, \mathsf{var}\ v : T_1 \vdash T_2 : \textsc{type}$ as its premise, which immediately proves the second conjunct. The first conjunct is proved as follows:

1. $cx, \mathsf{var}\ v{:}T_1 \vdash T_2 : \textsc{type}$ [premise of tyArr]
2. $\vdash cx, \mathsf{var}\ v{:}T_1 : \textsc{context}$ [Theorem 4.37, 1]
3. $cx \vdash T_1 : \textsc{type}$ [Theorem 4.44, 2]

Proof of Theorem 4.48

The only rule that can derive a judgement of the form $cx \vdash \prod_i f_i\ T_i : \text{TYPE}$ is tyRec, which has $\forall i.\ \ cx \vdash T_i : \text{TYPE}$ as one of its premises.

Proof of Theorem 4.49

The only rule that can derive a judgement of the form $cx \vdash T|r : \text{TYPE}$ is tyRestr, which has $cx \vdash r : T \to \mathsf{Bool}$ as one of its premises.

Proof of Theorem 4.50

The only rule that can derive a judgement of the form $cx \vdash T|r : \text{TYPE}$ is tyRestr, which has $\mathcal{FV}(r) = \emptyset$ as one of its premises.

Proof of Theorem 4.51

The only rules that can derive a judgement of the form $(cx \vdash e_1\ e_2 : \ldots)$ are exApp, exSuper, and exSub. For exApp, the theorem is readily proved from the premises of the rule. For exSuper and exSub, the theorem is proved by induction on the premise that has also the form $(cx \vdash e_1\ e_2 : \ldots)$.

Proof of Theorem 4.52

The only rules that can derive a judgement of the form $(cx \vdash \lambda v : T.\ e : \ldots)$ are exAbs, exSuper, exSub, and exAbsAlpha. For exAbs, the theorem is readily proved from the conclusion of the rule. For exSuper and exSub, the theorem is proved by induction on the premise that has also the form $(cx \vdash \lambda v : T.\ e : \ldots)$. For exAbsAlpha, the theorem is proved as follows:

1. $cx \vdash \lambda v : T.\ e : \ldots$ [premise of exAbsAlpha]
2. $\exists T'.\ \ cx \vdash \lambda v : T.\ e : T \to T'$ [induction hypothesis, 1]
3. $v' \notin \mathcal{FV}(e) \cup \mathcal{CV}(e, v)$ [premise of exAbsAlpha]
4. $cx \vdash \lambda v' : T.\ e[v/v'] : T \to T'$ [exAbsAlpha, 2, 3]

Proof of Theorem 4.53

By induction on the derivation of $cx \vdash T_1 \prec_r T_2$:

stRestr)

1. $cx \vdash T|r : \text{TYPE}$ [premise of rule]
2. $\mathcal{FV}(r) = \emptyset$ [Theorem 4.50, 1]

stRefl)

$\mathcal{FV}(r) = \mathcal{FV}(\lambda v : T.\ \mathsf{true}) = \emptyset$ [Theorem 4.32]

stArr)

1. $cx, \mathsf{var}\ v : T \vdash T_1 \prec_r T_2$ [premise of rule]
2. $\mathcal{FV}(r) = \emptyset$ [induction hypothesis, 1]

3. $\mathcal{FV}(r')$
   $=$
   $\mathcal{FV}(\lambda v{:}T \to T_2.\ \forall v'{:}T.\ r\ (v\ v'))$
   $=$ [Theorem 4.32]
   $\mathcal{FV}(r) - \{v, v'\}$
   $=$ [2]
   $\emptyset$

stRec)

1. $\forall i.\ \ cx \vdash T_i \prec_{r_i} T'_i$ [premise of rule]
2. $\forall i.\ \ \mathcal{FV}(r_i) = \emptyset$ [induction hypothesis, 1]
3. $\mathcal{FV}(r)$
   $=$
   $\mathcal{FV}(\lambda v{:}\prod_i f_{P(i)}\ T'_{P(i)}.\ \bigwedge_i r_i\ v.f_i)$
   $=$ [Theorem 4.32]
   $\bigcup_i \mathcal{FV}(r_i) - \{v\}$
   $=$ [2]
   $\emptyset$

Proof of Theorem 4.54

By induction on the derivation of the judgement in the antecedent of the implication:

exOp, exThe, exProj)
$\mathcal{FV}(e) = \emptyset$

exVar)

1. $\mathsf{var}\ v{:}T \in cx$ [premise of exVar]
2. $v \in \mathcal{V}(cx)$ [1]
3. $\mathcal{FV}(v) = \{v\} \subseteq \mathcal{V}(cx)$ [2]

exApp)

1. $cx \vdash e_1 : T_1 \to T_2$ [premise of exApp]
2. $cx \vdash e_2 : T_1$ [premise of exApp]
3. $\mathcal{FV}(e_1) \subseteq \mathcal{V}(cx)$ [induction hypothesis, 1]
4. $\mathcal{FV}(e_2) \subseteq \mathcal{V}(cx)$ [induction hypothesis, 2]
5. $\mathcal{FV}(e_1\ e_2) = \mathcal{FV}(e_1) \cup \mathcal{FV}(e_2) \subseteq \mathcal{V}(cx)$ [3, 4]

exEq, exIf, exSuper, exSub)
Analogous to exApp.

exAbs)

1. $cx, \mathsf{var}\ v{:}T \vdash e : T'$ [premise of exAbs]
2. $\mathcal{FV}(e) \subseteq \mathcal{V}(cx, \mathsf{var}\ v{:}T)$ [induction hypothesis, 1]
3. $\mathcal{FV}(e) \subseteq \mathcal{V}(cx) \cup \{v\}$ [2]
4. $\mathcal{FV}(e) - \{v\} \subseteq \mathcal{V}(cx) - \{v\}$ [3]
5. $\vdash cx, \mathsf{var}\ v{:}T : \text{CONTEXT}$ [Theorem 4.37, 1]
6. $v \notin \mathcal{V}(cx)$ [Theorem 4.41, 5]
7. $\mathcal{V}(cx) - \{v\} = \mathcal{V}(cx)$ [6]

8. $\mathcal{FV}(\lambda v{:}T.\ e) \subseteq \mathcal{V}(cx)$ [4, 7]

exAbsAlpha)

1. $\mathcal{FV}(\lambda v{:}T.\ e) \subseteq \mathcal{V}(cx)$ [induction hypothesis]
2. $v' \notin \mathcal{FV}(e) \cup \mathcal{CV}(e, v)$ [premise of exAbsAlpha]
3. $\mathcal{FV}(\lambda v{:}T.\ e) = \mathcal{FV}(\lambda v'{:}T.\ e[v/v'])$ [Theorem 4.18, 2]
4. $\mathcal{FV}(\lambda v'{:}T.\ e[v/v']) \subseteq \mathcal{V}(cx)$ [1, 3]

Proof of Theorem 4.55

1. $\vdash cx_1, \mathsf{ax}\ \{\overline{\beta}\}\ e, cx_2 : \text{CONTEXT}$ [hypothesis]
2. $cx_1, \mathsf{tvar}\ \overline{\beta} \vdash e : \mathsf{Bool}$ [Theorem 4.43, 1]
3. $\mathcal{FV}(e) \subseteq \mathcal{V}(cx_1)$ [Theorem 4.54, 2]

Proof of Theorem 4.56

By induction on the derivation of the judgement in the antecedent of the implication:

thAx)

1. $\mathsf{ax}\ \{\overline{\beta}\}\ e \in cx$ [premise of thAx]
2. $\exists cx_1, cx_2.\ \ cx = cx_1, \mathsf{ax}\ \{\overline{\beta}\}\ e, cx_2$ [1]
3. $\vdash cx : \text{CONTEXT}$ [premise of thAx]
4. $\mathcal{FV}(e) \subseteq \mathcal{V}(cx_1)$ [Theorem 4.55, 2, 3]
5. $\mathcal{V}(cx_1) \subseteq \mathcal{V}(cx)$ [2]
6. $\mathcal{FV}(e[\overline{\beta}/\overline{T}]) = \mathcal{FV}(e)$ [Theorem 4.19]
7. $\mathcal{FV}(e[\overline{\beta}/\overline{T}]) \subseteq \mathcal{V}(cx)$ [6, 4, 5]

thAbs)

1. $\mathcal{FV}((\lambda v{:}T.\ e)\ e' \equiv e[v/e']) = \mathcal{FV}((\lambda v{:}T.\ e)\ e') \cup \mathcal{FV}(e[v/e'])$
2. $cx \vdash (\lambda v{:}T.\ e)\ e' : \ldots$ [premise of thAbs]
3. $\mathcal{FV}((\lambda v{:}T.\ e)\ e') \subseteq \mathcal{V}(cx)$ [Theorem 4.54, 2]
4. $\mathcal{FV}(\lambda v{:}T.\ e) \subseteq \mathcal{V}(cx)$ [3]
5. $\mathcal{FV}(e) - \{v\} \subseteq \mathcal{V}(cx)$ [4]

If $v \notin \mathcal{FV}(e)$:

6. $\mathcal{FV}(e[v/e']) = \mathcal{FV}(e)$ [Theorem 4.1]
7. $\mathcal{FV}(e) \subseteq \mathcal{V}(cx)$ [5, hypothesis $v \notin \mathcal{FV}(e)$]
8. $\mathcal{FV}((\lambda v{:}T.\ e)\ e' \equiv e[v/e']) \subseteq \mathcal{V}(cx)$ [1, 3, 6, 7]

If $v \in \mathcal{FV}(e)$:

6. $OKsbs(e, v, e')$ [premise of thAbs]
7. $\mathcal{FV}(e[v/e']) = (\mathcal{FV}(e) - \{v\}) \cup \mathcal{FV}(e')$ [Theorem 4.8, 6, hypothesis $v \in \mathcal{FV}(e)$]
8. $\mathcal{FV}(e') \subseteq \mathcal{V}(cx)$ [3]
9. $\mathcal{FV}((\lambda v{:}T.\ e)\ e' \equiv e[v/e']) \subseteq \mathcal{V}(cx)$ [1, 3, 7, 8, 5]

thSub)

1. $\mathcal{FV}(r\ e) = \mathcal{FV}(r) \cup \mathcal{FV}(e)$
2. $cx \vdash e : T$ [premise of thSub]
3. $\mathcal{FV}(e) \subseteq \mathcal{V}(cx)$ [Theorem 4.54, 2]
4. $cx \vdash T \prec_r T'$ [premise of thSub]
5. $\mathcal{FV}(r) = \emptyset$ [Theorem 4.53, 4]
6. $\mathcal{FV}(r\ e) \subseteq \mathcal{V}(cx)$ [1, 3, 5]

all other rules)
The conclusion $cx \vdash e$ of some rules is such that $\mathcal{FV}(e) = \emptyset$ (proved by the definition of $\mathcal{FV}$ and in some cases by Theorem 4.32), which immediately proves $\mathcal{FV}(e) \subseteq \mathcal{V}(cx)$. The conclusion $cx \vdash e$ of the other rules is such that every variable in $\mathcal{FV}(e)$ is also in $\mathcal{FV}(e')$ for some e' that occurs in a premise $cx' \vdash e' : \ldots$ or $cx' \vdash e'$ of the rule such that $\mathcal{V}(cx') = \mathcal{V}(cx)$, which proves $\mathcal{FV}(e) \subseteq \mathcal{V}(cx)$ by Theorem 4.54 or by induction hypothesis (the only rules for which $cx' \neq cx$ are thIfSubst and thIf, where $cx' = cx, \mathsf{ax}\ \ldots$ and so clearly $\mathcal{V}(cx') = \mathcal{V}(cx)$).

Proof of Theorem 4.57

By induction on the derivation of the judgements in the antecedents of the implications:

cxMT)
This rule is impossible because $cx_1, cxel, cx_2 \neq \epsilon$.

cxTDec)
If $cx_2 \neq \epsilon$:

1. $\exists cx_2', \tau, n.\ \ cx_2 = cx_2', \mathsf{ty}\ \tau : n$
2. $\vdash cx_1, cxel, cx_2' : \text{CONTEXT}$ [premise of cxTDec, 1]
3. $\mathcal{TV}(cx_1) \supseteq \mathcal{FTV}(cxel)$ [induction hypothesis, 2]

If $cx_2 = \epsilon$:

1. $\exists \tau, n.\ \ cxel = \mathsf{ty}\ \tau : n$
2. $\mathcal{FTV}(cxel) = \emptyset$ [1]
3. $\mathcal{TV}(cx_1) \supseteq \mathcal{FTV}(cxel)$ [2]

cxTVDec)
Analogous to cxTDec.

cxODec)
If $cx_2 \neq \epsilon$:

1. $\exists cx_2', o, \overline{\beta}, T.\ \ cx_2 = cx_2', \mathsf{op}\ o : \{\overline{\beta}\}\ T$
2. $cx_1, cxel, cx_2', \mathsf{tvar}\ \overline{\beta} \vdash T : \text{TYPE}$ [premise of cxODec, 1]
3. $\vdash cx_1, cxel, cx_2', \mathsf{tvar}\ \overline{\beta} : \text{CONTEXT}$ [Theorem 4.37, 2]
4. $\mathcal{TV}(cx_1) \supseteq \mathcal{FTV}(cxel)$ [induction hypothesis, 3]

If $cx_2 = \epsilon$:

1. $\exists o, \overline{\beta}, T.\ \ cxel = \mathsf{op}\ o : \{\overline{\beta}\}\ T$
2. $\mathcal{FTV}(cxel) = \mathcal{FTV}(T) - \overline{\beta}$ [1]
3. $cx_1, \mathsf{tvar}\ \overline{\beta} \vdash T : \text{TYPE}$ [premise of cxODec]
4. $\mathcal{TV}(cx_1) \cup \overline{\beta} \supseteq \mathcal{FTV}(T)$ [induction hypothesis, 3]
5. $(\mathcal{TV}(cx_1) \cup \overline{\beta}) - \overline{\beta} \supseteq \mathcal{FTV}(T) - \overline{\beta}$ [4]

6. $\vdash cx_1, \mathsf{tvar}\ \overline{\beta} : \textsc{context}$ [Theorem 4.37, 3]
7. $\overline{\beta} \cap \mathcal{FTV}(cx_1) = \emptyset$ [Theorem 4.40, 6]
8. $(\mathcal{FTV}(cx_1) \cup \overline{\beta}) - \overline{\beta} = \mathcal{FTV}(cx_1)$ [7]
9. $\mathcal{FTV}(cx_1) \supseteq \mathcal{FTV}(cxel)$ [5, 8, 2]

cxVdec)
If $cx_2 \neq \epsilon$, the proof is analogous to cxOdec.

If $cx_2 = \epsilon$:

1. $\exists v, T.\ \ cxel = \mathsf{var}\ v{:}T$
2. $cx_1 \vdash T : \textsc{type}$ [premise of cxVdec]
3. $\mathcal{FTV}(cx_1) \supseteq \mathcal{FTV}(T)$ [induction hypothesis, 2]
4. $\mathcal{FTV}(cx_1) \supseteq \mathcal{FTV}(cxel)$ [3, 1]

cxAx)
Analogous to cxOdec.

tyBool)

$\mathcal{FTV}(\mathsf{Bool}) = \emptyset$ [definition of $\mathcal{FTV}$]

tyVar)

1. $\beta \in \mathcal{TV}(cx)$ [premise of tyVar]
2. $\mathcal{TV}(cx) \supseteq \mathcal{FTV}(\beta)$ [2]

tyInst)

1. $\mathcal{FTV}(\tau[\overline{T}]) = \bigcup_i \mathcal{FTV}(T_i)$
2. $\forall i.\ \ cx \vdash T_i : \textsc{type}$ [premise of tyInst]
3. $\forall i.\ \ \mathcal{TV}(cx) \supseteq \mathcal{FTV}(T_i)$ [2]
4. $\mathcal{TV}(cx) \supseteq \mathcal{FTV}(\tau[\overline{T}])$ [1, 3]

tyArr)

1. $\mathcal{FTV}(T_1 \to T_2) = \mathcal{FTV}(T_1) \cup \mathcal{FTV}(T_2)$
2. $cx, \mathsf{var}\ v{:}T_1 \vdash T_2 : \textsc{type}$ [premise of tyArr]
3. $\mathcal{TV}(cx, \mathsf{var}\ v{:}T_1) \supseteq \mathcal{FTV}(T_2)$ [induction hypothesis, 2]
4. $\mathcal{TV}(cx, \mathsf{var}\ v{:}T_1) = \mathcal{TV}(cx)$ [definition of $\mathcal{TV}$]
5. $\vdash cx, \mathsf{var}\ v{:}T_1 : \textsc{context}$ [Theorem 4.37, 2]
6. $\mathcal{TV}(cx) \supseteq \mathcal{FTV}(T_1)$ [induction hypothesis, 5]
7. $\mathcal{TV}(cx) \supseteq \mathcal{FTV}(T_1 \to T_2)$ [1, 6, 4, 3]

stArr)

1. $cx, \mathsf{var}\ v{:}T \vdash T_1 \prec_r T_2$ [premise of stArr]
2. $\mathcal{FTV}(T_1) \cup \mathcal{FTV}(r) \cup \mathcal{FTV}(T_2) \subseteq \mathcal{TV}(cx, \mathsf{var}\ v{:}T)$ [induction hypothesis, 1]
3. $\mathcal{FTV}(T_1) \cup \mathcal{FTV}(r) \cup \mathcal{FTV}(T_2) \subseteq \mathcal{TV}(cx)$ [2, $\mathcal{TV}(\mathsf{var}\ v{:}T) = \emptyset$]
4. $\vdash cx, \mathsf{var}\ v{:}T : \textsc{context}$ [Theorem 4.37, 1]
5. $\mathcal{FTV}(\mathsf{var}\ v{:}T) \subseteq \mathcal{TV}(cx)$ [induction hypothesis, 4]
6. $\mathcal{FTV}(T) \subseteq \mathcal{TV}(cx)$ [definition of $\mathcal{FTV}$, 5]

Let $r' = \lambda v' : T \to T_2.\ \forall v'' : T.\ r\ (v'\ v'')$:

7. $\mathcal{FTV}(r') \subseteq \mathcal{TV}(cx)$ [3, 6, Theorem 4.36]
8. $\mathcal{FTV}(T \to T_1) \subseteq \mathcal{TV}(cx)$ [3, 6]
9. $\mathcal{FTV}(T \to T_2) \subseteq \mathcal{TV}(cx)$ [3, 6]
10. $\mathcal{FTV}(T \to T_1) \cup \mathcal{FTV}(r') \cup \mathcal{FTV}(T \to T_2) \subseteq \mathcal{TV}(cx)$ [7, 8, 9]

exVar)

1. $\mathsf{var}\ v : T \in cx$ [premise of exVar]
2. $\exists cx_1, cx_2.\ \ cx = cx_1, \mathsf{var}\ v : T, cx_2$ [1]
3. $\vdash cx : \text{CONTEXT}$ [premise of exVar]
4. $\mathcal{TV}(cx)$
   $\supseteq$ [2]
   $\mathcal{TV}(cx_1)$
   $\supseteq$ [induction hypothesis, 3, 2]
   $\mathcal{FTV}(T)$

exOp)

1. $\mathcal{FTV}(o[\overline{T}]) = \bigcup_i \mathcal{FTV}(T_i)$
2. $\mathcal{FTV}(T[\overline{\beta}/\overline{T}]) = (\mathcal{FTV}(T) - \overline{\beta}) \cup \bigcup_{\beta_i \in \mathcal{FTV}(T)} \mathcal{FTV}(T_i)$ [Theorem 4.29]
3. $\forall i.\ \ cx \vdash T_i : \text{TYPE}$ [premise of exOp]
4. $\forall i.\ \ \mathcal{TV}(cx) \supseteq \mathcal{FTV}(T_i)$ [induction hypothesis, 3]
5. $\mathcal{TV}(cx) \supseteq \mathcal{FTV}(o[\overline{T}])$ [4, 1]
6. $\mathsf{op}\ o : \{\overline{\beta}\}\ T \in cx$ [premise of exOp]
7. $\exists cx_1, cx_2.\ \ cx = cx_1, \mathsf{op}\ o : \{\overline{\beta}\}\ T, cx_2$ [6]
8. $\vdash cx : \text{CONTEXT}$ [premise of exOp]
9. $\mathcal{TV}(cx)$
   $\supseteq$ [7]
   $\mathcal{TV}(cx_1)$
   $\supseteq$ [induction hypothesis, 8, 7]
   $\mathcal{FTV}(\mathsf{op}\ o : \{\overline{\beta}\}\ T)$
   $\supseteq$
   $\mathcal{FTV}(T) - \overline{\beta}$
10. $\bigcup_{\beta_i \in \mathcal{FTV}(T)} \mathcal{FTV}(T_i) \subseteq \bigcup_i \mathcal{FTV}(T_i)$
11. $\mathcal{TV}(cx) \supseteq \bigcup_{\beta_i \in \mathcal{FTV}(T)} \mathcal{FTV}(T_i)$ [4, 10]
12. $\mathcal{TV}(cx) \supseteq \mathcal{FTV}(o[\overline{T}]) \cup \mathcal{FTV}(T[\overline{\beta}/\overline{T}])$ [5, 2, 9, 11]

exAbs)

1. $cx, \mathsf{var}\ v : T \vdash e : T'$ [premise of exAbs]
2. $\mathcal{TV}(cx, \mathsf{var}\ v : T) \supseteq \mathcal{FTV}(e) \cup \mathcal{FTV}(T')$ [induction hypothesis, 1]
3. $\mathcal{TV}(cx) \supseteq \mathcal{FTV}(e) \cup \mathcal{FTV}(T')$ [2]
4. $\mathcal{FTV}(\lambda v : T.\ e) = \mathcal{FTV}(T) \cup \mathcal{FTV}(e)$
5. $\mathcal{FTV}(T \to T') = \mathcal{FTV}(T) \cup \mathcal{FTV}(T')$
6. $\vdash cx, \mathsf{var}\ v : T : \text{CONTEXT}$ [Theorem 4.37, 1]
7. $\mathcal{TV}(cx) \supseteq \mathcal{FTV}(T)$ [induction hypothesis, 6]
8. $\mathcal{TV}(cx) \supseteq \mathcal{FTV}(\lambda v : T.\ e) \cup \mathcal{FTV}(T \to T')$ [3, 4, 5, 7]

thAx)
Analogous to exOp.

thBool)

$$\mathcal{FTV}(\forall v : \mathsf{Bool} \to \mathsf{Bool}.\ (v\ \mathsf{true} \wedge v\ \mathsf{false} \Leftrightarrow (\forall v' : \mathsf{Bool}.\ v\ v'))) = \emptyset \qquad \text{[Theorem 4.36]}$$

all other rules)
Analogous to tyInst: each free type variable in the right-hand side of the judgement $cx \vdash \ldots$ is also a free type variable in the right-hand side of some premise $cx' \vdash \ldots$ of the rule such that $\mathcal{TV}(cx') = \mathcal{TV}(cx)$ (the only rules in which $cx' \neq cx$ are exIf, thIfSubst, thIf). We use Theorem 4.36 for stRefl, stRec, exThe, thExt, thRec, and thProjSub. We use Theorem 4.22 for exAbsAlpha. We use Theorem 4.21 for thAbs.

Proof of Theorem 4.58

By induction on the derivation of the judgements in the antecedents of the implications:

cxMT)
This rule is impossible because $cx_1, \mathsf{var}\ u : S, cx_2 \neq \epsilon$.

cxTdec)
The judgement in the antecedent of the implication must end with a type declaration, so the cx_2 in the judgement must end with that type declaration. For convenience we "rename" cx_2 to $cx_2, \mathsf{ty}\ \tau : n$. The instance of cxTdec used to derive the judgement is thus

$$\frac{\vdash cx_1, \mathsf{var}\ u : S, cx_2 : \text{CONTEXT} \quad \tau \notin \mathcal{TN}(cx_1, \mathsf{var}\ u : S, cx_2)}{\vdash cx_1, \mathsf{var}\ u : S, cx_2, \mathsf{ty}\ \tau : n : \text{CONTEXT}} \quad (0)$$

and the judgement $\vdash cx_1, \mathsf{var}\ u' : S, cx_2[u/u'], \mathsf{ty}\ \tau : n : \text{CONTEXT}$ is derived as follows:

1. $\vdash cx_1, \mathsf{var}\ u : S, cx_2 : \text{CONTEXT}$ [0]
2. $\vdash cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] : \text{CONTEXT}$ [induction hypothesis, 1]
3. $\tau \notin \mathcal{TN}(cx_1, \mathsf{var}\ u : S, cx_2)$ [0]
4. $\mathcal{TN}(cx_1, \mathsf{var}\ u : S, cx_2)$
   $=$
   $\mathcal{TN}(cx_1) \cup \mathcal{TN}(cx_2)$
   $=$ [Theorem 4.25]
   $\mathcal{TN}(cx_1) \cup \mathcal{TN}(cx_2[u/u'])$
   $=$
   $\mathcal{TN}(cx_1, \mathsf{var}\ u' : S, cx_2[u/u'])$
5. $\tau \notin \mathcal{TN}(cx_1, \mathsf{var}\ u' : S, cx_2[u/u'])$ [3, 4]
6. $\vdash cx_1, \mathsf{var}\ u' : S, cx_2[u/u'], \mathsf{ty}\ \tau : n : \text{CONTEXT}$ [cxTdec, 2, 5]

cxOdec, cxAx, cxTVdec)
Analogous to cxTdec: starting with the instance of the rule used to derive the judgement in the antecedent of the implication, we replace u with u' in the premises of the rule instance and then we apply the rule to the transformed premises to obtain the conclusion with u replaced with u' (using Theorem 4.25 as needed).

cxVdec)
If $cx_2 = \epsilon$, the instance of cxVdec used to derive the judgement in the antecedent of the implication is

$$\frac{cx_1 \vdash T : \text{TYPE} \quad u \notin \mathcal{V}(cx_1)}{\vdash cx_1, \mathsf{var}\ u : T : \text{CONTEXT}} \quad (0)$$

and the judgement $\vdash cx_1, \mathsf{var}\ u' : T : \text{CONTEXT}$ is derived as follows:

1. $cx_1 \vdash T : \textsc{type}$ [0]
2. u' fresh [hypothesis]
3. $u' \notin \mathcal{V}(cx_1)$ [2]
4. $\vdash cx_1, \mathsf{var}\ u' : T : \textsc{context}$ [cxVdec, 1, 3]

If $cx_2 \neq \epsilon$, the judgement in the antecedent of the implication must end with a variable declaration, so the cx_2 in the judgement must end with that variable declaration. For convenience we "rename" cx_2 to $cx_2, \mathsf{var}\ v : T$. The instance of cxVdec used to derive the judgement is thus

$$\frac{cx_1, \mathsf{var}\ u : S, cx_2 \vdash T : \textsc{type} \quad v \notin \mathcal{V}(cx_1, \mathsf{var}\ u : S, cx_2)}{\vdash cx_1, \mathsf{var}\ u : S, cx_2, \mathsf{var}\ v : T : \textsc{context}} \quad (0)$$

and the judgement $\vdash cx_1, \mathsf{var}\ u' : S, cx_2[u/u'], \mathsf{var}\ v : T : \textsc{context}$ is derived as follows:

1. $v \notin \mathcal{V}(cx_1, \mathsf{var}\ u : S, cx_2)$ [0]
2. $\mathcal{V}(cx_1, \mathsf{var}\ u : S, cx_2) = \mathcal{V}(cx_1) \cup \{u\} \cup \mathcal{V}(cx_2)$
3. $v \notin \mathcal{V}(cx_1)$ [1, 2]
4. $\mathcal{V}(cx_2) = \mathcal{V}(cx_2[u/u'])$ [Theorem 4.25]
5. $v \notin \mathcal{V}(cx_2[u/u'])$ [1, 2, 4]
6. u' fresh [hypothesis]
7. $v \neq u'$ [6]
8. $\mathcal{V}(cx_1, \mathsf{var}\ u' : S, cx_2[u/u']) = \mathcal{V}(cx_1) \cup \{u'\} \cup \mathcal{V}(cx_2[u/u'])$
9. $v \notin \mathcal{V}(cx_1, \mathsf{var}\ u' : S, cx_2[u/u'])$ [8, 3, 7, 5]
10. $cx_1, \mathsf{var}\ u : S, cx_2 \vdash T : \textsc{type}$ [0]
11. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash T : \textsc{type}$ [10, induction hypothesis]
12. $\vdash cx_1, \mathsf{var}\ u' : S, cx_2[u/u'], \mathsf{var}\ v : T : \textsc{context}$ [cxVdec, 11, 9]

tyBool)

Let the instance of tyBool used to derive the judgement in the antecedent of the implication be

$$\frac{\vdash cx_1, \mathsf{var}\ u : S, cx_2 : \textsc{context}}{cx_1, \mathsf{var}\ u : S, cx_2 \vdash \mathsf{Bool} : \textsc{type}} \quad (0)$$

The judgement in the consequent of the implication is derived as follows:

1. $\vdash cx_1, \mathsf{var}\ u : S, cx_2 : \textsc{context}$ [0]
2. $\vdash cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] : \textsc{context}$ [induction hypothesis, 1]
3. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 2]

exVar)

If the judgement in the antecedent of the implication is $cx_1, \mathsf{var}\ u : S, cx_2 \vdash u : \ldots$, the instance of exVar used to derive the judgement is

$$\frac{\vdash cx_1, \mathsf{var}\ u : S, cx_2 : \textsc{context} \quad \mathsf{var}\ u : S \in cx_1, \mathsf{var}\ u : S, cx_2}{cx_1, \mathsf{var}\ u : S, cx_2 \vdash u : S} \quad (0)$$

(the type $\ldots$ is S by Theorem 4.41) and the judgement in the consequent of the implication is derived as follows:

1. $\vdash cx_1, \mathsf{var}\ u : S, cx_2 : \textsc{context}$ [0]
2. $\vdash cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] : \textsc{context}$ [induction hypothesis, 1]

3. $\mathsf{var}\ u' : S \in cx_1, \mathsf{var}\ u' : S, cx_2[u/u']$
4. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash u' : S$ [exVar, 2, 3]
5. $u[u/u'] = u'$
6. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash u[u/u'] : S$ [4, 5]

If the judgement in the antecedent of the implication is $cx_1, \mathsf{var}\ u : S, cx_2 \vdash v : T$ with $v \neq u$, the instance of exVar used to derive the judgement is

$$\frac{\vdash cx_1, \mathsf{var}\ u : S, cx_2 : \textsc{context} \quad \mathsf{var}\ v : T \in cx_1, \mathsf{var}\ u : S, cx_2}{cx_1, \mathsf{var}\ u : S, cx_2 \vdash v : T} \quad (0)$$

and the judgement in the consequent of the implication is derived as follows:

1. $\vdash cx_1, \mathsf{var}\ u : S, cx_2 : \textsc{context}$ [0]
2. $\vdash cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] : \textsc{context}$ [induction hypothesis, 1]
3. $\mathsf{var}\ v : T \in cx_1, \mathsf{var}\ u : S, cx_2$ [0]
4. $\mathsf{var}\ v : T \in cx_1, cx_2$ [3, hypothesis $v \neq u$]
5. $\mathsf{var}\ v : T \in cx_1, cx_2[u/u']$ [4, Theorem 4.25]
6. $\mathsf{var}\ v : T \in cx_1, \mathsf{var}\ u' : S, cx_2[u/u']$ [5]
7. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash v : T$ [exVar, 2, 6]
8. $v[u/u'] = v$ [hypothesis $v \neq u$]
9. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash v[u/u'] : T$ [7, 8]

exAbs)
Let the instance of exAbs used to derive the judgement in the antecedent of the implication be

$$\frac{cx_1, \mathsf{var}\ u : S, cx_2, \mathsf{var}\ v : T \vdash e : T'}{cx_1, \mathsf{var}\ u : S, cx_2 \vdash \lambda v : T.\ e : T \to T'} \quad (0)$$

The judgement in the consequent of the implication is derived as follows:

1. $cx_1, \mathsf{var}\ u : S, cx_2, \mathsf{var}\ v : T \vdash e : T'$ [0]
2. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'], \mathsf{var}\ v : T \vdash e[u/u'] :$ [induction hypothesis, 1]
3. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash \lambda v : T.\ e[u/u'] : T \to T'$ [exAbs, 2]
4. $\vdash cx_1, \mathsf{var}\ u : S, cx_2, \mathsf{var}\ v : T : \textsc{context}$ [Theorem 4.37, 1]
5. $v \neq u$ [Theorem 4.41, 4]
6. $(\lambda v : T.\ e)[u/u'] = \lambda v : T.\ e[u/u']$ [5]
7. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash (\lambda v : T.\ e)[u/u'] : T \to T'$ [3, 6]

exAbsAlpha)
Let the instance of exAbsAlpha used to derive the judgement in the antecedent of the implication be

$$\frac{cx_1, \mathsf{var}\ u : S, cx_2 \vdash \lambda v : T.\ e : T' \quad v' \notin \mathcal{FV}(e) \cup \mathcal{CV}(e, v)}{cx_1, \mathsf{var}\ u : S, cx_2 \vdash \lambda v' : T.\ e[v/v'] : T'} \quad (0)$$

The judgement in the consequent of the implication is derived as follows:

1. $cx_1, \mathsf{var}\ u : S, cx_2 \vdash \lambda v : T.\ e : T'$ [0]
2. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash (\lambda v : T.\ e)[u/u'] : T'$ [induction hypothesis, 1]
3. $v' \notin \mathcal{FV}(e) \cup \mathcal{CV}(e, v)$ [0]

4. $\mathcal{FV}(\lambda v'{:}T.\ e[v/v']) = \mathcal{FV}(\lambda v{:}T.\ e)$ [Theorem 4.18, 3]

If $u \notin \mathcal{FV}(\lambda v{:}T.\ e)$:

5. $(\lambda v{:}T.\ e)[u/u'] = \lambda v{:}T.\ e$ [Theorem 4.1]
6. $cx_1, \mathsf{var}\ u'{:}S, cx_2[u/u'] \vdash \lambda v{:}T.\ e : T'$ [2, 5]
7. $cx_1, \mathsf{var}\ u'{:}S, cx_2[u/u'] \vdash \lambda v'{:}T.\ e[v/v'] : T'$ [exAbsAlpha, 6, 3]
8. $u \notin \mathcal{FV}(\lambda v'{:}T.\ e[v/v'])$ [4, hypothesis $u \notin \mathcal{FV}(\lambda v{:}T.\ e)$]
9. $(\lambda v'{:}T.\ e[v/v'])[u/u'] = \lambda v'{:}T.\ e[v/v']$ [Theorem 4.1, 8]
10. $cx_1, \mathsf{var}\ u'{:}S, cx_2[u/u'] \vdash (\lambda v'{:}T.\ e[v/v'])[u/u'] : T'$ [7, 9]

If $u \in \mathcal{FV}(\lambda v{:}T.\ e)$:

5. $u \in \mathcal{FV}(e) - \{v\}$
6. $u \in \mathcal{FV}(e)$ [5]
7. $u \neq v$ [5]
8. $(\lambda v{:}T.\ e)[u/u'] = \lambda v{:}T.\ e[u/u']$ [7]
9. $cx_1, \mathsf{var}\ u'{:}S, cx_2[u/u'] \vdash \lambda v{:}T.\ e[u/u'] : T'$ [2, 8]
10. u' fresh [hypothesis]
11. $u' \notin \mathcal{CV}(e, u)$ [10]
12. $OKsbs(e, u, u')$ [11]
13. $\mathcal{FV}(e[u/u']) = (\mathcal{FV}(e) - \{u\}) \cup \{u'\}$ [Theorem 4.8, 6, 12]
14. $v' \notin \mathcal{FV}(e)$ [3]
15. $v' \notin \mathcal{FV}(e) - \{u\}$ [14]
16. $v' \neq u'$ [10]
17. $v' \notin \mathcal{FV}(e[u/u'])$ [13, 15, 16]
18. $v \neq u'$ [10]
19. $\mathcal{CV}(e, v) = \mathcal{CV}(e[u/u'], v)$ [Theorem 4.17, 12, 7, 18]
20. $v' \notin \mathcal{CV}(e, v)$ [3]
21. $v' \notin \mathcal{CV}(e[u/u'], v)$ [19, 20]
22. $v' \notin \mathcal{FV}(e[u/u']) \cup \mathcal{CV}(e[u/u'], v)$ [17, 21]
23. $cx_1, \mathsf{var}\ u'{:}S, cx_2[u/u'] \vdash \lambda v'{:}T.\ e[u/u'][v/v'] : T'$ [exAbsAlpha, 9, 22]
24. $u \neq v'$ [6, 14]
25. $e[u/u'][v/v'] = e[v/v'][u/u']$ [Theorem 4.10, 7, 24, 18]
26. $(\lambda v'{:}T.\ e[v/v'])[u/u'] = \lambda v'{:}T.\ e[v/v'][u/u']$ [24]
27. $cx_1, \mathsf{var}\ u'{:}S, cx_2[u/u'] \vdash (\lambda v'{:}T.\ e[v/v'])[u/u'] : T'$ [23, 25, 26]

thAx)

Let the instance of thAx used to derive the judgement in the antecedent of the implication be

$$\frac{\begin{array}{c}\vdash cx_1, \mathsf{var}\ u{:}S, cx_2 : \text{CONTEXT} \\ \mathsf{ax}\ \{\overline{\beta}\}\ e \in cx_1, \mathsf{var}\ u{:}S, cx_2 \\ \forall i.\quad cx_1, \mathsf{var}\ u{:}S, cx_2 \vdash T_i : \text{TYPE}\end{array}}{cx_1, \mathsf{var}\ u{:}S, cx_2 \vdash e[\overline{\beta}/\overline{T}]} \quad (0)$$

The judgement in the consequent of the implication is derived as follows:

1. $\vdash cx_1, \mathsf{var}\ u{:}S, cx_2 : \text{CONTEXT}$ [0]

2. $\vdash cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] : \textsc{context}$ [induction hypothesis, 1]
3. $\forall i.\ \ cx_1, \mathsf{var}\ u : S, cx_2 \vdash T_i : \textsc{type}$ [0]
4. $\forall i.\ \ cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash T_i : \textsc{type}$ [induction hypothesis, 3]
5. $\mathsf{ax}\ \{\overline{\beta}\}\ e \in cx_1, \mathsf{var}\ u : S, cx_2$ [0]
6. $\mathsf{ax}\ \{\overline{\beta}\}\ e \in cx_1, cx_2$ [5]

If $\mathsf{ax}\ \{\overline{\beta}\}\ e \in cx_1$:

7. $\exists cx_1', cx_1''.\ \ cx_1 = cx_1', \mathsf{ax}\ \{\overline{\beta}\}\ e, cx_1''$
8. $cx_1', \mathsf{tvar}\ \overline{\beta} \vdash e : \mathsf{Bool}$ [Theorem 4.43, 1, 7]
9. $\mathcal{FV}(e) \subseteq \mathcal{V}(cx_1')$ [Theorem 4.54, 8]
10. $u \notin \mathcal{V}(cx_1')$ [Theorem 4.41, 1, 7]
11. $u \notin \mathcal{FV}(e)$ [10, 9]
12. $\mathsf{ax}\ \{\overline{\beta}\}\ e \in cx_1, \mathsf{var}\ u' : S, cx_2[u/u']$ [hypothesis $\mathsf{ax}\ \{\overline{\beta}\}\ e \in cx_1$]
13. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash e[\overline{\beta}/\overline{T}]$ [thAx, 2, 12, 4]
14. $\mathcal{FV}(e[\overline{\beta}/\overline{T}]) = \mathcal{FV}(e)$ [Theorem 4.19]
15. $u \notin \mathcal{FV}(e[\overline{\beta}/\overline{T}])$ [11, 14]
16. $e[\overline{\beta}/\overline{T}][u/u'] = e[\overline{\beta}/\overline{T}]$ [Theorem 4.1, 15]
17. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash e[\overline{\beta}/\overline{T}][u/u']$ [13, 16]

If $\mathsf{ax}\ \{\overline{\beta}\}\ e \notin cx_1$:

7. $\mathsf{ax}\ \{\overline{\beta}\}\ e \in cx_2$ [6]
8. $\mathsf{ax}\ \{\overline{\beta}\}\ e[u/u'] \in cx_2[u/u']$ [Theorem 4.25, 7]
9. $\mathsf{ax}\ \{\overline{\beta}\}\ e[u/u'] \in cx_1, \mathsf{var}\ u' : S, cx_2[u/u']$ [8]
10. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash e[u/u'][\overline{\beta}/\overline{T}]$ [thAx, 2, 9, 4]
11. $e[u/u'][\overline{\beta}/\overline{T}] = e[\overline{\beta}/\overline{T}][u/u']$ [Theorem 4.24]
12. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash e[\overline{\beta}/\overline{T}][u/u']$ [10, 11]

thAbs)

Let the instance of thAbs used to derive the judgement in the antecedent of the implication be

$$\frac{cx_1, \mathsf{var}\ u : S, cx_2 \vdash (\lambda v : T.\ e)\ e' : \ldots \qquad OKsbs(e, v, e')}{cx_1, \mathsf{var}\ u : S, cx_2 \vdash (\lambda v : T.\ e)\ e' \equiv e[v/e']} \quad (0)$$

The judgement in the consequent of the implication is derived as follows:

1. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash ((\lambda v : T.\ e)\ e')[u/u'] : \ldots$ [induction hypothesis, 0]
2. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash (\lambda v : T.\ e)[u/u']\ e'[u/u'] : \ldots$ [1, definition of $\_[\_/\_]$]
3. $OKsbs(e, v, e')$ [0]
4. u' fresh [hypothesis]
5. $OKsbs(e', u, u')$ [4]

If $u = v$:

6. $(\lambda v : T.\ e)[u/u'] = \lambda v : T.\ e$ [definition of $\_[\_/\_]$]
7. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash (\lambda v : T.\ e)\ e'[u/u'] : \ldots$ [2, 6]

If $u \notin \mathcal{FV}(e')$:

8. $e'[u/u'] = e'$ [Theorem 4.1]
9. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash (\lambda v : T.\ e)\ e' : \ldots$ [7, 8]
10. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash (\lambda v : T.\ e)\ e' \equiv e[v/e']$ [thAbs, 9, 3]
11. $v \notin \mathcal{FV}(e')$ [hypothesis $u \notin \mathcal{FV}(e')$, hypothesis $u = v$]
12. $v \notin \mathcal{FV}(e[v/e'])$ [Theorem 4.9, 3, 11]
13. $u \notin \mathcal{FV}(e[v/e'])$ [12, hypothesis $u = v$]
14. $e[v/e'][u/u'] = e[v/e']$ [Theorem 4.1, 13]
15. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash ((\lambda v : T.\ e)\ e' \equiv e[v/e'])[u/u']$ [10, 6, 8, 14]

If $u \in \mathcal{FV}(e')$:

8. $\mathcal{FV}(e'[u/u']) = (\mathcal{FV}(e') - \{u\}) \cup \{u'\}$ [Theorem 4.8, 5]
9. $\mathcal{FV}(e') \cap \mathcal{CV}(e, v) = \emptyset$ [3]
10. $(\mathcal{FV}(e') - \{u\}) \cap \mathcal{CV}(e, v) = \emptyset$ [9]
11. $u' \notin \mathcal{CV}(e, v)$ [4]
12. $OKsbs(e, v, e'[u/u'])$ [8, 10, 11]
13. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash (\lambda v : T.\ e)\ e'[u/u'] \equiv e[v/e'[u/u']]$ [thAbs, 7, 12]
14. $e[v/e'[u/u']] = e[u/e'[u/u']]$ [hypothesis $u = v$]
15. $e[u/e'[u/u']] = e[u/e'][u/u']$ [Theorem 4.11]
16. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash (\lambda v : T.\ e)\ e'[u/u'] \equiv e[v/e'][u/u']$ [13, 14, 15, hypothesis $u = v$]
17. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash ((\lambda v : T.\ e)\ e' \equiv e[v/e'])[u/u']$ [16, 6]

If $u \neq v$:

6. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash (\lambda v : T.\ e[u/u'])\ e'[u/u'] : \ldots$ [2]
7. $OKsbs(e, u, u')$ [4]
8. $v \neq u'$ [4]
9. $\mathcal{CV}(e, v) = \mathcal{CV}(e[u/u'], v)$ [Theorem 4.17, 7, hypothesis $v \neq u$, 8]
10. $\mathcal{FV}(e') \cap \mathcal{CV}(e, v) = \emptyset$ [3]
11. $\mathcal{FV}(e') \cap \mathcal{CV}(e[u/u'], v) = \emptyset$ [9, 10]

If $u \notin \mathcal{FV}(e')$:

12. $e'[u/u'] = e'$ [Theorem 4.1]
13. $\mathcal{FV}(e'[u/u']) = \mathcal{FV}(e')$ [12]
14. $OKsbs(e[u/u'], v, e'[u/u'])$ [11, 13]
15. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash (\lambda v : T.\ e[u/u'])\ e'[u/u'] \equiv e[u/u'][v/e'[u/u']]$ [thAbs, 6, 14]
16. $u' \neq u$ [4]
17. $u \notin \mathcal{FV}(e'[u/u'])$ [Theorem 4.9, 5, 16]
18. $e[u/u'][v/e'[u/u']] = e[v/e'[u/u']][u/u']$ [Theorem 4.10, hypothesis $u \neq v$, 8, 17]
19. $e[v/e'[u/u']][u/u'] = e[v/e'][u/u']$ [Theorem 4.13, 3]
20. $(\lambda v : T.\ e)[u/u'] = \lambda v : T.\ e[u/u']$ [hypothesis $v \neq u$]
21. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash ((\lambda v : T.\ e)\ e' \equiv e[v/e'])[u/u']$ [15, 20, 18, 19]

If $u \in \mathcal{FV}(e')$:

12. $\mathcal{FV}(e'[u/u']) = (\mathcal{FV}(e') - \{u\}) \cup \{u'\}$ [Theorem 4.8, 5]

13. $(\mathcal{FV}(e') - \{u\}) \cap \mathcal{CV}(e[u/u'], v) = \emptyset$ [11]
14. $u' \notin \mathcal{CV}(e[u/u'], v)$ [4]
15. $\mathcal{FV}(e'[u/u']) \cap \mathcal{CV}(e[u/u'], v) = \emptyset$ [12, 13, 14]
16. $OKsbs(e[u/u'], v, e'[u/u'])$ [15]
17. $cx_1, \mathsf{var}\ u' : S, cx_2[u/u'] \vdash ((\lambda v{:}T.\ e)\ e' \equiv e[v/e'])[u/u']$
[same derivation as 14–21 above for case $u \notin \mathcal{FV}(e')$]

all other rules)

Analogous to tyBool: starting with the instance of the rule used to derive the judgement in the antecedent of the implication, we replace u with u' in the premises of the rule instance and then we apply the rule to the transformed premises to obtain the conclusion with u replaced with u'. We use Theorem 4.25 for tyVar, tyInst, exVar, and exOp and we use Theorem 4.33 for exIf, thIfSubst, and thIf.

For tyRestr, we use the fact that $\mathcal{FV}(r) = \emptyset$ (explicit premise of the rule instance) and so $r[u/u'] = r$ by Theorem 4.1.

For exSub we use the fact that $\mathcal{FV}(r) = \emptyset$ by Theorem 4.50 and so $(r\ e)[u/u'] = r\ e[u/u']$ by Theorem 4.1.

For thSub we use the fact that $\mathcal{FV}(r) = \emptyset$ by Theorem 4.53 and so $(r\ e)[u/u'] = r\ e[u/u']$ by Theorem 4.1.

For thBool, thExt, thRec, and thProjSub we use the fact that the theorem e that is the conclusion of the rule has no free variables (by Theorem 4.32) and so $e[u/u'] = e$ by Theorem 4.1.

Proof of Theorem 4.59

By induction on the derivation of the judgement $cx \vdash \ldots$:

tyInst)

1. $\vdash cx'$: context [hypothesis]
2. $cx \subseteq cx'$ [hypothesis]
3. $\mathsf{ty}\ \tau{:}n \in cx$ [premise of tyInst]
4. $\mathsf{ty}\ \tau{:}n \in cx'$ [3, 2]
5. $|\overline{T}| = n$ [premise of tyInst]
6. $\forall i.\ \ cx \vdash T_i$: type [premise of tyInst]
7. $\forall i.\ \ cx' \vdash T_i$: type [induction hypothesis, 6]
8. $cx' \vdash \tau[\overline{T}]$: type [tyInst, 1, 4, 5, 7]

all other rules except tyArr, stArr, and exAbs)

Analogous to tyInst: starting with the rule used to derive the judgement $cx \vdash \ldots$, we replace cx with cx' in the premises of the rule and then we apply the rule to the transformed premises to obtain the conclusion with cx' instead of cx. For exIf, thIfSubst, and thIf, we use the fact that

$$\vdash cx' : \text{context} \quad \wedge \quad cx \subseteq cx' \quad \wedge \quad cx, \mathsf{ax}\ e \vdash \ldots \quad \Rightarrow \quad \vdash cx', \mathsf{ax}\ e : \text{context}$$

which is proved as follows:

1. $cx, \mathsf{ax}\ e \vdash \ldots$ [hypothesis]
2. $\vdash cx, \mathsf{ax}\ e$: context [Theorem 4.37, 1]
3. $cx \vdash e$: Bool [Theorem 4.43, 2]

4. $\vdash cx'$: CONTEXT [hypothesis]
5. $cx \subseteq cx'$ [hypothesis]
6. $cx' \vdash e$: Bool [induction hypothesis, 5, 4, 3]
7. $\vdash cx', \mathsf{ax}\ e$: CONTEXT [cxAx, 4, 6]

(This fact is used twice in each of exIf, thIfSubst, and thIf, the first time with $e = e_0$ and the second time with $e = \neg\, e_0$.)

tyArr)

1. $cx, \mathsf{var}\ v{:}T_1 \vdash T_2$: TYPE [premise of tyArr]
2. $\vdash cx, \mathsf{var}\ v{:}T_1$: CONTEXT [Theorem 4.37, 1]
3. $cx \vdash T_1$: TYPE [Theorem 4.44, 2]
4. $cx' \vdash T_1$: TYPE [induction hypothesis, 3]
5. $cx \subseteq cx'$ [hypothesis]

If $v \notin \mathcal{V}(cx')$:

6. $\vdash cx', \mathsf{var}\ v{:}T_1$: CONTEXT [cxVdec, 4]
7. $cx, \mathsf{var}\ v{:}T_1 \subseteq cx', \mathsf{var}\ v{:}T_1$ [5]
8. $cx', \mathsf{var}\ v{:}T_1 \vdash T_2$: TYPE [induction hypothesis, 6, 7, 1]
9. $cx' \vdash T_1 \to T_2$: TYPE [tyArr, 8]

If $v \in \mathcal{V}(cx')$, pick a fresh variable v':

6. $cx, \mathsf{var}\ v'{:}T_1 \vdash T_2$: TYPE [Theorem 4.58, 1]
7. $\vdash cx', \mathsf{var}\ v'{:}T_1$: CONTEXT [cxVdec, 4]
8. $cx, \mathsf{var}\ v'{:}T_1 \subseteq cx', \mathsf{var}\ v'{:}T_1$ [5]
9. $cx', \mathsf{var}\ v'{:}T_1 \vdash T_2$: TYPE [induction hypothesis, 7, 8, 6]
10. $cx' \vdash T_1 \to T_2$: TYPE [tyArr, 9]

stArr)

1. $cx, \mathsf{var}\ v{:}T \vdash T_1 \prec_r T_2$ [premise of stArr]
2. $\vdash cx, \mathsf{var}\ v{:}T$: CONTEXT [Theorem 4.37, 1]
3. $cx \vdash T$: TYPE [Theorem 4.44, 2]
4. $cx' \vdash T$: TYPE [induction hypothesis, 3]
5. $cx \subseteq cx'$ [hypothesis]
6. $v' \neq v''$ [premise of stArr]
7. $r' = \lambda v'{:}T \to T_2.\ \forall v''{:}T.\ r\ (v'\ v'')$ [premise of stArr]

If $v \notin \mathcal{V}(cx')$:

8. $\vdash cx', \mathsf{var}\ v{:}T$: CONTEXT [cxVdec, 4]
9. $cx, \mathsf{var}\ v{:}T \subseteq cx', \mathsf{var}\ v{:}T$ [5]
10. $cx', \mathsf{var}\ v{:}T \vdash T_1 \prec_r T_2$ [induction hypothesis, 8, 9, 1]
11. $cx' \vdash T \to T_1 \prec_{r'} T \to T_2$ [stArr, 10, 6, 7]

If $v \in \mathcal{V}(cx')$, pick a fresh variable u:

8. $cx, \mathsf{var}\ u{:}T \vdash T_1 \prec_r T_2$ [Theorem 4.58, 1]

9. $\vdash cx', \mathsf{var}\ u{:}T : \textsc{context}$ [cxVdec, 4]
10. $cx, \mathsf{var}\ u{:}T \subseteq cx', \mathsf{var}\ u{:}T$ [5]
11. $cx', \mathsf{var}\ u{:}T \vdash T_1 \prec_r T_2$ [induction hypothesis, 9, 10, 8]
12. $cx' \vdash T \to T_1 \prec_{r'} T \to T_2$ [stArr, 11, 6, 7]

exAbs)

1. $cx, \mathsf{var}\ v{:}T \vdash e : T'$ [premise of exAbs]
2. $\vdash cx, \mathsf{var}\ v{:}T : \textsc{context}$ [Theorem 4.37, 1]
3. $cx \vdash T : \textsc{type}$ [Theorem 4.44, 2]
4. $cx' \vdash T : \textsc{type}$ [induction hypothesis, 3]
5. $cx \subseteq cx'$ [hypothesis]

If $v \notin \mathcal{V}(cx')$:

6. $\vdash cx', \mathsf{var}\ v{:}T : \textsc{context}$ [cxVdec, 4]
7. $cx, \mathsf{var}\ v{:}T \subseteq cx', \mathsf{var}\ v{:}T$ [5]
8. $cx', \mathsf{var}\ v{:}T \vdash e : T'$ [induction hypothesis, 1, 6, 7]
9. $cx' \vdash \lambda v{:}T.\ e : T \to T'$ [exAbs, 8]

If $v \in \mathcal{V}(cx')$, pick a fresh variable v':

6. $cx, \mathsf{var}\ v'{:}T \vdash e[v/v'] : T'$ [Theorem 4.58, 1]
7. $\vdash cx', \mathsf{var}\ v'{:}T : \textsc{context}$ [cxVdec, 4]
8. $cx, \mathsf{var}\ v'{:}T \subseteq cx', \mathsf{var}\ v'{:}T$ [5]
9. $cx', \mathsf{var}\ v'{:}T \vdash e[v/v'] : T'$ [induction hypothesis, 6, 7, 8]
10. $cx' \vdash \lambda v'{:}T.\ e[v/v'] : T'$ [exAbs, 9]
11. $OKsbs(e, v, v')$ [hypothesis v' fresh]
12. $v \neq v'$ [hypothesis v' fresh]
13. $v \notin \mathcal{FV}(e[v/v'])$ [Theorem 4.9, 11, 12]
14. $v' \notin \mathcal{FV}(e)$ [hypothesis v' fresh]
15. $\mathcal{CV}(e[v/v'], v') = \mathcal{CV}(e, v)$ [Theorem 4.15, 9, 12]
16. $v \notin \mathcal{CV}(e, v)$ [Theorem 4.2]
17. $v \notin \mathcal{CV}(e[v/v'], v')$ [15, 16]
18. $cx' \vdash \lambda v{:}T.\ e[v/v'][v'/v] : T \to T'$ [exAbsAlpha, 10, 13, 17]
19. $e[v/v'][v/v'] = e$ [Theorem 4.14, 11, 14]
20. $cx' \vdash \lambda v{:}T.\ e : T \to T'$ [18, 19]

Proof of Theorem 4.60

By induction on the derivation of the judgements in the antecedents of the implications:

tyBool)

1. $\vdash cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] : \textsc{context}$ [hypothesis]
2. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 1]
3. $\mathsf{Bool}[\overline{\alpha}/\overline{S}] = \mathsf{Bool}$
4. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \mathsf{Bool}[\overline{\alpha}/\overline{S}] : \textsc{type}$ [2, 3]

tyVar)

Let the instance of tyVar used to derive the judgement be

$$\frac{\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 : \textsc{context} \quad \beta \in \mathcal{TV}(cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3)}{cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash \beta : \textsc{type}} \quad (0)$$

The judgement $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \beta[\overline{\alpha}/\overline{S}] : \textsc{type}$ is derived as follows:

1. $\vdash cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] : \textsc{context}$ [hypothesis]
2. $\beta \in \mathcal{TV}(cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3)$ [0]
3. $\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 : \textsc{context}$ [0]

If $\beta \in \mathcal{TV}(cx_1)$:

4. $\beta \notin \overline{\alpha}$ [Theorem 4.40, 3]
5. $\beta[\overline{\alpha}/\overline{S}] = \beta$ [Theorem 4.28, 4]
6. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \beta[\overline{\alpha}/\overline{S}] : \textsc{type}$ [tyVar, 1, hypothesis $\beta \in \mathcal{TV}(cx_1)$, 5]

If $\beta = \alpha_i$:

4. $\beta[\overline{\alpha}/\overline{S}] = S_i$
5. $cx_1, cx_2 \vdash S_i : \textsc{type}$ [hypothesis]
6. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \beta[\overline{\alpha}/\overline{S}] : \textsc{type}$ [Theorem 4.59, 5, 1, 4]

If $\beta \in \mathcal{TV}(cx_3)$:

4. $\beta \notin \overline{\alpha}$ [Theorem 4.40, 3]
5. $\beta \in \mathcal{TV}(cx_3[\overline{\alpha}/\overline{S}])$
6. $\beta[\overline{\alpha}/\overline{S}] = \beta$ [Theorem 4.28, 4]
7. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \beta[\overline{\alpha}/\overline{S}] : \textsc{type}$ [tyVar, 1, 5, 6]

tyArr)

Let the instance of tyArr used to derive the judgement be

$$\frac{cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{var}\ v{:}T_1 \vdash T_2 : \textsc{type}}{cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash T_1 \to T_2 : \textsc{type}} \quad (0)$$

The judgement $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash (T_1 \to T_2)[\overline{\alpha}/\overline{S}] : \textsc{type}$ is derived as follows:

1. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{var}\ v{:}T_1 \vdash T_2 : \textsc{type}$ [0]
2. $\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{var}\ v{:}T_1 : \textsc{context}$ [Theorem 4.37, 1]
3. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash T_1 : \textsc{type}$ [Theorem 4.44, 2]
4. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash T_1[\overline{\alpha}/\overline{S}] : \textsc{type}$ [induction hypothesis, 3]
5. $v \notin \mathcal{V}(cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3)$ [Theorem 4.41, 2]

If $v \notin \mathcal{V}(cx_2)$:

6. $v \notin \mathcal{V}(cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}])$ [5]
7. $\vdash cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}], \mathsf{var}\ v{:}T_1[\overline{\alpha}/\overline{S}] : \textsc{context}$ [cxVdec, 4, 6]
8. $OKsbs(\mathsf{var}\ v{:}T_1, \overline{\alpha}, \overline{S})$ [definition of $OKsbs$]
9. $OKsbs(cx_3, \mathsf{var}\ v{:}T_1, \overline{\alpha}, \overline{S})$ [8, hypothesis $OKsbs(cx_3, \overline{\alpha}, \overline{S})$]
10. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}], \mathsf{var}\ v{:}T_1[\overline{\alpha}/\overline{S}] \vdash T_2[\overline{\alpha}/\overline{S}] : \textsc{type}$ [induction hypothesis, 9, 7, 1]

11. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash T_1[\overline{\alpha}/\overline{S}] \to T_2[\overline{\alpha}/\overline{S}] : \textsc{type}$ [tyArr, 10]

If $v \in \mathcal{V}(cx_2)$, pick a fresh variable v':

6. $cx_1, \overline{\alpha}, cx_3, \mathsf{var}\ v' : T_1 \vdash T_2 : \textsc{type}$ [Theorem 4.58, 1]
7. $\vdash cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}], \mathsf{var}\ v' : T_1[\overline{\alpha}/\overline{S}] : \textsc{context}$ [cxVdec, 4]
8. $OKsbs(\mathsf{var}\ v' : T_1, \overline{\alpha}, \overline{S})$ [definition of $OKsbs$]
9. $OKsbs(cx_3, \mathsf{var}\ v' : T_1, \overline{\alpha}, \overline{S})$ [8, hypothesis $OKsbs(cx_3, \overline{\alpha}, \overline{S})$]
10. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}], \mathsf{var}\ v' : T_1[\overline{\alpha}/\overline{S}] \vdash T_2[\overline{\alpha}/\overline{S}] : \textsc{type}$ [induction hypothesis, 9, 7, 6]
11. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash T_1[\overline{\alpha}/\overline{S}] \to T_2[\overline{\alpha}/\overline{S}] : \textsc{type}$ [tyArr, 10]

stArr)

Let the instance of stArr used to derive the judgement be

$$\frac{\begin{array}{c} cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{var}\ v : T \vdash T_1 \prec_r T_2 \\ v' \neq v'' \\ r' = \lambda v' : T \to T_2.\ \forall v'' : T.\ r\ (v'\ v'') \end{array}}{cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash T \to T_1 \prec_{r'} T \to T_2} \quad (0)$$

The judgement $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash T \to T_1 \prec_{r'} T \to T_2$ is derived as follows:

1. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{var}\ v : T \vdash T_1 \prec_r T_2$ [0]
2. $\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{var}\ v : T : \textsc{context}$ [Theorem 4.37, 1]
3. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash T : \textsc{type}$ [Theorem 4.44]
4. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash T[\overline{\alpha}/\overline{S}] : \textsc{type}$ [induction hypothesis, 3]
5. $v \notin \mathcal{V}(cx_1, cx_3)$ [Theorem 4.41, 2]

If $v \notin \mathcal{V}(cx_2)$:

6. $v \notin cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}]$ [5]
7. $\vdash cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}], \mathsf{var}\ v : T[\overline{\alpha}/\overline{S}] : \textsc{context}$ [cxVdec, 4, 6]
8. $OKsbs(cx_3, \overline{\alpha}, \overline{S})$ [hypothesis]
9. $\overline{\alpha} \cap \mathcal{TV}(cx_3) = \emptyset$ [8]
10. $\overline{\alpha} \cap \mathcal{TV}(cx_3, \mathsf{var}\ v : T[\overline{\alpha}/\overline{S}]) = \emptyset$ [9]
11. $\forall i.\ \ \mathcal{FTV}(S_i) \cap \mathcal{CTV}(cx_3, \alpha_i) = \emptyset$ [8]
12. $\forall i.\ \ \mathcal{CTV}(\mathsf{var}\ v : T[\overline{\alpha}/\overline{S}], \alpha_i) = \emptyset$ [definition of $\mathcal{CTV}$]
13. $\forall i.\ \ \mathcal{FTV}(S_i) \cap \mathcal{CTV}((cx_3, \mathsf{var}\ v : T[\overline{\alpha}/\overline{S}]), \alpha_i) = \emptyset$ [11, 12]
14. $OKsbs((cx_3, \mathsf{var}\ v : T[\overline{\alpha}/\overline{S}]), \overline{\alpha}, \overline{S})$ [10, 13]
15. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}], \mathsf{var}\ v : T[\overline{\alpha}/\overline{S}] \vdash T_1[\overline{\alpha}/\overline{S}] \prec_{r[\overline{\alpha}/\overline{S}]} T_2[\overline{\alpha}/\overline{S}]$ [induction hypothesis, 7, 14, 1]
16. $v' \neq v''$ [0]
17. $r' = \lambda v' : T \to T_2.\ \forall v'' : T.\ r\ (v'\ v'')$ [0]
18. $r'[\overline{\alpha}/\overline{S}] = \lambda v' : T[\overline{\alpha}/\overline{S}] \to T_2[\overline{\alpha}/\overline{S}].\ \forall v'' : T[\overline{\alpha}/\overline{S}].\ r[\overline{\alpha}/\overline{S}]\ (v'\ v'')$ [17, Theorem 4.35]
19. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash T[\overline{\alpha}/\overline{S}] \to T_1[\overline{\alpha}/\overline{S}] \prec_{r'[\overline{\alpha}/\overline{S}]} T[\overline{\alpha}/\overline{S}] \to T_2[\overline{\alpha}/\overline{S}]$ [stArr, 15, 16, 18]
20. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash (T \to T_1)[\overline{\alpha}/\overline{S}] \prec_{r'[\overline{\alpha}/\overline{S}]} (T \to T_2)[\overline{\alpha}/\overline{S}]$ [19]

If $v \in \mathcal{V}(cx_2)$, pick a fresh variable u:

6. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{var}\ u : T \vdash T_1 \prec_r T_2$ [Theorem 4.58, 1]

7. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash (T \to T_1)[\overline{\alpha}/\overline{S}] \prec_{r'[\overline{\alpha}/\overline{S}]} (T \to T_2)[\overline{\alpha}/\overline{S}]$ [analogous derivation as 6–20 above for case $v \notin \mathcal{V}(cx_2)$]

exVar)

Let the instance of exVar used to derive the judgement be

$$\frac{\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 : \textsc{context} \quad \mathsf{var}\ v{:}T \in cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3}{cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash v : T} \quad (0)$$

The judgement $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash v : T[\overline{\alpha}/\overline{S}]$ is derived as follows:

1. $\vdash cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] : \textsc{context}$ [hypothesis]
2. $\mathsf{var}\ v{:}T \in cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3$ [0]
3. $\mathsf{var}\ v{:}T \in cx_1, cx_3$ [2]

If $\mathsf{var}\ v{:}T \in cx_1$:

4. $\mathsf{var}\ v{:}T \in cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}]$
5. $cx_1, cx_3, cx_3[\overline{\alpha}/\overline{S}] \vdash v : T$ [exVar, 1, 4]
6. $\exists cx_1', cx_1''.\ \ cx_1 = cx_1', \mathsf{var}\ v{:}T, cx_1''$ [hypothesis $\mathsf{var}\ v{:}T \in cx_1$]
7. $\mathcal{FTV}(T) \subseteq \mathcal{TV}(cx_1')$ [Theorem 4.57, 1, 6]
8. $\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 : \textsc{context}$ [0]
9. $\overline{\alpha} \cap \mathcal{TV}(cx_1) = \emptyset$ [Theorem 4.40, 8]
10. $\mathcal{FTV}(T) \cap \overline{\alpha} = \emptyset$ [7, 6, 9]
11. $T[\overline{\alpha}/\overline{S}] = T$ [Theorem 4.28, 10]
12. $cx_1, cx_3, cx_3[\overline{\alpha}/\overline{S}] \vdash v : T[\overline{\alpha}/\overline{S}]$ [5, 11]

If $\mathsf{var}\ v{:}T \in cx_3$:

4. $\mathsf{var}\ v{:}T[\overline{\alpha}/\overline{S}] \in cx_3[\overline{\alpha}/\overline{S}]$
5. $\mathsf{var}\ v{:}T[\overline{\alpha}/\overline{S}] \in cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}]$ [4]
6. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash v : T[\overline{\alpha}/\overline{S}]$ [exVar, 1, 5]

exOp)

Let the instance of exOp used to derive the judgement be

$$\frac{\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 : \textsc{context} \quad \mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \quad \forall i.\ \ cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash T_i : \textsc{type}}{cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash o[\overline{T}] : T[\overline{\beta}/\overline{T}]} \quad (0)$$

The judgement $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash o[\overline{T}][\overline{\alpha}/\overline{S}] : T[\overline{\beta}/\overline{T}][\overline{\alpha}/\overline{S}]$ is derived as follows:

1. $\vdash cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] : \textsc{context}$ [hypothesis]
2. $\forall i.\ \ cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash T_i : \textsc{type}$ [0]
3. $\forall i.\ \ cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash T_i[\overline{\alpha}/\overline{S}] : \textsc{type}$ [induction hypothesis, 2]
4. $\mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3$ [0]
5. $\mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx_1, cx_3$ [4]

If $\mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx_1$:

6. $\mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}]$ [5]

7. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash o[\overline{T}[\overline{\alpha}/\overline{S}]] : T[\overline{\beta}/\overline{T}[\overline{\alpha}/\overline{S}]]$ [exOp, 1, 6, 3]
8. $o[\overline{T}[\overline{\alpha}/\overline{S}]] = o[\overline{T}][\overline{\alpha}/\overline{S}]$ [definition of $\_[\_]$]
9. $\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 : \textsc{context}$ [0]
10. $\overline{\alpha} \cap \mathcal{TV}(cx_1) = \emptyset$ [Theorem 4.40, 9]
11. $\exists cx_1', cx_1''.\ \ cx_1 = cx_1', \mathsf{op}\ o{:}\{\overline{\beta}\}\ T, cx_1''$ [hypothesis $\mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx_1$]
12. $\mathcal{FTV}(\mathsf{op}\ o{:}\{\overline{\beta}\}\ T) \subseteq \mathcal{TV}(cx_1')$ [Theorem 4.57, 1, 11]
13. $\mathcal{FTV}(\mathsf{op}\ o{:}\{\overline{\beta}\}\ T) \subseteq \mathcal{TV}(cx_1)$ [12, 11]
14. $\mathcal{FTV}(\mathsf{op}\ o{:}\{\overline{\beta}\}\ T) \cap \overline{\alpha} = \emptyset$ [10, 13]
15. $(\mathcal{FTV}(T) - \overline{\beta}) \cap \overline{\alpha} = \emptyset$ [14]
16. $T[\overline{\beta}/\overline{T}][\overline{\alpha}/\overline{S}] = T[\overline{\beta}/\overline{T}[\overline{\alpha}/\overline{S}]]$ [Theorem 4.30, 15]
17. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash o[\overline{T}][\overline{\alpha}/\overline{S}] : T[\overline{\beta}/\overline{T}][\overline{\alpha}/\overline{S}]$ [7, 8, 16]

If $\mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx_3$:

6. $\exists cx_3', cx_3''.\ \ cx_3 = cx_3', \mathsf{op}\ o{:}\{\overline{\beta}\}\ T, cx_3''$
7. $\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 : \textsc{context}$ [0]
8. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3', \mathsf{tvar}\ \overline{\beta} \vdash T : \textsc{type}$ [Theorem 4.42, 7, 6]
9. $\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3', \mathsf{tvar}\ \overline{\beta} : \textsc{context}$ [Theorem 4.37, 8]
10. $\overline{\alpha} \cap \overline{\beta} = \emptyset$ [Theorem 4.40, 9]
11. $\mathsf{op}\ o{:}\{\overline{\beta}\}\ T[\overline{\alpha}/\overline{S}] \in cx_3[\overline{\alpha}/\overline{S}]$ [10, hypothesis $\mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx_3$]
12. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash o[\overline{T}[\overline{\alpha}/\overline{S}]] : T[\overline{\alpha}/\overline{S}][\overline{\beta}/\overline{T}[\overline{\alpha}/\overline{S}]]$ [exOp, 1, 11, 3]
13. $o[\overline{T}[\overline{\alpha}/\overline{S}]] = o[\overline{T}][\overline{\alpha}/\overline{S}]$ [definition of $\_[\_]$]
14. $OKsbs(cx_3, \overline{\alpha}, \overline{S})$ [hypothesis]
15. $\forall i.\ \ \mathcal{FTV}(S_i) \cap \mathcal{CTV}(cx_3, \alpha_i) = \emptyset$ [14, definition of $OKsbs$]
16. $\forall i.\ \ \mathcal{FTV}(S_i) \cap \mathcal{CTV}(\mathsf{op}\ o{:}\{\overline{\beta}\}\ T, \alpha_i) = \emptyset$ [15, 6]
17. $\forall i.\ \ \mathcal{FTV}(S_i) \cap \begin{cases} \overline{\beta} & \text{if} \quad \alpha_i \in \mathcal{FTV}(T) - \overline{\beta} \\ \emptyset & \text{otherwise} \end{cases} = \emptyset$ [16]
18. $\forall \alpha_i \in \mathcal{FTV}(T) - \overline{\beta}.\ \ \mathcal{FTV}(S_i) \cap \overline{\beta} = \emptyset$ [17]
19. $\forall \alpha_i \in \mathcal{FTV}(T).\ \ \mathcal{FTV}(S_i) \cap \overline{\beta} = \emptyset$ [18, 10]
20. $T[\overline{\beta}/\overline{T}][\overline{\alpha}/\overline{S}] = T[\overline{\alpha}/\overline{S}][\overline{\beta}/\overline{T}[\overline{\alpha}/\overline{S}]]$ [Theorem 4.31, 10, 19]
21. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash o[\overline{T}][\overline{\alpha}/\overline{S}] : T[\overline{\beta}/\overline{T}][\overline{\alpha}/\overline{S}]$ [12, 13, 20]

exAbs)

Let the instance of exAbs used to defive the judgement be

$$\frac{cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{var}\ v{:}T \vdash e : T'}{cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash \lambda v{:}T.\ e : T \to T'} \quad (0)$$

The judgement $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash (\lambda v{:}T.\ e)[\overline{\alpha}/\overline{S}] : (T \to T')[\overline{\alpha}/\overline{S}]$ is derived as follows:

1. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{var}\ v{:}T \vdash e : T'$ [0]
2. $\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{var}\ v{:}T : \textsc{context}$ [Theorem 4.37, 1]
3. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash T : \textsc{type}$ [Theorem 4.44, 2]
4. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash T[\overline{\alpha}/\overline{S}] : \textsc{type}$ [induction hypothesis, 3]
5. $v \notin \mathcal{V}(cx_1, cx_3)$ [Theorem 4.41, 2]

If $v \notin \mathcal{V}(cx_2)$:

6. $v \notin cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}]$ [5]
7. $\vdash cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}], \mathsf{var}\ v{:}T[\overline{\alpha}/\overline{S}] : \textsc{context}$ [cxVdec, 4, 6]
8. $OKsbs(cx_3, \overline{\alpha}, \overline{S})$ [hypothesis]
9. $\overline{\alpha} \cap \mathcal{TV}(cx_3) = \emptyset$ [8]
10. $\overline{\alpha} \cap \mathcal{TV}(cx_3, \mathsf{var}\ v{:}T[\overline{\alpha}/\overline{S}]) = \emptyset$ [9]
11. $\forall i.\ \ \mathcal{FTV}(S_i) \cap \mathcal{CTV}(cx_3, \alpha_i) = \emptyset$ [8]
12. $\forall i.\ \ \mathcal{CTV}(\mathsf{var}\ v{:}T[\overline{\alpha}/\overline{S}], \alpha_i) = \emptyset$ [definition of $\mathcal{CTV}$]
13. $\forall i.\ \ \mathcal{FTV}(S_i) \cap \mathcal{CTV}((cx_3, \mathsf{var}\ v{:}T[\overline{\alpha}/\overline{S}]), \alpha_i) = \emptyset$ [11, 12]
14. $OKsbs((cx_3, \mathsf{var}\ v{:}T[\overline{\alpha}/\overline{S}]), \overline{\alpha}, \overline{S})$ [10, 13]
15. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}], \mathsf{var}\ v{:}T[\overline{\alpha}/\overline{S}] \vdash e[\overline{\alpha}/\overline{S}] : T'[\overline{\alpha}/\overline{S}]$ [induction hypothesis, 7, 14, 1]
16. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \lambda v{:}T[\overline{\alpha}/\overline{S}].\ e[\overline{\alpha}/\overline{S}] : T[\overline{\alpha}/\overline{S}] \to T'[\overline{\alpha}/\overline{S}]$ [exAbs, 15]
17. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash (\lambda v{:}T.\ e)[\overline{\alpha}/\overline{S}] : (T \to T')[\overline{\alpha}/\overline{S}]$ [16]

If $v \in \mathcal{V}(cx_2)$, pick a fresh variable v':

6. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{var}\ v'{:}T \vdash e[v/v'] : T'$ [Theorem 4.58, 1]
7. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \lambda v'{:}T[\overline{\alpha}/\overline{S}].\ e[v/v'][\overline{\alpha}/\overline{S}] : T[\overline{\alpha}/\overline{S}] \to T'[\overline{\alpha}/\overline{S}]$
[analogous derivation as 6–16 above for case $v \notin \mathcal{V}(cx_2)$]
8. v' fresh [hypothesis]
9. $OKsbs(e, v, v')$ [8]
10. $v \notin \mathcal{FV}(e[v/v'])$ [Theorem 4.9, 9, 8]
11. $v \notin \mathcal{FV}(e[v/v'][\overline{\alpha}/\overline{S}])$ [Theorem 4.19, 10]
12. $v \notin \mathcal{CV}(e, v)$ [Theorem 4.2]
13. $v \notin \mathcal{CV}(e[v/v'], v')$ [Theorem 4.15, 9, 8, 12]
14. $v \notin \mathcal{CV}(e[v/v'][\overline{\alpha}/\overline{S}], v')$ [Theorem 4.20, 13]
15. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \lambda v{:}T[\overline{\alpha}/\overline{S}].\ e[v/v'][\overline{\alpha}/\overline{S}][v'/v] : (T \to T')[\overline{\alpha}/\overline{S}]$
[exAbsAlpha, 7, 11, 14]
16. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \lambda v{:}T[\overline{\alpha}/\overline{S}].\ e[v/v'][v'/v][\overline{\alpha}/\overline{S}] : (T \to T')[\overline{\alpha}/\overline{S}]$ [Theorem 4.24, 15]
17. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \lambda v{:}T[\overline{\alpha}/\overline{S}].\ e[\overline{\alpha}/\overline{S}] : (T \to T')[\overline{\alpha}/\overline{S}]$ [Theorem 4.14, 16, 9, 8]
18. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash (\lambda v{:}T.\ e)[\overline{\alpha}/\overline{S}] : (T \to T')[\overline{\alpha}/\overline{S}]$ [17]

exIf)

Let the instance of exIf used to derive the judgement be

$$\frac{\begin{array}{c} cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash e_0 : \mathsf{Bool} \\ cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{ax}\ e_0 \vdash e_1 : T \\ cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{ax}\ \neg\, e_0 \vdash e_2 : T \\ cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash T : \textsc{type} \end{array}}{cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash \mathsf{if}\ e_0\ e_1\ e_2 : T} \quad (0)$$

The judgement $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash (\mathsf{if}\ e_0\ e_1\ e_2)[\overline{\alpha}/\overline{S}] : T[\overline{\alpha}/\overline{S}]$ is derived as follows:

1. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash e_0 : \mathsf{Bool}$ [0]
2. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash e_0[\overline{\alpha}/\overline{S}] : \mathsf{Bool}$ [induction hypothesis, 1]
3. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{ax}\ e_0 \vdash e_1 : T$ [0]
4. $\vdash cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3, \mathsf{ax}\ e_0 : \textsc{context}$ [Theorem 4.37, 3]
5. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash e_0 : \mathsf{Bool}$ [Theorem 4.43, 4]

6. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash e_0[\overline{\alpha}/\overline{S}] : \mathsf{Bool}$ [induction hypothesis, 5]
7. $\vdash cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}], \mathsf{ax}\ e_0[\overline{\alpha}/\overline{S}] : \textsc{context}$ [cxAx, 6]
8. $OKsbs(cx_3, \overline{\alpha}, \overline{S})$ [hypothesis]
9. $\forall \alpha.\ \mathcal{CTV}(\mathsf{ax}\ e_0, \alpha) = \emptyset$ [definition of $\mathcal{CTV}$]
10. $OKsbs(\mathsf{ax}\ e_0, \overline{\alpha}, \overline{S})$ [definition of $OKsbs$, 9]
11. $OKsbs((cx_3, \mathsf{ax}\ e_0), \overline{\alpha}, \overline{S})$ [8, 10]
12. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}], \mathsf{ax}\ e_0[\overline{\alpha}/\overline{S}] \vdash e_1[\overline{\alpha}/\overline{S}] : T[\overline{\alpha}/\overline{S}]$ [induction hypothesis, 11, 7, 3]
13. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}], \mathsf{ax}\ \neg\ e_0[\overline{\alpha}/\overline{S}] \vdash e_2[\overline{\alpha}/\overline{S}] : T[\overline{\alpha}/\overline{S}]$ [analogous derivation as 3–12]
14. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash T : \textsc{type}$ [0]
15. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash T[\overline{\alpha}/\overline{S}] : \textsc{type}$ [induction hypothesis, 14]
16. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash \mathsf{if}\ e_0[\overline{\alpha}/\overline{S}]\ e_1[\overline{\alpha}/\overline{S}]\ e_2[\overline{\alpha}/\overline{S}] : T[\overline{\alpha}/\overline{S}]$ [exIf, 2, 12, 13, 15]

thAx)
Analogous to exOp.

thIfSubst)
Analogous to exIf.

thAbs)
Let the instance of thAbs used to derive the judgement be

$$\frac{\begin{array}{c} cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash (\lambda v{:}T.\ e)\ e' : \ldots \\ OKsbs(e, v, e') \end{array}}{cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash (\lambda v{:}T.\ e)\ e' \equiv e[v/e']} \quad (0)$$

The judgement $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash ((\lambda v{:}T.\ e)\ e' \equiv e[v/e'])[\overline{\alpha}/\overline{S}]$ is derived as follows:

1. $cx_1, \mathsf{tvar}\ \overline{\alpha}, cx_3 \vdash (\lambda v{:}T.\ e)\ e' : \ldots$ [0]
2. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash ((\lambda v{:}T.\ e)\ e')[\overline{\alpha}/\overline{S}] : \ldots$ [induction hypothesis, 1]
3. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash (\lambda v{:}T[\overline{\alpha}/\overline{S}].\ e[\overline{\alpha}/\overline{S}])\ e'[\overline{\alpha}/\overline{S}] : \ldots$ [2]
4. $OKsbs(e, v, e')$ [0]
5. $\mathcal{FV}(e') \cap \mathcal{CV}(e, v) = \emptyset$ [4]
6. $\mathcal{FV}(e'[\overline{\alpha}/\overline{S}]) \cap \mathcal{CV}(e[\overline{\alpha}/\overline{S}], v) = \emptyset$ [Theorem 4.19, Theorem 4.20]
7. $OKsbs(e[\overline{\alpha}/\overline{S}], v, e'[\overline{\alpha}/\overline{S}])$ [6]
8. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash (\lambda v{:}T[\overline{\alpha}/\overline{S}].\ e[\overline{\alpha}/\overline{S}])\ e'[\overline{\alpha}/\overline{S}] \equiv e[\overline{\alpha}/\overline{S}][v/e'[\overline{\alpha}/\overline{S}]]$ [thAbs, 3, 7]
9. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash (\lambda v{:}T[\overline{\alpha}/\overline{S}].\ e[\overline{\alpha}/\overline{S}])\ e'[\overline{\alpha}/\overline{S}] \equiv e[v/e'][\overline{\alpha}/\overline{S}]$ [Theorem 4.23, 8]
10. $cx_1, cx_2, cx_3[\overline{\alpha}/\overline{S}] \vdash ((\lambda v{:}T.\ e)\ e' \equiv e[v/e'])[\overline{\alpha}/\overline{S}]$ [9]

thIf)
Analogous to exIf.

all other rules)
Analogous to tyBool: starting with the instance of the rule used to derive the judgement in the antecedent of the implication, we replace $\mathsf{tvar}\ \overline{\alpha}$ with cx_2 and $\overline{\alpha}$ with $\overline{S}$ in the premises of the rule instance and then we apply the rule to the transformed premises to obtain the conclusion with $\overline{\alpha}$ replaced with $\overline{S}$. We use Theorem 4.19 for tyRestr. We use Theorem 4.35 for stRefl, stRec, exThe, thBool, thExt, thRec, and thProjSub. We use Theorem 4.19, Theorem 4.20, and Theorem 4.24 for exAbsAlpha.

Proof of Theorem 4.61

1. $cx \vdash T : \textsc{type}$ [hypothesis]
2. $\vdash cx, cx' : \textsc{context}$ [hypothesis]
3. $cx, cx' \vdash T : \textsc{type}$ [Theorem 4.59, 1, 2]
4. $v \notin \mathcal{V}(cx, cx')$ [hypothesis]
5. $\vdash cx, cx', \mathsf{var}\ v{:}T : \textsc{context}$ [cxVdec, 3, 4]

Proof of Theorem 4.62

1. $\vdash cx : \textsc{context}$ [hypothesis]
2. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 1]
3. $\vdash cx, \mathsf{var}\ v{:}\mathsf{Bool} : \textsc{context}$ [cxVdec, 2, hypothesis]

Proof of Theorem 4.63

1. $cx \vdash e : \mathsf{Bool}$ [hypothesis]
2. $\vdash cx, \mathsf{ax}\ e : \textsc{context}$ [cxAx, 1]

Proof of Theorem 4.64

1. $cx \vdash T_1 : \textsc{type}$ [hypothesis]

Pick a variable $v \in \mathcal{N} - \mathcal{V}(cx)$:

2. $\vdash cx, \mathsf{var}\ v{:}T_1 : \textsc{context}$ [cxVdec, 1]
3. $cx \vdash T_2 : \textsc{type}$ [hypothesis]
4. $cx, \mathsf{var}\ v{:}T_1 \vdash T_2 : \textsc{type}$ [Theorem 4.59, 1, 3]
5. $cx \vdash T_1 \to T_2 : \textsc{type}$ [tyArr, 4]

Proof of Theorem 4.65

Pick a variable $\widetilde{v} \in \mathcal{N} - (\mathcal{V}(cx) \cup \{v\})$:

1. $cx \vdash T : \textsc{type}$ [hypothesis]
2. $\vdash cx, \mathsf{var}\ \widetilde{v}{:}T : \textsc{context}$ [cxVdec, 1, hypothesis $\widetilde{v} \notin \mathcal{V}(cx)$]
3. $cx, \mathsf{var}\ \widetilde{v}{:}T \vdash \widetilde{v} : T$ [exVar, 2]
4. $cx \vdash \lambda\widetilde{v}{:}T.\ \widetilde{v} : T \to T$ [exAbs, 3]
5. $cx \vdash \lambda v{:}T.\ v : T \to T$ [exAbsAlpha, 4, hypothesis $\widetilde{v} \neq v$]

Proof of Theorem 4.66

1. $\vdash cx : \textsc{context}$ [hypothesis]
2. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 1]
3. $cx \vdash \lambda v{:}\mathsf{Bool}.\ v : \mathsf{Bool} \to \mathsf{Bool}$ [exId, 2]

Proof of Theorem 4.67

1. $\vdash cx : \textsc{context}$ [hypothesis]
2. $cx \vdash \lambda\gamma{:}\mathsf{Bool}.\ \gamma : \mathsf{Bool} \to \mathsf{Bool}$ [exIdBool, 1]
3. $cx \vdash (\lambda\gamma{:}\mathsf{Bool}.\ \gamma \equiv \lambda\gamma{:}\mathsf{Bool}.\ \gamma) : \mathsf{Bool}$ [exEq, 2, 2]

Proof of Theorem 4.68

Pick a variable $\widetilde{v} \in \mathcal{N} - (\mathcal{V}(cx) \cup \{v\})$:

1. $cx \vdash T : \textsc{type}$ [hypothesis]
2. $\vdash cx, \mathsf{var}\ \widetilde{v}{:}T : \textsc{context}$ [cxVdec, 1, hypothesis $\widetilde{v} \notin \mathcal{V}(cx)$]
3. $cx, \mathsf{var}\ \widetilde{v}{:}T \vdash \mathsf{true} : \mathsf{Bool}$ [exTrue, 2]
4. $cx \vdash \lambda\widetilde{v}{:}T.\ \mathsf{true} : T \to \mathsf{Bool}$ [exAbs, 3]
5. $cx \vdash \lambda v{:}T.\ \mathsf{true} : T \to \mathsf{Bool}$ [exAbsAlpha, 4, hypothesis $\widetilde{v} \neq v$]

Proof of Theorem 4.69

1. $\vdash cx : \textsc{context}$ [hypothesis]
2. $cx \vdash \lambda\gamma{:}\mathsf{Bool}.\ \gamma : \mathsf{Bool} \to \mathsf{Bool}$ [exIdBool, 1]
3. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 1]
4. $cx \vdash \lambda\gamma{:}\mathsf{Bool}.\ \mathsf{true} : \mathsf{Bool} \to \mathsf{Bool}$ [exConstTrue, 3]
5. $cx \vdash (\lambda\gamma{:}\mathsf{Bool}.\ \gamma \equiv \lambda\gamma{:}\mathsf{Bool}.\ \mathsf{true}) : \mathsf{Bool}$ [exEq, 2, 4]

Proof of Theorem 4.70

Pick a variable $\widetilde{\gamma} \in \mathcal{N} - (\mathcal{V}(cx) \cup \{\gamma\})$:

1. $\vdash cx : \textsc{context}$ [hypothesis]
2. $\vdash cx, \mathsf{var}\ \widetilde{\gamma}{:}\mathsf{Bool} : \textsc{context}$ [cxVdecBool, 1, hypothesis $\widetilde{\gamma} \notin \mathcal{V}(cx)$]
3. $cx, \mathsf{var}\ \widetilde{\gamma}{:}\mathsf{Bool} \vdash \widetilde{\gamma} : \mathsf{Bool}$ [exVar, 2]
4. $cx, \mathsf{var}\ \widetilde{\gamma}{:}\mathsf{Bool} \vdash \mathsf{false} : \mathsf{Bool}$ [exFalse, 2]
5. $cx, \mathsf{var}\ \widetilde{\gamma}{:}\mathsf{Bool} \vdash \widetilde{\gamma} \equiv \mathsf{false} : \mathsf{Bool}$ [exEq, 3, 4]
6. $cx \vdash \lambda\widetilde{\gamma}{:}\mathsf{Bool}.\ (\widetilde{\gamma} \equiv \mathsf{false}) : \mathsf{Bool} \to \mathsf{Bool}$ [exAbs, 5]
7. $cx \vdash \lambda\gamma{:}\mathsf{Bool}.\ (\gamma \equiv \mathsf{false}) : \mathsf{Bool} \to \mathsf{Bool}$ [exAbsAlpha, 6, hypothesis $\widetilde{\gamma} \neq \gamma$]

Proof of Theorem 4.71

1. $cx \vdash e : \mathsf{Bool}$ [hypothesis]
2. $\vdash cx : \textsc{context}$ [Theorem 4.37, 1]
3. $cx \vdash \neg : \mathsf{Bool} \to \mathsf{Bool}$ [exNot, 2]
4. $cx \vdash \neg\ e : \mathsf{Bool}$ [exApp, 3, 1]

Proof of Theorem 4.72

1. $cx \vdash e_1 : \mathsf{Bool}$ [hypothesis]
2. $cx \vdash \neg\ e_1 : \mathsf{Bool}$ [exNeg, 1]
3. $\vdash cx, \mathsf{ax}\ \neg\ e_1 : \textsc{context}$ [cxAx0, 2]
4. $cx, \mathsf{ax}\ \neg\ e_1 \vdash \mathsf{false} : \mathsf{Bool}$ [exFalse, 3]
5. $cx, \mathsf{ax}\ e_1 \vdash e_2 : \mathsf{Bool}$ [hypothesis]
6. $\vdash cx : \textsc{context}$ [Theorem 4.37, 1]
7. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 6]
8. $cx \vdash \mathsf{if}\ e_1\ e_2\ \mathsf{false} : \mathsf{Bool}$ [exIf, 1, 5, 4, 7]

Proof of Theorem 4.73

1. $cx \vdash e_1 : \mathsf{Bool}$ [hypothesis]
2. $cx \vdash \neg\ e_1 : \mathsf{Bool}$ [exNeg, 1]
3. $\vdash cx, \mathsf{ax}\ \neg\ e_1 : \textsc{context}$ [cxAx0, 2]
4. $cx, \mathsf{ax}\ \neg\ e_1 \vdash \mathsf{false} : \mathsf{Bool}$ [exFalse, 3]
5. $cx \vdash e_2 : \mathsf{Bool}$ [hypothesis]
6. $\vdash cx, \mathsf{ax}\ e_1 : \textsc{context}$ [cxAx0, 1]
7. $cx, \mathsf{ax}\ e_1 \vdash e_2 : \mathsf{Bool}$ [Theorem 4.59, 5, 6]
8. $\vdash cx : \textsc{context}$ [Theorem 4.37, 1]
9. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 8]
10. $cx \vdash \mathsf{if}\ e_1\ e_2\ \mathsf{false} : \mathsf{Bool}$ [exIf, 1, 7, 4, 9]

Proof of Theorem 4.74

1. $cx \vdash e_1 : \mathsf{Bool}$ [hypothesis]
2. $\vdash cx, \mathsf{ax}\ e_1 : \textsc{context}$ [cxAx0, 1]
3. $cx, \mathsf{ax}\ e_1 \vdash \mathsf{true} : \mathsf{Bool}$ [exTrue, 2]
4. $cx, \mathsf{ax}\ \neg\ e_1 \vdash e_2 : \mathsf{Bool}$ [hypothesis]
5. $\vdash cx : \textsc{context}$ [Theorem 4.37, 1]
6. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 5]
7. $cx \vdash \mathsf{if}\ e_1\ \mathsf{true}\ e_2 : \mathsf{Bool}$ [exIf, 1, 3, 4, 6]

Proof of Theorem 4.75

1. $cx \vdash e_1 : \mathsf{Bool}$ [hypothesis]
2. $\vdash cx, \mathsf{ax}\ e_1 : \textsc{context}$ [cxAx0, 1]
3. $cx, \mathsf{ax}\ e_1 \vdash \mathsf{true} : \mathsf{Bool}$ [exTrue, 2]
4. $cx \vdash e_2 : \mathsf{Bool}$ [hypothesis]
5. $cx \vdash \neg\ e_1 : \mathsf{Bool}$ [exNeg, 1]
6. $\vdash cx, \mathsf{ax} \neg\ e_1 : \textsc{context}$ [cxAx0, 5]
7. $cx, \mathsf{ax} \neg\ e_1 \vdash e_2 : \mathsf{Bool}$ [Theorem 4.59, 4, 6]
8. $\vdash cx : \textsc{context}$ [Theorem 4.37, 1]
9. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 8]
10. $cx \vdash \mathsf{if}\ e_1\ \mathsf{true}\ e_2 : \mathsf{Bool}$ [exIf, 1, 3, 7, 9]

Proof of Theorem 4.76

1. $cx \vdash e_1 : \mathsf{Bool}$ [hypothesis]
2. $cx \vdash \neg\ e_1 : \mathsf{Bool}$ [exNeg, 1]
3. $\vdash cx, \mathsf{ax} \neg\ e_1 : \textsc{context}$ [cxAx0, 2]
4. $cx, \mathsf{ax} \neg\ e_1 \vdash \mathsf{true} : \mathsf{Bool}$ [exTrue, 3]
5. $cx, \mathsf{ax}\ e_1 \vdash e_2 : \mathsf{Bool}$ [hypothesis]
6. $\vdash cx : \textsc{context}$ [Theorem 4.37, 1]
7. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 6]
8. $cx \vdash \mathsf{if}\ e_1\ e_2\ \mathsf{true} : \mathsf{Bool}$ [exIf, 1, 5, 4, 7]

Proof of Theorem 4.77

1. $cx \vdash e_1 : \mathsf{Bool}$ [hypothesis]
2. $cx \vdash \neg\ e_1 : \mathsf{Bool}$ [exNeg, 1]
3. $\vdash cx, \mathsf{ax} \neg\ e_1 : \textsc{context}$ [cxAx0, 2]
4. $cx, \mathsf{ax} \neg\ e_1 \vdash \mathsf{true} : \mathsf{Bool}$ [exTrue, 3]
5. $cx \vdash e_2 : \mathsf{Bool}$ [hypothesis]
6. $\vdash cx, \mathsf{ax}\ e_1 : \textsc{context}$ [cxAx0, 1]
7. $cx, \mathsf{ax}\ e_1 \vdash e_2 : \mathsf{Bool}$ [Theorem 4.59, 5, 6]
8. $\vdash cx : \textsc{context}$ [Theorem 4.37, 1]
9. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 8]
10. $cx \vdash \mathsf{if}\ e_1\ e_2\ \mathsf{true} : \mathsf{Bool}$ [exIf, 1, 7, 4, 9]

Proof of Theorem 4.78

We first prove the derived rule

$$\frac{\begin{array}{c}\vdash cx, \mathsf{var}\ v : \mathsf{Bool} : \textsc{context} \\ v' \neq v\end{array}}{cx, \mathsf{var}\ v : \mathsf{Bool} \vdash \lambda v' : \mathsf{Bool}.\ v \equiv v' : \mathsf{Bool} \to \mathsf{Bool}} \quad (*)$$

Pick a variable $\widetilde{v}' \in \mathcal{N} - (\mathcal{V}(cx) \cup \{v, v'\})$:

1. $\vdash cx, \mathsf{var}\ v : \mathsf{Bool} : \textsc{context}$ [hypothesis]
2. $\vdash cx, \mathsf{var}\ v : \mathsf{Bool}, \mathsf{var}\ \widetilde{v}' : \mathsf{Bool} : \textsc{context}$ [cxVdecBool, 1, hypothesis $\widetilde{v}' \notin \mathcal{V}(cx) \cup \{v\}$]
3. $cx, \mathsf{var}\ v : \mathsf{Bool}, \mathsf{var}\ \widetilde{v}' : \mathsf{Bool} \vdash v : \mathsf{Bool}$ [exVar, 2]
4. $cx, \mathsf{var}\ v : \mathsf{Bool}, \mathsf{var}\ \widetilde{v}' : \mathsf{Bool} \vdash \widetilde{v}' : \mathsf{Bool}$ [exVar, 2]
5. $cx, \mathsf{var}\ v : \mathsf{Bool}, \mathsf{var}\ \widetilde{v}' : \mathsf{Bool} \vdash v \equiv \widetilde{v}' : \mathsf{Bool}$ [exEq, 3, 4]
6. $cx, \mathsf{var}\ v : \mathsf{Bool} \vdash \lambda \widetilde{v}' : \mathsf{Bool}.\ v \equiv \widetilde{v}' : \mathsf{Bool} \to \mathsf{Bool}$ [exAbs, 5]
7. $cx, \mathsf{var}\ v : \mathsf{Bool} \vdash \lambda v' : \mathsf{Bool}.\ v \equiv v' : \mathsf{Bool} \to \mathsf{Bool}$ [exAbsAlpha, 6, hypothesis $\widetilde{v}' \neq v'$]

This concludes the proof of rule $*$.

We now prove exIff. Pick a variable $\widetilde{\gamma} \in \mathcal{N} - (\mathcal{V}(cx) \cup \{\gamma, \gamma'\})$:

1. $\vdash cx : \textsc{context}$ [hypothesis]
2. $\vdash cx, \mathsf{var}\ \widetilde{\gamma} : \mathsf{Bool} : \textsc{context}$ [cxVdecBool, 1, hypothesis $\widetilde{\gamma} \notin \mathcal{V}(cx)$]
3. $cx, \mathsf{var}\ \widetilde{\gamma} : \mathsf{Bool} \vdash \lambda \gamma' : \mathsf{Bool}.\ \widetilde{\gamma} \equiv \gamma' : \mathsf{Bool} \to \mathsf{Bool}$ [$*$, 2, $\widetilde{\gamma} \neq \gamma'$]
4. $cx \vdash \lambda \widetilde{\gamma} : \mathsf{Bool}.\ \lambda \gamma' : \mathsf{Bool}.\ \widetilde{\gamma} \equiv \gamma' : \mathsf{Bool} \to \mathsf{Bool} \to \mathsf{Bool}$ [exAbs, 3]
5. $cx \vdash \lambda \gamma : \mathsf{Bool}.\ \lambda \gamma' : \mathsf{Bool}.\ \gamma \equiv \gamma' : \mathsf{Bool} \to \mathsf{Bool} \to \mathsf{Bool}$ [exAbsAlpha, 4, hypothesis $\widetilde{\gamma} \neq \gamma$]

Proof of Theorem 4.79

1. $cx \vdash e_1 : \mathsf{Bool}$ [hypothesis]
2. $\vdash cx : \textsc{context}$ [Theorem 4.37, 1]
3. $cx \vdash \Leftrightarrow : \mathsf{Bool} \to \mathsf{Bool} \to \mathsf{Bool}$ [exIff, 2]
4. $cx \vdash \Leftrightarrow\ e_1 : \mathsf{Bool} \to \mathsf{Bool}$ [exApp, 3, 1]
5. $cx \vdash e_2 : \mathsf{Bool}$ [hypothesis]
6. $cx \vdash \Leftrightarrow\ e_1\ e_2 : \mathsf{Bool}$ [exApp, 4, 5]

Proof of Theorem 4.80

1. $cx \vdash e_1 : T$ [hypothesis]
2. $cx \vdash e_2 : T$ [hypothesis]
3. $cx \vdash e_1 \equiv e_2 : \mathsf{Bool}$ [exEq, 1, 2]
4. $cx \vdash \neg\ (e_1 \equiv e_2) : \mathsf{Bool}$ [exNeg, 3]

Proof of Theorem 4.81

Pick a variable $\widetilde{\psi} \in \mathcal{N} - (\mathcal{V}(cx) \cup \{\psi\})$:

1. $cx \vdash T : \textsc{type}$ [hypothesis]
2. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 1]
3. $cx \vdash T \to \mathsf{Bool} : \textsc{type}$ [tyArr', 1, 2]
4. $\vdash cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} : \textsc{context}$ [cxVdec, 3, hypothesis $\widetilde{\psi} \notin \mathcal{V}(cx)$]
5. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \widetilde{\psi} : T \to \mathsf{Bool}$ [exVar, 4]
6. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash T : \textsc{type}$ [Theorem 4.59, 1, 4]
7. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \lambda\gamma{:}T.\ \mathsf{true} : T \to \mathsf{Bool}$ [exConstTrue, 6]
8. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \widetilde{\psi} \equiv \lambda\gamma{:}T.\ \mathsf{true} : \mathsf{Bool}$ [exEq, 5, 7]
9. $cx \vdash \lambda\widetilde{\psi}{:}T \to \mathsf{Bool}.\ (\widetilde{\psi} \equiv \lambda\gamma{:}T.\ \mathsf{true}) : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exAbs, 8]
10. $cx \vdash \lambda\psi{:}T \to \mathsf{Bool}.\ (\psi \equiv \lambda\gamma{:}T.\ \mathsf{true}) : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exAbsAlpha, 9, hypothesis $\widetilde{\psi} \neq \psi$]

Proof of Theorem 4.82

1. $cx, \mathsf{var}\ v{:}T \vdash e : \mathsf{Bool}$ [hypothesis]
2. $\vdash cx, \mathsf{var}\ v{:}T : \textsc{context}$ [Theorem 4.37, 1]
3. $cx \vdash \lambda v{:}T.\ e : T \to \mathsf{Bool}$ [exAbs, 1]
4. $cx \vdash T : \textsc{type}$ [Theorem 4.44, 2]
5. $cx \vdash \forall_T : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exFA, 4]
6. $cx \vdash \forall_T\ (\lambda v{:}T.\ e) : \mathsf{Bool}$ [exApp, 5, 3]

Proof of Theorem 4.83

Let $\widetilde{\psi}, \widetilde{\gamma} \in \mathcal{N}$ such that $\widetilde{\psi} \neq \widetilde{\gamma}$ and $\{\widetilde{\psi}, \widetilde{\gamma}\} \cap (\mathcal{V}(cx) \cup \{\psi, \gamma\}) = \emptyset$. The rule is derived as follows:

1. $cx \vdash T : \textsc{type}$ [hypothesis]
2. $\vdash cx : \textsc{context}$ [Theorem 4.37, 1]
3. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 2]
4. $cx \vdash T \to \mathsf{Bool} : \textsc{type}$ [tyArr', 1, 3]
5. $\vdash cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} : \textsc{context}$ [cxVdec, 4, hypothesis $\widetilde{\psi} \notin \mathcal{V}(cx)$]
6. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash T : \textsc{type}$ [Theorem 4.59, 1, 5]
7. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \forall_T : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exFA, 6]
8. $\vdash cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T : \textsc{context}$ [cxVdec, 6, hypothesis $\widetilde{\gamma} \notin \mathcal{V}(cx) \cup \{\widetilde{\psi}\}$]
9. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \widetilde{\psi} : T \to \mathsf{Bool}$ [exVar, 8]
10. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \widetilde{\gamma} : T$ [exVar, 8]

11. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \widetilde{\psi}\ \widetilde{\gamma} : \mathsf{Bool}$ [exApp, 9, 10]
12. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \neg\ (\widetilde{\psi}\ \widetilde{\gamma}) : \mathsf{Bool}$ [exNeg, 11]
13. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \lambda\widetilde{\gamma}{:}T.\ \neg\ (\widetilde{\psi}\ \widetilde{\gamma}) : T \to \mathsf{Bool}$ [exAbs, 12]
14. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \lambda\gamma{:}T.\ \neg\ (\widetilde{\psi}\ \gamma) : T \to \mathsf{Bool}$ [exAbsAlpha, 13]
15. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \forall\gamma{:}T.\ \neg\ (\widetilde{\psi}\ \gamma) : \mathsf{Bool}$ [exApp, 7, 14]
16. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \neg\ (\forall\gamma{:}T.\ \neg\ (\widetilde{\psi}\ \gamma)) : \mathsf{Bool}$ [exNeg, 15]
17. $cx \vdash \lambda\widetilde{\psi}{:}T \to \mathsf{Bool}.\ \neg\ (\forall\gamma{:}T.\ \neg\ (\widetilde{\psi}\ \gamma)) : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exAbs, 16]
18. $cx \vdash \lambda\psi{:}T \to \mathsf{Bool}.\ \neg\ (\forall\gamma{:}T.\ \neg\ (\psi\ \gamma)) : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exAbsAlpha, 17]

Proof of Theorem 4.84

Analogous to Theorem 4.82, using exEX instead of exFA.

Proof of Theorem 4.85

Let $\widetilde{\psi}, \widetilde{\gamma}, \widetilde{\gamma}' \in \mathcal{N}$ such that they are distinct and $\{\widetilde{\psi}, \widetilde{\gamma}, \widetilde{\gamma}'\} \cap (\mathcal{V}(cx) \cup \{\psi, \gamma, \gamma'\}) = \emptyset$. The rule is derived as follows:

1. $cx \vdash T : \text{TYPE}$ [hypothesis]
2. $\vdash cx : \text{CONTEXT}$ [Theorem 4.37, 1]
3. $cx \vdash \mathsf{Bool} : \text{TYPE}$ [tyBool, 2]
4. $cx \vdash T \to \mathsf{Bool} : \text{TYPE}$ [tyArr', 1, 3]
5. $\vdash cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} : \text{CONTEXT}$ [cxVdec, 4]
6. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash T : \text{TYPE}$ [Theorem 4.59, 1, 5]
7. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \exists_T : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exEX, 6]
8. $\vdash cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T : \text{CONTEXT}$ [cxVdec, 6]
9. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \widetilde{\psi} : T \to \mathsf{Bool}$ [exVar, 8]
10. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \widetilde{\gamma} : T$ [exVar, 8]
11. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \widetilde{\psi}\ \widetilde{\gamma} : \mathsf{Bool}$ [exApp, 9, 10]
12. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash T : \text{TYPE}$ [Theorem 4.59, 6, 8]
13. $\vdash cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T, \mathsf{var}\ \widetilde{\gamma}'{:}T : \text{CONTEXT}$ [cxVdec, 12]
14. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T, \mathsf{var}\ \widetilde{\gamma}'{:}T \vdash \widetilde{\psi} : T \to \mathsf{Bool}$ [exVar, 13]
15. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T, \mathsf{var}\ \widetilde{\gamma}'{:}T \vdash \widetilde{\gamma} : T$ [exVar, 13]
16. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T, \mathsf{var}\ \widetilde{\gamma}'{:}T \vdash \widetilde{\gamma}' : T$ [exVar, 13]
17. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T, \mathsf{var}\ \widetilde{\gamma}'{:}T \vdash \widetilde{\gamma}' \equiv \widetilde{\gamma} : \mathsf{Bool}$ [exEq, 16, 15]
18. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T, \mathsf{var}\ \widetilde{\gamma}'{:}T \vdash \widetilde{\psi}\ \widetilde{\gamma}' : \mathsf{Bool}$ [exApp, 14, 16]

19. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T, \mathsf{var}\ \widetilde{\gamma}'{:}T \vdash \widetilde{\psi}\ \widetilde{\gamma}' \Rightarrow \widetilde{\gamma}' \equiv \widetilde{\gamma} : \mathsf{Bool}$ [exImpl0, 18, 17]
20. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \forall_T : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exFA, 12]
21. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \lambda\widetilde{\gamma}'{:}T.\ (\widetilde{\psi}\ \widetilde{\gamma}' \Rightarrow \widetilde{\gamma}' \equiv \widetilde{\gamma}) : T \to \mathsf{Bool}$ [exAbs, 19]
22. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \lambda\gamma'{:}T.\ (\widetilde{\psi}\ \gamma' \Rightarrow \gamma' \equiv \widetilde{\gamma}) : T \to \mathsf{Bool}$ [exAbsAlpha, 21]
23. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \forall\gamma'{:}T.\ (\widetilde{\psi}\ \gamma' \Rightarrow \gamma' \equiv \widetilde{\gamma}) : \mathsf{Bool}$ [exApp, 20, 22]
24. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool}, \mathsf{var}\ \widetilde{\gamma}{:}T \vdash \widetilde{\psi}\ \widetilde{\gamma} \wedge \forall\gamma'{:}T.\ (\widetilde{\psi}\ \gamma' \Rightarrow \gamma' \equiv \widetilde{\gamma}) : \mathsf{Bool}$ [exConj0, 11, 23]
25. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \exists_T : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exEX, 6]
26. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \lambda\widetilde{\gamma}{:}T.\ (\widetilde{\psi}\ \widetilde{\gamma} \wedge \forall\gamma'{:}T.\ (\widetilde{\psi}\ \gamma' \Rightarrow \gamma' \equiv \widetilde{\gamma})) : T \to \mathsf{Bool}$ [exAbs, 24]
27. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \lambda\gamma{:}T.\ (\widetilde{\psi}\ \gamma \wedge \forall\gamma'{:}T.\ (\widetilde{\psi}\ \gamma' \Rightarrow \gamma' \equiv \gamma)) : T \to \mathsf{Bool}$ [exAbsAlpha, 26]
28. $cx, \mathsf{var}\ \widetilde{\psi}{:}T \to \mathsf{Bool} \vdash \exists\gamma{:}T.\ (\widetilde{\psi}\ \gamma \wedge \forall\gamma'{:}T.\ (\widetilde{\psi}\ \gamma' \Rightarrow \gamma' \equiv \gamma)) : \mathsf{Bool}$ [exApp, 25, 27]
29. $cx \vdash \lambda\widetilde{\psi}{:}T \to \mathsf{Bool}.\ \exists\gamma{:}T.\ (\widetilde{\psi}\ \gamma \wedge \forall\gamma'{:}T.\ (\widetilde{\psi}\ \gamma' \Rightarrow \gamma' \equiv \gamma)) : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exAbs, 28]
30. $cx \vdash \lambda\psi{:}T \to \mathsf{Bool}.\ \exists\gamma{:}T.\ (\psi\ \gamma \wedge \forall\gamma'{:}T.\ (\psi\ \gamma' \Rightarrow \gamma' \equiv \gamma)) : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exAbsAlpha, 29]

Proof of Theorem 4.86

Analogous to Theorem 4.82, using exEX1 instead of exFA.

Proof of Theorem 4.87

1. $cx, \mathsf{var}\ v{:}T \vdash e : \mathsf{Bool}$ [hypothesis]
2. $cx \vdash \lambda v{:}T.\ e : T \to \mathsf{Bool}$ [exAbs, 1]
3. $v' \notin \mathcal{FV}(e) \cup \mathcal{CV}(e, v)$ [hypothesis]
4. $cx \vdash \lambda v'{:}T.\ e[v/v'] : T \to \mathsf{Bool}$ [exAbsAlpha, 2, 3]
5. $\vdash cx, \mathsf{var}\ v{:}T : \textsc{context}$ [Theorem 4.37, 1]
6. $cx \vdash T : \textsc{type}$ [Theorem 4.44, 5]
7. $cx \vdash \forall_T : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exFA, 6]
8. $cx \vdash \forall v'{:}T.\ e[v/v'] : \mathsf{Bool}$ [exApp, 7, 4]

Proof of Theorem 4.88

1. $cx \vdash \prod_i f_i\ T_i : \textsc{type}$ [hypothesis]
2. $cx \vdash \mathsf{proj}\ f_j : \prod_i f_i\ T_i \to T_j$ [exProj, 1]
3. $cx \vdash e : \prod_i f_i\ T_i$ [hypothesis]
4. $cx \vdash \mathsf{proj}\ f_j\ e : T_j$ [exApp, 2, 3]

Proof of Theorem 4.89

The conjunction of this theorem and Theorem 4.90 is proved by induction on the derivation of the antecedents of the two conjoined implications. Here we present the cases for the subtyping rules, while the cases for the well-typedness rules are presented in the next subsection.

stRestr)

1. $cx \vdash T|r : \text{TYPE}$ [premise of stRestr]
2. $cx \vdash r : T \to \mathsf{Bool}$ [Theorem 4.49, 1]
3. $cx \vdash T \to \mathsf{Bool} : \text{TYPE}$ [induction hypothesis (Theorem 4.90), 2]
4. $cx \vdash T : \text{TYPE}$ [Theorem 4.47, 3]
5. $cx \vdash T|r : \text{TYPE} \quad \wedge \quad cx \vdash T : \text{TYPE} \quad \wedge \quad cx \vdash r : T \to \mathsf{Bool}$ [1, 4, 2]

stRefl)

1. $cx \vdash T : \text{TYPE}$ [premise of stRefl]
2. $cx \vdash \lambda v{:}T.\ \mathsf{true} : T \to \mathsf{Bool}$ [exConstTrue, 1]
3. $cx \vdash T : \text{TYPE} \quad \wedge \quad cx \vdash \lambda v{:}T.\ \mathsf{true} : T \to \mathsf{Bool}$ [1, 2]

stArr)

1. $cx \vdash T_1 \prec_r T_2$ [premise of stArr]
2. $cx \vdash T_1 : \text{TYPE}$ [induction hypothesis, 1]
3. $cx \vdash T_2 : \text{TYPE}$ [induction hypothesis, 1]
4. $cx \vdash r : T_2 \to \mathsf{Bool}$ [induction hypothesis, 1]
5. $cx \vdash T : \text{TYPE}$ [premise of stArr]
6. $cx \vdash T \to T_1 : \text{TYPE}$ [tyArr', 5, 2]
7. $cx \vdash T \to T_2 : \text{TYPE}$ [tyArr', 5, 3]

Pick a variable $u \in \mathcal{N} - (\mathcal{V}(cx) \cup \{v, v'\})$ (where v and v' are the variables used in the instance of stArr):

8. $\vdash cx, \mathsf{var}\ u{:}T \to T_2 : \text{CONTEXT}$ [cxVdec, 7]
9. $cx, \mathsf{var}\ u{:}T \to T_2 \vdash T : \text{TYPE}$ [Theorem 4.59, 8, 5]

Pick a variable $u' \in \mathcal{N} - (\mathcal{V}(cx) \cup \{v, v', u\})$:

10. $\vdash cx, \mathsf{var}\ u{:}T \to T_2, \mathsf{var}\ u'{:}T : \text{CONTEXT}$ [cxVdec, 9]
11. $cx, \mathsf{var}\ u{:}T \to T_2, \mathsf{var}\ u'{:}T \vdash u : T \to T_2$ [exVar, 10]
12. $cx, \mathsf{var}\ u{:}T \to T_2, \mathsf{var}\ u'{:}T \vdash u' : T$ [exVar, 10]
13. $cx, \mathsf{var}\ u{:}T \to T_2, \mathsf{var}\ u'{:}T \vdash u\ u' : T_2$ [exApp, 11, 12]
14. $cx, \mathsf{var}\ u{:}T \to T_2, \mathsf{var}\ u'{:}T \vdash r : T_2 \to \mathsf{Bool}$ [Theorem 4.59, 4, 10]
15. $cx, \mathsf{var}\ u{:}T \to T_2, \mathsf{var}\ u'{:}T \vdash r\ (u\ u') : \mathsf{Bool}$ [exApp, 14, 13]
16. $cx, \mathsf{var}\ u{:}T \to T_2 \vdash \lambda u'{:}T.\ r\ (u\ u') : T \to \mathsf{Bool}$ [exAbs, 15]
17. $\mathcal{FV}(r) = \emptyset$ [Theorem 4.53, 1]
18. $v' \notin \mathcal{FV}(r\ (u\ u'))$ [17, hypothesis $u \neq v'$, hypothesis $u' \neq v'$]
19. $\mathcal{CV}(r\ (u\ u'), u')$
    $=$
    $\mathcal{CV}(r, u') \cup \mathcal{CV}(u, u') \cup \mathcal{CV}(u', u')$
    $=$ [Theorem 4.1, 17, definition of $\mathcal{CV}$]
    $\emptyset$

20. $v' \notin \mathcal{CV}(r\ (u\ u'), u')$ [19]
21. $cx, \mathsf{var}\ u{:}T \to T_2 \vdash \lambda v'{:}T.\ r\ (u\ v') : T \to \mathsf{Bool}$ [exAbsAlpha, 16, 18, 20]
22. $cx, \mathsf{var}\ u{:}T \to T_2 \vdash \forall_T : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exFA, 9]
23. $cx, \mathsf{var}\ u{:}T \to T_2 \vdash \forall v'{:}T.\ r\ (u\ v') : \mathsf{Bool}$ [exApp, 22, 21]
24. $cx \vdash \lambda u{:}T \to T_2.\ \forall v'{:}T.\ r\ (u\ v') : (T \to T_2) \to \mathsf{Bool}$ [exAbs, 23]
25. $\mathcal{FV}(\forall v'{:}T.\ r\ (u\ v')) = \{u\}$ [Theorem 4.32, 17, hypothesis $u \neq v'$]
26. $v \notin \mathcal{FV}(\forall v'{:}T.\ r\ (u\ v'))$ [25, hypothesis $u \neq v$]
27. $\mathcal{CV}(\forall v'{:}T.\ r\ (u\ v'), u)$
    $=$ [Theorem 4.34, 25]
    $\{v'\} \cup \mathcal{CV}(r\ (u\ v'), u)$
    $=$ [Theorem 4.1, 17, definition of $\mathcal{CV}$]
    $\{v'\}$
28. $v \neq v'$ [premise of stArr]
29. $v \notin \mathcal{CV}(\forall v'{:}T.\ r\ (u\ v'), u)$ [27, 28]
30. $cx \vdash \lambda v{:}T \to T_2.\ \forall v'{:}T.\ r\ (v\ v') : (T \to T_2) \to \mathsf{Bool}$ [exAbsAlpha, 24, 26, 29]
31. $cx \vdash T \to T_1 : \textsc{type} \quad \wedge \quad cx \vdash T \to T_2 : \textsc{type} \quad \wedge$
    $cx \vdash \lambda v{:}T \to T_2.\ \forall v'{:}T.\ r\ (v\ v') : (T \to T_2) \to \mathsf{Bool}$ [6, 7, 30]

stRec)

1. $cx \vdash \prod_i f_i\ T_i : \textsc{type}$ [premise of stRec]
2. $\vdash cx : \textsc{context}$ [Theorem 4.37, 1]
3. $\forall i.\ \ cx \vdash T_i \prec_{r_i} T_i'$ [premise of stRec]
4. $\forall i.\ \ cx \vdash T_i' : \textsc{type}$ [induction hypothesis, 3]
5. $cx \vdash \prod_i f_i\ T_i' : \textsc{type}$ [tyRec, 2, 4]

Pick a variable $u \in \mathcal{N} - (\mathcal{V}(cx) \cup \{v\})$ (where v is the variable used in the instance of stRec):

6. $\vdash cx, \mathsf{var}\ u{:}\prod_i f_i\ T_i' : \textsc{context}$ [cxVdec, 5]
7. $cx, \mathsf{var}\ u{:}\prod_i f_i\ T_i' \vdash \prod_i f_i\ T_i' : \textsc{type}$ [Theorem 4.59, 5, 6]
8. $cx, \mathsf{var}\ u{:}\prod_i f_i\ T_i' \vdash u : \prod_i f_i\ T_i'$ [exVar, 6]
9. $\forall i.\ \ cx, \mathsf{var}\ u{:}\prod_i f_i\ T_i' \vdash u.f_i : T_i'$ [exDotProj0, 7, 8]
10. $\forall i.\ \ cx \vdash r_i : T_i' \to \mathsf{Bool}$ [induction hypothesis, 3]
11. $\forall i.\ \ cx, \mathsf{var}\ u{:}\prod_i f_i\ T_i' \vdash r_i : T_i' \to \mathsf{Bool}$ [Theorem 4.59, 10, 6]
12. $\forall i.\ \ cx, \mathsf{var}\ u{:}\prod_i f_i\ T_i' \vdash r_i\ u.f_i : \mathsf{Bool}$ [exApp, 11, 9]
13. $cx, \mathsf{var}\ u{:}\prod_i f_i\ T_i' \vdash \mathsf{true} : \mathsf{Bool}$ [exTrue, 6]
14. $cx, \mathsf{var}\ u{:}\prod_i f_i\ T_i' \vdash \bigwedge_i r_i\ u.f_i : \mathsf{Bool}$ [13[7], exConj0, 12]
15. $cx \vdash \lambda u{:}\prod_i f_i\ T_i'.\ \bigwedge_i r_i\ u.f_i : \prod_i f_i\ T_i' \to \mathsf{Bool}$ [exAbs, 14]
16. $\forall i.\ \ \mathcal{FV}(r_i) = \emptyset$ [Theorem 4.53, 3]
17. $\mathcal{FV}(\bigwedge_i r_i\ u.f_i) = \{u\}$ [Theorem 4.32, 16]
18. $v \notin \mathcal{FV}(\bigwedge_i r_i\ u.f_i)$ [17, hypothesis $u \neq v$]
19. $\mathcal{CV}(\bigwedge_i r_i\ u.f_i, u)$
    $=$ [Theorem 4.34]
    $\bigcup_i \mathcal{CV}(r_i, u) \cup \mathcal{CV}(u, u)$
    $=$ [Theorem 4.1, 16, definition of $\mathcal{CV}$]
    $\emptyset$

[7] Recall that $\bigwedge_i e_i = \mathsf{true}$ if $n = 0$

20. $v \notin \mathcal{CV}(\bigwedge_i r_i\ u.f_i, u)$ [19]
21. $cx \vdash \lambda v{:}\prod_i f_i\ T'_i.\ \bigwedge_i r_i\ v.f_i : \prod_i f_i\ T_i \to \mathsf{Bool}$ [exAbsAlpha, 15, 18, 20]
22. $cx \vdash \prod_i f_i\ T_i : \textsc{type} \quad \wedge \quad cx \vdash \prod_i f_i\ T'_i : \textsc{type} \quad \wedge$
$cx \vdash \lambda v{:}\prod_i f_i\ T'_i.\ \bigwedge_i r_i\ v.f_i : \prod_i f_i\ T_i \to \mathsf{Bool}$ [1, 5, 21]

Proof of Theorem 4.90

The conjunction of this theorem and Theorem 4.89 is proved by induction on the derivation of the antecedents of the two conjoined implications. Here we present the cases for the well-typedness rules, while the cases for the subtyping rules are presented in the previous subsection.

exVar)

1. $\mathsf{var}\ v{:}T \in cx$ [premise of exVar]
2. $\exists cx_1, cx_2.\ \ cx = cx_1, \mathsf{var}\ v{:}T, cx_2$ [1]
3. $\vdash cx : \textsc{context}$ [premise of exVar]
4. $cx_1 \vdash T : \textsc{type}$ [Theorem 4.44, 3, 2]
5. $cx \vdash T : \textsc{type}$ [Theorem 4.59, 4, 3, 2]

exOp)

1. $\mathsf{op}\ o{:}\{\overline{\beta}\}\ T \in cx$ [premise of exOp]
2. $\exists cx_1, cx_2.\ \ cx = cx_1, \mathsf{op}\ o{:}\{\overline{\beta}\}\ T, cx_2$ [1]
3. $\vdash cx : \textsc{context}$ [premise of exOp]
4. $cx_1, \mathsf{tvar}\ \overline{\beta} \vdash T : \textsc{type}$ [Theorem 4.42, 3, 2]
5. $\forall i.\ \ cx \vdash T_i : \textsc{type}$ [premise of exOp]
6. $OKsbs(\epsilon, \overline{\beta}, \overline{T})$ [definition of $OKsbs$]
7. $cx \vdash T[\overline{\beta}/\overline{T}] : \textsc{type}$ [Theorem 4.60, 5, 6, 3, 2, 4]

exApp)

1. $cx \vdash e_1 : T_1 \to T_2$ [premise of exApp]
2. $cx \vdash T_1 \to T_2 : \textsc{type}$ [induction hypothesis, 1]
3. $cx \vdash T_2 : \textsc{type}$ [Theorem 4.47, 2]

exAbs)

1. $cx, \mathsf{var}\ v{:}T \vdash e : T'$ [premise of exAbs]
2. $cx, \mathsf{var}\ v{:}T \vdash T' : \textsc{type}$ [induction hypothesis, 1]
3. $cx \vdash T \to T' : \textsc{type}$ [tyArr, 2]

exEq)

1. $cx \vdash e_1 : T$ [premise of exEq]
2. $\vdash cx : \textsc{context}$ [Theorem 4.37, 1]
3. $cx \vdash \mathsf{Bool} : \textsc{type}$ [tyBool, 2]

exIf)

$cx \vdash T : \textsc{type}$ [premise of exIf]

exThe)

1. $cx \vdash T : \text{TYPE}$ [premise of exThe]
2. $cx \vdash \exists!_T : (T \to \mathsf{Bool}) \to \mathsf{Bool}$ [exEX1, 1]
3. $\mathcal{FV}(\exists!_T) = \emptyset$ [Theorem 4.32]
4. $cx \vdash (T \to \mathsf{Bool})|\exists!_T : \text{TYPE}$ [tyRestr, 2, 3]
5. $cx \vdash (T \to \mathsf{Bool})|\exists!_T \to T : \text{TYPE}$ [tyArr', 4, 1]

exProj)

1. $cx \vdash \prod_i f_i\ T_i : \text{TYPE}$ [premise of exProj]
2. $cx \vdash T_j : \text{TYPE}$ [Theorem 4.48, 1]
3. $cx \vdash \prod_i f_i\ T_i \to T_j : \text{TYPE}$ [tyArr', 1, 2]

exSuper)

1. $cx \vdash T \prec_r T'$ [premise of exSuper]
2. $cx \vdash T' : \text{TYPE}$ [induction hypothesis (Theorem 4.89), 1]

exSub)

1. $cx \vdash T \prec_r T'$ [premise of exSub]
2. $cx \vdash T : \text{TYPE}$ [induction hypothesis (Theorem 4.89), 1]

exAbsAlpha)

1. $cx \vdash \lambda v{:}T.\ e : T'$ [premise of exAbsAlpha]
2. $cx \vdash T' : \text{TYPE}$ [induction hypothesis, 1]

Proof of Theorem 4.91

1. $cx \vdash e : \prod_i f_i\ T_i$ [hypothesis]
2. $cx \vdash \prod_i f_i\ T_i : \text{TYPE}$ [Theorem 4.90, 1]
3. $cx \vdash e.f_j : T_j$ [exDotProj0, 2, 1]

Proof of Theorem 4.92

1. $\vdash cx : \text{CONTEXT}$ [hypothesis]
2. $cx \vdash \lambda\gamma{:}\mathsf{Bool}.\ \gamma : \mathsf{Bool} \to \mathsf{Bool}$ [exIdBool, 1]
3. $cx \vdash \lambda\gamma{:}\mathsf{Bool}.\ \gamma \equiv \lambda\gamma{:}\mathsf{Bool}.\ \gamma$ [thRefl, 2]

Proof of Theorem 4.93

1. $cx \vdash e \equiv \mathsf{true}$ [hypothesis]
2. $\vdash cx : \text{CONTEXT}$ [Theorem 4.37, 1]
3. $cx \vdash \mathsf{true}$ [thTrue, 2]
4. $cx \vdash \mathsf{true} \equiv e$ [thSymm, 1]
5. $cx \vdash e$ [thSubst, 3, 4]

Proof of Theorem 4.94

1. $\forall v{:}T.\ e = (\lambda\psi{:}T \to \mathsf{Bool}.\ (\psi \equiv \lambda\gamma{:}T.\ \mathsf{true}))\ (\lambda v{:}T.\ e)$ [abbreviation]
2. $cx \vdash \forall v{:}T.\ e : \mathsf{Bool}$ [hypothesis]
3. $cx \vdash (\lambda\psi{:}T \to \mathsf{Bool}.\ (\psi \equiv \lambda\gamma{:}T.\ \mathsf{true}))\ (\lambda v{:}T.\ e) : \mathsf{Bool}$ [2, 1]
4. $\mathcal{CV}(\psi \equiv \lambda\gamma{:}T.\ \mathsf{true}, \psi) = \emptyset$ [Theorem 4.32]
5. $OKsbs(\psi \equiv \lambda\gamma{:}T.\ \mathsf{true}, \psi, \lambda v{:}T.\ e)$ [4]
6. $cx \vdash (\lambda\psi{:}T \to \mathsf{Bool}.\ (\psi \equiv \lambda\gamma{:}T.\ \mathsf{true}))\ (\lambda v{:}T.\ e) \equiv (\lambda v{:}T.\ e \equiv \lambda\gamma{:}T.\ \mathsf{true})$ [thAbs, 3, 5]
7. $cx \vdash \forall v{:}T.\ e \equiv (\lambda v{:}T.\ e \equiv \lambda\gamma{:}T.\ \mathsf{true})$ [6, 1]
8. $cx \vdash \forall v{:}T.\ e$ [hypothesis]
9. $cx \vdash \lambda v{:}T.\ e \equiv \lambda\gamma{:}T.\ \mathsf{true}$ [thSubst, 8, 7]
10. $cx \vdash e' : T$ [hypothesis]
11. $cx \vdash e' \equiv e'$ [thRefl, 10]
12. $cx \vdash \lambda v{:}T.\ e : \ldots$ [Theorem 4.51, 3]
13. $\exists T'.\ \ cx \vdash \lambda v{:}T.\ e : T \to T'$ [Theorem 4.52, 12]
14. $cx \vdash (\lambda v{:}T.\ e)\ e' : \ldots$ [exApp, 13, 10]
15. $cx \vdash (\lambda v{:}T.\ e)\ e' \equiv (\lambda\gamma{:}T.\ \mathsf{true})\ e'$ [thAppSubst, 14, 9, 11]
16. $OKsbs(e, v, e')$ [hypothesis]
17. $cx \vdash (\lambda v{:}T.\ e)\ e' \equiv e[v/e']$ [thAbs, 14, 16]
18. $cx \vdash T : \mathrm{TYPE}$ [Theorem 4.90, 10]
19. $cx \vdash \lambda\gamma{:}T.\ \mathsf{true} : T \to \mathsf{Bool}$ [exConstTrue, 18]
20. $cx \vdash (\lambda\gamma{:}T.\ \mathsf{true})\ e' : \ldots$ [exApp, 19, 10]
21. $\mathcal{CV}(\mathsf{true}, \gamma) = \emptyset$ [Theorem 4.34]
22. $OKsbs(\mathsf{true}, \gamma, e')$ [21]
23. $cx \vdash (\lambda\gamma{:}T.\ \mathsf{true})\ e' \equiv \mathsf{true}$ [thAbs, 20, 22]
24. $cx \vdash e[v/e'] \equiv (\lambda v{:}T.\ e)\ e'$ [thSymm, 17]
25. $cx \vdash e[v/e'] \equiv (\lambda\gamma{:}T.\ \mathsf{true})\ e'$ [thTrans, 24, 15]
26. $cx \vdash e[v/e'] \equiv \mathsf{true}$ [thTrans, 25, 23]
27. $cx \vdash e[v/e']$ [thEqTrue, 26]

Acknowledgments

Peter Homeier has carefully reviewed this document and has provided very useful feedback and suggestions. He has also formalized the Metaslang logic in the HOL theorem prover and has used HOL to prove many of the (meta) theorems contained in this document.

References

[1] Kestrel Institute and Kestrel Technology LLC. *Specware 4.2 Language Manual*. Available at www.specware.org.

[2] Alessandro Coglio. The semantics of Metaslang. To appear.

[3] Peter Andrews. *An Introduction to Mathematical Logic and Type Theory: To Thruth Through Proof*. Academic Press, 1986.

[4] *The HOL System Description*, July 1997.

[5] Sam Owre and Natarajan Shankar. The formal semantics of PVS. Technical Report CSL–97–2R, SRI International, August 1997. Revised March 1999.