Map theory with classical maps

Klaus Grue

June 14, 2001

Abstract

Introduction: Map theory (MT) is a seamless integration of a programming language and a classical axiomatic theory. The language is Turing complete and the theory has power like set theory. The version of MT presented here is substantially simpler than the original version from 1992.

Contents: The paper contains • a short overview of MT, • a comprehensive presentation of MT and its fundamental proof techniques, and • 380 kilobytes of handwritten formal proofs suited for getting started with MT on automatic proof systems. Among other, the formal proofs prove the consistency of ZFC set theory within MT.

Aims: Some aims of MT are listed in the following.

Correctness proofs: The ability to express programs and properties about programs in the same formalism simplifies correctness proofs. The classical parts of MT allow to use direct classical proofs in place of roundabout intuitionistic ones.

Semantics: The ability to quantify over non-countable sets simplifies definitions of distributed, parallel programming languages.

Numerical analysis: The ability to quantify over the continuum of real numbers simplifies the formulation of properties and correctness proofs for numerical algorithms.

Physics: Work is in progress on formalizing quantum electrodynamics (QED) in MT, the aim being to have QED and simulation programs for QED expressed in the same framework. The situation today is that QED is ultimately expressed in set theory whereas simulation programs are expressed in Fortran and similar languages.

Mathematics: The free combination of recursion and choice in MT simplifies mathematical proofs that have an algorithmic flavor. As an example, one can well-order any set by recursive choice. Also, the development of mathematical analysis is expected to benefit from the ability to mix in algorithms in formal proofs.

Foundations: MT draws its power from the concept of "classicality", which is a defined concept in MT. The definition of classicality says at least as much about the universe of ZFC as all the axioms of ZFC together. As opposed to ZFC, MT makes the "limitation of size" concept explicit. This leads to a strengthening of transfinite induction and might lead to new insight into set theory.

Teaching: MT has been used for teaching mathematics to first year computer science students at the University of Copenhagen in place of a traditional discrete mathematics course. Using MT as a foundation allows the students to learn more mathematics relevant to computer science with less effort. The students do not learn MT, but they learn to manipulate programs in ways that are impossible in set theory.

Contents

Chapter 1

Introduction

Physics and computer science are two fields that push the development of mathematics. The needs of physics have pushed the development of structures like differential manifolds and measure theory whereas the needs of computer science have pushed the development of structures like reduction systems and programming languages. The structures needed in physics fit well into set theory whereas set theory is less convenient for the structures needed in computer science.

For that reason, mathematics has broken in two parts. One part is based on set theory and is mainly studied in mathematical institutes. The other part is mainly studied in computer science institutes and has no common foundation.

One purpose of map theory (MT) is to allow a seamless integration of the two parts of mathematics. MT allows programs, theorems, and proofs to be expressed within a single system, which is small, homogeneous, and powerful. Formally, MT has the same power as set theory, but it enhances set theory in that it has a computer programming languages as a natural subset.

The present paper presents a new version of MT which enhances the original version from 1992. The new version has simpler axioms. Furthermore, the new version has some new axioms.

Essentially, ZFC set theory is a theory of finite sets which has been generalized to transfinite sets: The power set of a finite set is finite; hence, ZFC has a power set operator. The complement of a finite set is infinite; hence, ZFC has no complement operator.

MT is constructed the same way except that it starts with the computable functions instead of the finite sets.

Later chapters will give a traditional bottom up presentation of MT. Among other, later chapters will do the following:

- Present the syntax, axioms, and inference rules of MT.
- Represent mathematical structures like numbers and sets in MT.
- State and prove elementary theorems of MT.

- Establish convenient proof techniques like the deduction theorem.
- Prove all axioms and inference rules of ZFC within MT.

A consistency proof for the new version of MT is outside the scope of the present paper. A consistency proof can be formulated in ZFC+SI where SI is the assumption that there exists an inaccessible ordinal.

For a comparison with related work see [3].

Before diving into the details it may be convenient to get a preview of what MT looks like for the working mathematician. Working mathematicians can easily base their work on ZFC without knowing the details of the axioms, and the same is true for MT.

1.1 Acknowledgements

My thanks are due to Chantal Berline for many useful comments on the 1996 version of the present paper and the developments after that. My thanks are due to Sebastian Skalberg for his port of Appendix A to the Isabelle system.

1.2 Preview of MT

ZFC is a theory about sets and nothing but sets. To deal with numbers in ZFC one has to represent numbers by sets. As an example, one may represent natural numbers by finite von Neumann ordinals. Likewise, MT is a theory about so-called "maps" and nothing but maps. To deal with numbers in MT one has to represent numbers by maps.

For reasonable definitions of numbers and addition,

$$[\, 2+2 \equiv 4 \,]$$

is a theorem of MT. The sign $[\, \equiv \,]$ is the equal sign of MT. The theorem says that the terms $[\, 2+2 \,]$ and $[\, 4 \,]$ have the same value.

To deal with truth values in MT one has to represent truth values by maps. ZFC treats truth values as something special whereas MT treats truth values as ordinary data. From now on, $[\, \mathsf{T} \,]$ denotes a map that represents truth and $[\, \mathsf{F} \,]$ denotes a map that represents falsehood. For reasonable definitions of the membership relation $[\, \in \,]$ and the set $[\, \mathbf{N} \,]$ of natural numbers,

$$[\, 2 \in \mathbf{N} \equiv \mathsf{T} \,]$$

is a theorem of MT. The theorem says that the terms $[\, 2 \in \mathbf{N} \,]$ and $[\, \mathsf{T} \,]$ have the same value. In other words, the theorem says that $[\, 2 \in \mathbf{N} \,]$ is true. As another example, for reasonable definition of $[\, \forall \,]$, $[\, \vee \,]$, $[\, < \,]$, and $[\, x^2 \,]$,

$$[\, \forall x{\in}\mathbf{N}\colon x < 2 \vee x < x^2 \equiv \mathsf{T} \,]$$

is a theorem of MT. The theorem says that the terms [$\forall x{\in}\mathbf{N}\colon x < 2 \vee x < x^2$] and [T] have the same value. In other words, the theorem says that [$\forall x{\in}\mathbf{N}\colon x < 2 \vee x < x^2$] is true. Note that [$\forall x{\in}\mathbf{N}\colon x < 2 \vee x < x^2$] is a term. In ZFC it would have been a well-formed formula. The parenthesis in

[$(\forall x{\in}\mathbf{N}\colon x < 2 \vee x < x^2) \equiv \mathsf{T}$]

emphasizes the structure of the theorem. See Appendix C for the priority of the operators of MT.

In general, all formulas (i.e. well-formed formulas) of MT have the form [$s \equiv t$] where [s] and [t] are terms. To prove that some statement [s] is true one has to prove [$s \equiv \mathsf{T}$].

For reasonable definitions of [$=$],

[$\forall x{\in}\mathbf{N}\colon \forall y{\in}\mathbf{N}\colon x + y = y + x \equiv \mathsf{T}$]

is a theorem of MT. [$x = y$] has value [T] if [x] and [y] represent the same number. [$x = y$] and [$x \equiv y$] differ in several ways:

- If [s] and [t] are terms then [$s = t$] is a term whereas [$x \equiv y$] is a formula.
- [$s = t$] is a defined concept, [$s \equiv t$] is built into MT.

A formula is said to either *hold* or *fail* (all words and phrases in italics occur in the index at the end of this paper). As examples,

[$2 + 2 \equiv 4$] holds, and

[$2 + 2 \equiv 5$] fails.

That [$2 + 2 \equiv 4$] holds means that [$2 + 2 \equiv 4$] holds in the "standard model" (a precise definition of the "standard model" is outside the scope of the present paper). This may also be written

[$\models 2 + 2 \equiv 4$].

That [$2 + 2 \equiv 5$] fails may be written

[$\not\models 2 + 2 \equiv 5$].

A formula with free variables holds if it holds for all values of free variables and fails otherwise. As an example,

[$\not\models 2 + x \equiv 4$]

because [$2 + x \equiv 4$] fails for at least one value of [x].

hold
fail

That [$2 + 2 \equiv 4$] is provable in MT will be written

$$[\vdash 2 + 2 \equiv 4]$$

and that [$2 + 2 \equiv 5$] is disprovable in MT will be written

$$[\not\vdash 2 + 2 \equiv 5].$$

The notion of disprovability will be elaborated later.

A formula with free variables is disprovable in MT if it is disprovable for at least one assignment of values to the free variables. As an example, [$2 + x \equiv 4$] is disprovable in MT:

$$[\not\vdash 2 + x \equiv 4].$$

The power of MT forces it to be incomplete by Gödels incompleteness theorem. Therefore, one can find formulas [$\mathcal{A}$] of MT for which neither [$\vdash \mathcal{A}$] nor [$\not\vdash \mathcal{A}$]. In contrast, any formula either holds or fails. Here are some relations between [$\vdash$], [$\models$], [$\not\vdash$], and [$\not\models$]:

$$\begin{array}{lllll} [& \vdash \mathcal{A} & \Rightarrow & \models \mathcal{A} &] \\ [& \not\vdash \mathcal{A} & \Rightarrow & \not\models \mathcal{A} &] \\ [& \models \mathcal{A} & \Leftrightarrow & \neg \not\models \mathcal{A} &] \end{array}$$

It is possible to define a model of MT in ZFC+SI where SI is the assumption that there exists an inaccessible ordinal. Hence, the three statements above about [$\vdash$], [$\models$], [$\not\vdash$], and [$\not\models$] are theorems of ZFC+SI. The consistency of MT follows from the three statements above, so the consistency of MT is also a theorem of ZFC+SI. A definition of a model of MT and a consistency proof for MT is outside the scope of the present paper.

Recursive definitions are legal in MT. As an example,

$$\left[n! \doteq n = 0 \begin{cases} 1 \\ n \cdot (n-1)! \end{cases} \right]$$

defines the faculty function. The construct

$$\left[x \begin{cases} y \\ z \end{cases} \right]$$

satisfies the equations

$$\left[\mathsf{T} \begin{cases} y \\ z \end{cases} \equiv y \right], \text{ and}$$

$$\left[\mathsf{F} \begin{cases} y \\ z \end{cases} \equiv z \right].$$

The construct

$$\left[x \left\{ \begin{array}{c} y \\ z \end{array} \right. \right]$$

is typically written

[if x then y else z]

in programming languages.

The definition of [$n!$] allows to prove theorems like

[$\forall n{\in}\mathbf{N}{:}\, n! \in \mathbf{N} \equiv \mathsf{T}$] and

[$\forall n{\in}\mathbf{N}{:}\, n! \geq n \equiv \mathsf{T}$]

in MT. The proofs are by induction in n.

Computation of [$(-1)!$] proceeds thus:

$$\left[\, (-1)! \overset{+}{\rightarrow} (-1) \cdot (-2)! \overset{+}{\rightarrow} (-1) \cdot (-2) \cdot (-3)! \overset{+}{\rightarrow} \cdots \right]$$

where $\left[\, \overset{+}{\rightarrow} \right]$ reads "reduces to". The computation never ends. Likewise, computation of [$(-2)!$] never ends:

$$\left[\, (-2)! \overset{+}{\rightarrow} (-2) \cdot (-3)! \overset{+}{\rightarrow} (-2) \cdot (-3) \cdot (-4)! \overset{+}{\rightarrow} \cdots \right]$$

Two terms whose computation never end are equal in MT. As an example,

[$\models (-1)! \equiv (-2)!$].

Hence, in MT, all terms whose computation never end have the same value. That value is denoted [$\perp$]:

[$\models (-1)! \equiv \perp$], and

[$\models (-2)! \equiv \perp$].

Actually, [$(-1)! \equiv \perp$] and [$(-2)! \equiv \perp$] are provable in MT.

The convention to equate terms whose computations never terminate is counter-intuitionistic. MT is classical of nature because of this convention. However, lots of phenomena of intuitionistic logic occur in MT because, informally, [$\perp$] of MT resembles falsehood in intuitionistic logic. However, MT is classical of nature because it has only one value for never-ending computations.

If [$\mathbf{B}$] is the set of truth values (Booleans), then

[$\vdash \forall x{\in}\mathbf{B}{:}\, x \vee \neg x \equiv \mathsf{T}$].

The theorem expresses the law of excluded middle. In contrast,

[$\nvdash x \vee \neg x \equiv \mathsf{T}$]

because [$x \vee \neg x \equiv \mathsf{T}$] differs from [T] when [$x \equiv \bot$]. The value of [$\bot \vee \neg\bot$] is [$\bot$]:

$$[\ \vdash \bot \vee \neg\bot \equiv \bot\ ].$$

MT has finitely many fundamental constructs and infinitely many variables. All terms of MT are built from variables and fundamental constructs. Some of the fundamental constructs are *computable*. All terms that are built from variables and computable, fundamental constructs are said to be computable. [$2+2$] and [4] and [$(-1)!$] are examples of computable terms. [$\forall n {\in} \mathbf{N}\colon n = n$] is an example of a non-computable term. For computable terms [$\mathcal{A}$] with no free variables,

[$(\models \mathcal{A} \equiv \mathsf{T}) \Rightarrow (\vdash \mathcal{A} \equiv \mathsf{T})$] holds in ZFC+SI.

In contrast,

[$(\models \mathcal{A} \equiv \bot) \Rightarrow (\vdash \mathcal{A} \equiv \bot)$] fails in ZFC+SI.

The universal quantifier is an example of a non-computable term of MT. Another one is Hilberts epsilon operator [7]. Under reasonable conditions,

$$[\ \varepsilon x.\mathcal{A}\ ]$$

denotes a value of [x] for which [$\mathcal{A} \equiv \mathsf{T}$]. Non-computable terms can be used freely in recursive definitions. As an example, if [S] is any set then

$$[\ f(\alpha, S) \doteq \varepsilon x.x \in S \setminus \textstyle\bigcup_{\beta\in\alpha} f(\beta, S)\ ]$$

defines a one-to-one mapping from an initial segment of the ordinals onto S. If [δ] is the least ordinal for which

$$[\ S \setminus \textstyle\bigcup_{\beta\in\alpha} f(\beta, S) = \emptyset\ ]$$

then [$f(\alpha, S)$] defines an isomorphism between [δ] and [S] (in MT, the existence of [δ] is provable by transfinite induction in [S]). In other words, [$f(\alpha, S)$] defines a well-ordering of [S].

As mentioned before, one purpose of MT is to provide a seamless integration of two hitherto separate parts of mathematics. The seamless integration comes into full play in the example above where the epsilon of classical mathematics is combined with the free recursion from programming languages.

computable construct

1.3 The basis of MT

As mentioned before, ZFC is a generalization of a theory of finite sets. Likewise, MT is a generalization of a theory of computable functions.

To make a theory like MT one has to choose a theory of computable functions to generalize. The choice made for MT is to base it on untyped lambda calculus with ur-elements which wee shall denote $[\boxed{\lambda U}]$ (all mathematical constructs in boxes occur in the index at the end of this paper).

One goal of MT was to make a theory based on computable functions with at least the same power as ZFC. So far, λU is the only theory of computable functions for which the generalization to a theory of that power has succeeded.

Once λU has been chosen as the basis for MT, one has to decide which ur-elements it should contain. One could choose to include the natural numbers as ur-elements or one could choose to include the truth values $[\,\mathsf{T}\,]$ and $[\,\mathsf{F}\,]$ as ur-elements. Both choices generalize to theories with ZFC-power.

Concerning the power of the resulting theory it is unimportant which objects are included as ur-elements, but the complexity of the axiom system increases with the number of ur-elements. Therefore it is convenient to keep the number of ur-elements as low as possible.

It is now time to apply Occams razor and minimize the number of ur-elements. λU with two ur-element (e.g. $[\,\mathsf{T}\,]$ and $[\,\mathsf{F}\,]$) generalizes to a theory with ZFC-power. λU with zero ur-elements (i.e. pure lambda calculus) does not seem to generalize to a powerful theory, but λU with just one ur-element does. Therefore, MT is based on λU with just one ur-element.

It may be a surprise that a theory based on functions needs ur-elements when ZFC seemingly does fine without. ZFC, however, has two "hidden" ur-elements, namely "truth" and "falsehood". The universe of ZFC is built up in stages starting with the empty set, and the empty set is just an abstraction of falsehood. Without falsehood there would be no empty set and the universe of ZFC would be void.

The universe of MT is also built up in stages. At the first stage one will find the ur-element. At the second stage one will find the function that maps anything to the ur-element. The stages of MT form a transfinite hierarchy.

The ur-element of MT has no meaning per se, but it certainly has a purpose: It is the stating point in building up the universe of MT just like $[\,0\,]$ is the starting point for making natural numbers, the empty list is the starting point for making lists, and the empty set is the starting point for making sets.

It is convenient to have a name for the ur-element, so let us call it *nil* and let us denote it $[\boxed{\mathsf{N}}]$.

λU with one ur-element will be referred to as $[\boxed{\lambda\mathsf{N}}]$. Chapters 2–7 present the details of $\lambda\mathsf{N}$. Chapter 8 presents MT as a generalization of $\lambda\mathsf{N}$.

The elements of the universe of MT will be referred to as *maps*. The word "map" is chosen because it is used more or less as a synonym for "function"

nil
map

in mathematical analysis. The ur-element is a map but it is not a function, so "map" and "function" are not completely synonyms.

1.4 Definition of $\lambda\mathsf{N}$

The role of $\lambda\mathsf{N}$ in MT is to "mediate" between computer science and classical mathematics. For that reason, $\lambda\mathsf{N}$ has features that computer scientists will appreciate and mathematicians will find hard to follow and vice versa. At first glance, $\lambda\mathsf{N}$ is just an ordinary reduction system in the spirit of untyped λ-calculus, but at a crucial step, $\lambda\mathsf{N}$ breaks with computer science tradition and becomes classical of nature.

This is $\lambda\mathsf{N}$ presented in computer science terms:

Syntax

$$\begin{array}{lllll}
\text{variables } \mathcal{V}: & [\,\mathcal{V} & ::= & x \mid y \mid z \mid \cdots &] \\
\text{terms } \mathcal{T}: & [\,\mathcal{T} & ::= & \mathcal{V} \mid \mathsf{N} \mid \lambda\mathcal{V}.\mathcal{T} \mid \mathcal{T}\,'\,\mathcal{T} \mid \text{if}(\mathcal{T},\mathcal{T},\mathcal{T}) &]
\end{array}$$

Reduction rules

$$\begin{array}{llll}
[\,\mathsf{N}\,'\,z & \stackrel{+}{\rightarrow} & z &] \\
[\,(\lambda x.y)\,'\,z & \stackrel{+}{\rightarrow} & \langle y \mid x{:=}z\rangle &] \quad \beta\text{-reduction with } \alpha\text{-renaming} \\
[\,\langle \mathsf{N} \mid u{:=}v\rangle & \stackrel{+}{\rightarrow} & u &] \\
[\,\langle \lambda x.y \mid u{:=}v\rangle & \stackrel{+}{\rightarrow} & v &]
\end{array}$$

Reduction order

Normal (i.e. left-most)

The above is completely in line with computer science tradition. The first sign of deviation is a non-standard definition of "normal form": A term [t] is in *root normal form* if it has one of the forms [N] or [$\lambda x.y$]. As an example, reduction of

$$[\,(\lambda x.x\,'\,x)\,'(\lambda y.\lambda z.y\,'\,y)\,]$$

to root normal form proceeds thus:

$$\begin{array}{lll}
[& (\lambda x.x\,'\,x)\,'(\lambda y.\lambda z.y\,'\,y) & \stackrel{+}{\rightarrow} \\
 & (\lambda y.\lambda z.y\,'\,y)\,'(\lambda y.\lambda z.y\,'\,y) & \stackrel{+}{\rightarrow} \\
 & \lambda z.(\lambda y.\lambda z.y\,'\,y)\,'(\lambda y.\lambda z.y\,'\,y) &]
\end{array}$$

Hence, the root normal form is [$\lambda z.(\lambda y.\lambda z.y\,'\,y)\,'(\lambda y.\lambda z.y\,'\,y)$] which is neither a head normal form, nor a normal form in the usual λ-sense. It is straightforward

root normal form

to implement a computer program that reduces given terms of $\lambda\mathsf{N}$ to root normal form. Such a program will, of course, loop indefinitely if it receives

$$[\,(\lambda x.x\,'\,x)\,'(\lambda x.x\,'\,x)\,]$$

as input.

The second sign of deviation from computer science tradition is the introduction of a non-computable and undecidable relation $\left[\,\boxed{s \approx t}\,\right]$ on terms. $[\,s \approx t\,]$ is true in the following three cases:

(1) $[\,s\,]$ and $[\,t\,]$ both reduce to $[\,\mathsf{N}\,]$

(2) $[\,s\,]$ and $[\,t\,]$ both reduce to terms of form $[\,\lambda x.y\,]$

(3) neither $[\,s\,]$ nor $[\,t\,]$ reduce to root normal form.

$[\,s \approx t\,]$ is false in all other cases. As examples, $[\,\lambda x.x \approx \lambda x.\mathsf{N}\,]$ is true and $[\,\lambda x.x \approx (\lambda x.x\,'\,x)\,'(\lambda x.x\,'\,x)\,]$ is false.

The final break with computer science tradition is: Two closed terms $[\,s\,]$ and $[\,t\,]$ are considered *equal*, written $\left[\,\boxed{s \equiv t}\,\right]$, if

$$[\,r\,'\,s \approx r\,'\,t\,]$$

for all closed terms $[\,r\,]$ of $\lambda\mathsf{N}$. This notion of equality is a *mathematical* or *intensional* or *ontological* notion of equality as opposed to the convertibility relations normally used with λ-calculus. Computer scientists who seek for a notion of convertibility that corresponds to $[\,s \equiv t\,]$ will search in vain and will be on the wrong track.

The decision to consider $[\,s\,]$ and $[\,t\,]$ equal if $[\,s \equiv t\,]$ has far reaching linguistic consequences because it turns on referential transparency. As an example, $[\,\lambda x.x\,]$ and $[\,\lambda y.y\,]$ are not just regarded as convertible terms. Rather, $[\,\lambda x.x\,]$ and $[\,\lambda y.y\,]$ are considered to be the <u>same</u>.

Likewise, if $[\,q\,]$ implements quick-sort and $[\,s\,]$ implements shell-sort then $[\,q\,]$ and $[\,s\,]$ are equal. In other words, quick-sort and shell-sort are the same as far as $\lambda\mathsf{N}$ is concerned.

1.5 Axiomatisation of $\lambda\mathsf{N}$

Now that we have broken with λ-calculus tradition by turning on referential transparency, we haste to turn referential transparency off again, but we do so the way it is done in axiomatic logic. In axiomatic logic, axioms, theorems and proofs consider the shape rather than the meaning of terms. To turn $\lambda\mathsf{N}$ into an axiomatic theory we introduce the syntax class $[\,\mathcal{F}\,]$ of well-formed formulas:

$$[\,\mathcal{F} ::= \mathcal{T} \equiv \mathcal{T}\,].$$

equal
mathematical
intensional
ontological

Hence, the only thing $\lambda\mathsf{N}$ can prove is that one term is equal to another.

The following rules (axioms and inference rules) of $\lambda\mathsf{N}$ are rather obvious:

[$\lambda\mathsf{N}$ rule
R1: $a \equiv b \vdash a \equiv c \vdash b \equiv c$;
R2: $b \equiv c \vdash a\,\text{'}\,b \equiv a\,\text{'}\,c$;
R3: $a \equiv b \vdash \lambda x.a \equiv \lambda x.b$;
R4: $\mathsf{N}\,\text{'}\,a \equiv \mathsf{N}$;
R5: $c \simeq \langle a \mid x{:=}b\rangle \Vdash (\lambda x.a)\,\text{'}\,b \equiv c$;
R6: $\text{if}(\mathsf{N}, b, c) \equiv b$;
R7: $\text{if}(\lambda x.a, b, c) \equiv c$]

Above, [R5] says that [$(\lambda x.a)\,\text{'}\,b \equiv c$] if [c] is identical except for naming of bound variables to the result of replacing [x] by [b] in [a]. R1 says that if [$a \equiv b$] and [$a \equiv c$] are provable then [$b \equiv c$] is provable. The above notation is explained in more detail later.

As an example of a lemma and a proof, consider the following:

[$\lambda\mathsf{N}$ lemma L1.5.1: $\lambda u.u\,\text{'}(\mathsf{N}\,\text{'}\,v) \equiv \lambda u.u\,\text{'}\,\mathsf{N}$]

[The $\lambda\mathsf{N}$ proof of L1.5.1 reads
L1: R4 $\gg$ $\mathsf{N}\,\text{'}\,v \equiv \mathsf{N}$;
L2: R2 $\rhd$ L1 $\gg$ $u\,\text{'}(\mathsf{N}\,\text{'}\,v) \equiv u\,\text{'}\,\mathsf{N}$;
L3: R3 $\rhd$ L2 $\gg$ $\lambda u.u\,\text{'}(\mathsf{N}\,\text{'}\,v) \equiv \lambda u.u\,\text{'}\,\mathsf{N}$]

Line [L1] says that [$\mathsf{N}\,\text{'}\,v \equiv \mathsf{N}$] holds according to axiom [R4]. Line [L2] says that [$u\,\text{'}(\mathsf{N}\,\text{'}\,v) \equiv u\,\text{'}\,\mathsf{N}$] holds because of rule [R2] applied to Line [L1] and so on.

Now let [$\perp$] denote some term that has no root normal form:

$$[\ \perp \doteq (\lambda x.x\,\text{'}\,x)\,\text{'}(\lambda x.x\,\text{'}\,x)\ ].$$

[$\perp$] allows to state two more rules:

[$\lambda\mathsf{N}$ rule
R8: $\perp\text{'}\,a \equiv \perp$;
R9: $\text{if}(\perp, b, c) \equiv \perp$]

In addition to these rules, Chapter 6 and Chapter 7 introduce four more rules: Quartum Non Datur (QND) says that every term [t] satisfies [$t \equiv \mathsf{N}$] or [$t \equiv \lambda x.t\,\text{'}\,x$] or [$t \equiv \perp$]. Extensionality says that if [$s\,\text{'}\,x_1\,\text{'}\cdots\text{'}\,x_n \approx t\,\text{'}\,x_1\,\text{'}\cdots\text{'}\,x_n$] for all [$n \geq 0$] and all [$x_1, \ldots, x_n$] then [$s \equiv t$]. Minimality says that [$\mathsf{Y}\,\text{'}\,f$] is the minimal fixed point of [f] w.r.t. a certain relation [$x \preceq y$]. Monotonicity says that [$a \preceq b \vdash f\,\text{'}\,a \preceq f\,\text{'}\,b$]. As far as axiomatic $\lambda\mathsf{N}$ is concerned, the relations [$x \approx y$] and [$x \preceq y$] and the fixed point operator [Y] are defined concepts that are defined on basis of the fundamental constructs of $\lambda\mathsf{N}$. The definitions are stated later.

1.6 Introduction to MT

$\lambda\mathsf{N}$ is easy to introduce because it is based on a simple reduction system. Furthermore, all theorems of $\lambda\mathsf{N}$ say something about the behavior of computing machines which makes the theorems rather concrete of nature.

MT is more complicated. An introduction of MT can be based on intuition or models or axioms. This section describes a model [$\mathcal{M}$] of MT in an intuitive manner. The formal construction of [$\mathcal{M}$] is outside the scope of the present paper. We shall refer to [$\mathcal{M}$] simply as "the model".

MT is an extension of $\lambda\mathsf{N}$. Compared to $\lambda\mathsf{N}$, MT has two more constructs, namely [$\mathsf{E}x$] and [εx]. Hence, the syntax reads:

$$\begin{array}{lllll}
\text{variables } \mathcal{V}\text{:} & [\,\mathcal{V} & ::= & x \mid y \mid z \mid \cdots &] \\
\text{terms } \mathcal{T}\text{:} & [\,\mathcal{T} & ::= & \mathcal{V} \mid \mathsf{N} \mid \lambda\mathcal{V}.\mathcal{T} \mid \mathcal{T}\text{'}\mathcal{T} \mid \text{if}(\mathcal{T},\mathcal{T},\mathcal{T}) \mid \mathsf{E}\mathcal{T} \mid \varepsilon\mathcal{T} &] \\
\text{formulas } \mathcal{F}\text{:} & [\,\mathcal{F} & ::= & \mathcal{T} \equiv \mathcal{T} &]
\end{array}$$

In $\lambda\mathsf{N}$, [$s \equiv t$] if [$r\text{'}s \approx r\text{'}t$] for all <u>terms</u> [r] of $\lambda\mathsf{N}$. In contrast, [$s \equiv t$] holds in MT if [$r\text{'}s \approx r\text{'}t$] for all [r] in the <u>model</u> of MT. The point is that there is no simple definition of [$s \equiv t$] in MT.

The construct [$\mathsf{E}x$] is a simple and rather weak existential quantifier:

$$\begin{array}{llllll}
[& \mathsf{E}x & \equiv & \mathsf{N} &] & \text{if } \exists y{\in}\mathcal{M}\colon x\text{'}y \equiv \mathsf{N} \\
[& \mathsf{E}x & \equiv & \perp &] & \text{otherwise}
\end{array}$$

MT has four axioms about [$\mathsf{E}x$] (c.f. Section 8.4).

[εx] is the construct from which MT draws its power. [εx] is related to a subset [L] of [$\mathcal{M}$]. The structure of [L] resembles the structure of the universe of ZFC: the elements of [L] are "well-founded" in a sense that resembles well-foundedness in ZFC as expressed by the axiom of restriction. Furthermore, the elements of [L] are of "limited size" like the sets of ZFC are. However, limitation of size is implicit in ZFC whereas it is explicit in MT (c.f. the discussion of "inner hulls" in Section 8.6).

Elements of [L] will be referred to as *classical* maps. The properties of [εx] are:

- If [$x\text{'}y \equiv \perp$] for some classical [y] then [$\varepsilon x \equiv \perp$].
- If [$x\text{'}y \not\equiv \perp$] for all classical [y] then [$\varepsilon x \in L$].
- If [$x\text{'}y \not\equiv \perp$] for all classical [y] and [$x\text{'}y \equiv N$] for some classical [y] then [$x\text{'}\varepsilon x \equiv N$].

The axioms about [εx] are stated in Section 8.9 and again in Section 9.5. The axioms are somewhat difficult to jump to because they rely on several defined concepts (the universal quantifier [$\forall x\colon a$], which is defined on basis of [εx], the implication relation [$x \rightarrow y$], which is defined on basis of [$\text{if}(x,y,z)$], and so one).

classical map

One of the defined concepts used in the axioms is the map [ℓ] which has the following properties:

[ℓ'x $\equiv$ N] if x is classical
[ℓ'x $\equiv$ $\perp$] otherwise

The definition of [ℓ] is complex and intricate (c.f. Section 8.6 and Section 8.10), but that seems unavoidable since [ℓ] gives a more precise description of the universe of ZFC than all the axioms of ZFC together (c.f. Chapter 11 where ZFC is modeled inside MT).

Chapter 2

Presentation of $\lambda\mathsf{N}$ in ZFC

2.1 Introduction

Chapter 2 presents $\lambda\mathsf{N}$ inside the framework of set theory. Chapter 3 presents $\lambda\mathsf{N}$ as a programming language. Chapter 4 presents $\lambda\mathsf{N}$ as a theory in its own right. Chapter 8 and on generalize $\lambda\mathsf{N}$ into full MT. The reader must have patience until Chapter 8 concerning quantifiers and other issues where the power of MT stand out. It is possible, however, to read Chapter 8 after Chapter 3.

2.2 Overloading

Conjunction in ZFC is denoted $[\, x \wedge y \,]$. Conjunction in $\lambda\mathsf{N}$ is also denoted $[\, x \wedge y \,]$. Here are some examples:

If $[\, x \,]$ and $[\, y \,]$	are formulas of	ZFC
then $[\, x \wedge y \,]$	is a formula of	ZFC.
If $[\, x \,]$ and $[\, y \,]$	are terms of	$\lambda\mathsf{N}$
then $[\, x \wedge y \,]$	is a term of	$\lambda\mathsf{N}$.
If $[\, x \,]$ and $[\, y \,]$	belong to	the term model of $\lambda\mathsf{N}$
then $[\, x \wedge y \,]$	belongs to	the term model of $\lambda\mathsf{N}$.

Hence, $[\, x \wedge y \,]$ has at least three different meanings in this paper, depending on context. In Chapter 2, however, matters are much simpler:

$[\, x \wedge y \,]$	is conjunction	in ZFC
$[\, x \vee y \,]$	is disjunction	in ZFC
$[\, \neg x \,]$	is negation	in ZFC
$[\, x \Rightarrow y \,]$	is implication	in ZFC
$[\, x \Leftrightarrow y \,]$	is biimplication	in ZFC
$[\, \forall x \colon \mathcal{A} \,]$	is universal quantification	in ZFC
$[\, \exists x \colon \mathcal{A} \,]$	is existential quantification	in ZFC
$[\, x \in y \,]$	is membership	in ZFC
$[\, x = y \,]$	is equality	in ZFC

In general, all terms and formulas of this chapter are terms and formulas of ZFC. All lemmas in this chapter are lemmas of ZFC.

2.3 Rigor

None of the lemmas in this chapter are proved. The lemmas are merely stated for information. A formal consistency proof for $\lambda\mathsf{N}$ and MT are outside the scope of the present paper.

2.4 The syntax of $\lambda\mathsf{N}$

$\lambda\mathsf{N}$ has a countable collection [$x_0, x_1, x_2, \ldots$] of variables. Let [$\mathcal{V}$] denote the syntax class of variables:

$$[\, \mathcal{V} ::= x_0 \mid x_1 \mid x_2 \mid \cdots \,].$$

let [$\mathcal{T}$] denote the syntax class of terms of $\lambda\mathsf{N}$. The definition of [$\mathcal{T}$] reads:

$$[\, \mathcal{T} ::= \mathcal{V} \mid \mathsf{N} \mid \lambda\mathcal{V}.\mathcal{T} \mid \mathcal{T}\,\text{'}\,\mathcal{T} \mid \text{if}(\mathcal{T},\mathcal{T},\mathcal{T}) \,].$$

The definition is stated in Bachus Naur Form (*BNF*) [1].

We represent terms of $\lambda\mathsf{N}$ by tuples in ZFC as follows:

$$\begin{array}{llll}
[\, x_i & \doteq & \langle 0, i\rangle &] \\
[\, \mathsf{N} & \doteq & \langle 1\rangle &] \\
[\, \lambda x.y & \doteq & \langle 2, x, y\rangle &] \\
[\, x\,\text{'}\,y & \doteq & \langle 3, x, y\rangle &] \\
[\, \text{if}(x,y,z) & \doteq & \langle 4, x, y, z\rangle &]
\end{array}$$

The definition of the set [$\mathcal{V}$] of variables of $\lambda\mathsf{N}$ reads

$$[\, \mathcal{V} \doteq \{x_i \mid i \in \omega\} \,]$$

where $\boxed{\omega}$ is the set of finite von Neumann ordinals [9] of ZFC. In other words, [ω] is the set of natural numbers.

The set [$\mathcal{T}$] of terms is the smallest set that satisfies

$$\begin{array}{llll}
[\, i \in \omega & \Rightarrow & x_i \in \mathcal{T} &] \\
[\, 0 = 0 & \Rightarrow & \mathsf{N} \in \mathcal{T} &] \\
[\, x \in \mathcal{V} \wedge y \in \mathcal{T} & \Rightarrow & \lambda x.y \in \mathcal{T} &] \\
[\, x \in \mathcal{T} \wedge y \in \mathcal{T} & \Rightarrow & x\,\text{'}\,y \in \mathcal{T} &] \\
[\, x \in \mathcal{T} \wedge y \in \mathcal{T} \wedge z \in \mathcal{T} & \Rightarrow & \text{if}(x,y,z) \in \mathcal{T} &]
\end{array}$$

Let [$x\,\text{'}\,y$] be left associative such that [$x\,\text{'}\,y\,\text{'}\,z$] means [$(x\,\text{'}\,y)\,\text{'}\,z$]. Let [$x\,\text{'}\,y$] have higher priority than [$\lambda x.y$] such that [$\lambda x.y\,\text{'}\,z$] means [$\lambda x.(y\,\text{'}\,z)$]. For a complete list of priority and associativity rules see Appendix C.

BNF

2.5 The intuitive meaning of terms

As mentioned in Section 1.3, $\boxed{\mathsf{N}}$ denotes the ur-element.

For all [$x \in \mathcal{V}$] and [$\mathcal{A} \in \mathcal{T}$], $\boxed{\lambda x.\mathcal{A}}$ denotes the function that maps [x] to [$\mathcal{A}$]. As an example,

[$\lambda x.\mathsf{N}$]

denotes the function that maps anything to [N]. As another example,

[$\lambda x.x$]

denotes the identity function.

If [$\mathcal{A}$] denotes a function then $\boxed{\mathcal{A}\,\text{'}\,\mathcal{B}}$ denotes that function applied to [$\mathcal{B}$]. As examples,

[$(\lambda x.x)\,\text{'}\,\mathsf{N}$]	and	[N]	denote the same map and
[$(\lambda x.x)\,\text{'}(\lambda x.\mathsf{N})$]	and	[$\lambda x.\mathsf{N}$]	denote the same map.

Finally, $\boxed{\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C})}$ denotes the same map as [$\mathcal{B}$] if [$\mathcal{A}$] denotes the ur-element; [$\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C})$] denotes the same map as [$\mathcal{C}$] if [$\mathcal{A}$] denotes a function.

2.6 Some predicates and functions on terms

Let $\boxed{\mathrm{free}(x,\mathcal{A})}$ be a predicate of ZFC with the following property: For all [$x \in \mathcal{V}$] and [$\mathcal{A} \in \mathcal{T}$], [$\mathrm{free}(x,\mathcal{A})$] is true if and only if [x] occurs free in [$\mathcal{A}$]. For all [$i,j \in \omega$] and all [$\mathcal{A},\mathcal{B},\mathcal{C} \in \mathcal{T}$], the predicate satisfies:

$$\begin{array}{llll}
[\ \mathrm{free}(x_i,x_j) & \Leftrightarrow & i=j &] \\
[\ \mathrm{free}(x_i,\mathsf{N}) & \Leftrightarrow & 0=1 &] \\
[\ \mathrm{free}(x_i,\lambda x_j.\mathcal{A}) & \Leftrightarrow & i\neq j \wedge \mathrm{free}(x_i,\mathcal{A}) &] \\
[\ \mathrm{free}(x_i,\mathcal{A}\,\text{'}\,\mathcal{B}) & \Leftrightarrow & \mathrm{free}(x_i,\mathcal{A}) \vee \mathrm{free}(x_i,\mathcal{B}) &] \\
[\ \mathrm{free}(x_i,\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C})) & \Leftrightarrow & \mathrm{free}(x_i,\mathcal{A}) \vee \mathrm{free}(x_i,\mathcal{B}) \vee \mathrm{free}(x_i,\mathcal{C}) &]
\end{array}$$

A term of λN is *closed* if it has no free variables. Let $\boxed{\bar{\mathcal{T}}}$ denote the set of closed terms.

For all [$\mathcal{A},\mathcal{B} \in \mathcal{T}$] let $\boxed{\mathcal{A} \simeq \mathcal{B}}$ denote that [$\mathcal{A}$] and [$\mathcal{B}$] are identical except for renaming of bound variables.

For all [$x \in \mathcal{V}$] and all [$\mathcal{A},\mathcal{B} \in \mathcal{T}$] let

$$\left[\ \boxed{\langle \mathcal{A} \mid x{:=}\mathcal{B}\rangle}\ \right]$$

denote the result of replacing all free occurrences of [x] in [$\mathcal{A}$] by [$\mathcal{B}$] with renaming of bound variables in case of variable clashes. The precise procedure for

closed term

renaming of bound variables is unimportant, but it will be assumed throughout this paper that $[\ \langle \mathcal{A} \mid x{:=}\mathcal{B} \rangle\ ]$ renames variables in some, definite way.

As examples of use,

$$[\ \langle \lambda x_0 . x_0 \mathbin{'} x_1 \mid x_1{:=}x_0 \rangle \simeq \lambda x_2 . x_2 \mathbin{'} x_0\ ], \text{ and}$$

$$[\ \langle \lambda x_0 . x_0 \mathbin{'} x_1 \mid x_1{:=}x_0 \rangle \simeq \lambda x_7 . x_7 \mathbin{'} x_0\ ]$$

are both true in ZFC.

2.7 The semantics of $\lambda\mathsf{N}$

The semantics of $\lambda\mathsf{N}$ is based on a procedure which, given a closed term, decides whether or not the term denotes the ur-element. The procedure is defined in Section 2.8. The properties and use of the procedure are stated in the following.

When the procedure is applied to a closed term $[\ \mathcal{A}\ ]$, then the procedure either gives a result in finite time or the procedure never gives a result. If the procedure never gives a result then $[\ \mathcal{A}\ ]$ will be said to be *undecidable*. If the procedure gives a result in finite time, then it either gives the result that $[\ \mathcal{A}\ ]$ denotes the ur-element or that $[\ \mathcal{A}\ ]$ denotes some function. In the former case, $[\ \mathcal{A}\ ]$ is said to be an *ur-term* and in the latter it is said to be a *function term*.

The procedure will be referred to as the trisection procedure because it divides the terms of $\lambda\mathsf{N}$ into three classes, namely the classes of ur-terms, function terms, and undecidable terms, respectively.

Two closed terms $[\ \mathcal{A}\ ]$ and $[\ \mathcal{B}\ ]$ are said to be *root equivalent*, written $\boxed{\mathcal{A} \approx \mathcal{B}}$, if they belong to the same class. As an example, $[\ \lambda x_0 . x_0 \approx \lambda x_0 . \mathsf{N}\ ]$ is true since $[\ \lambda x_0 . x_0\ ]$ and $[\ \lambda x_0 . \mathsf{N}\ ]$ both happen to be function terms.

The semantics of $\lambda\mathsf{N}$ is defined by the relation $\boxed{\mathcal{A} \equiv \mathcal{B}}$ on terms. If $[\ \mathcal{A}\ ]$ and $[\ \mathcal{B}\ ]$ are closed terms, then

$$[\ \mathcal{A} \equiv \mathcal{B} \Leftrightarrow \forall \mathcal{C} \in \bar{\mathcal{T}} \colon \mathcal{C} \mathbin{'} \mathcal{A} \approx \mathcal{C} \mathbin{'} \mathcal{B}\ ].$$

This is equivalent to one of the two common definitions of $[\ x = y\ ]$ in ZFC, namely

$$[\ x = y \Leftrightarrow \forall z \colon (x \in z \Leftrightarrow y \in z)\ ].$$

There is also a definition of $[\ \mathcal{A} \equiv \mathcal{B}\ ]$ which is equivalent to $[\ x = y \Leftrightarrow \forall z \colon (z \in x \Leftrightarrow z \in y)\ ]$. That definition is stated in Section 2.11.

If $[\ \mathcal{A}\ ]$ or $[\ \mathcal{B}\ ]$ or both are not closed, then

$$[\ \mathcal{A} \equiv \mathcal{B} \Leftrightarrow \lambda v_1 \ldots \lambda v_n . \mathcal{A} \equiv \lambda v_1 \ldots \lambda v_n . \mathcal{B}\ ]$$

undecidable
ur-term
function term
root equivalent

where [$v_1, \ldots, v_n$] are the free variables of [$\mathcal{A}$] and [$\mathcal{B}$].

It is undecidable in general whether or not a term is undecidable. Therefore the relations [$\mathcal{A} \approx \mathcal{B}$] and [$\mathcal{A} \equiv \mathcal{B}$] are undecidable, and $\lambda\mathsf{N}$ is an undecidable theory.

2.8 The trisection procedure of λN

In short, the *trisection procedure* of $\lambda\mathsf{N}$ is: Given a term, apply the reduction rules

$$\begin{array}{lll}[\,\mathsf{N}\,\text{'}\,\mathcal{A} & \overset{+}{\to} & \mathsf{N} \quad]\\ [\,(\lambda x.\mathcal{A})\,\text{'}\,\mathcal{B} & \overset{+}{\to} & \langle \mathcal{A} \mid x{:=}\mathcal{B}\rangle \quad]\\ [\,\mathrm{if}(\mathsf{N},\mathcal{B},\mathcal{C}) & \overset{+}{\to} & \mathcal{B} \quad]\\ [\,\mathrm{if}(\lambda x.\mathcal{A},\mathcal{B},\mathcal{C}) & \overset{+}{\to} & \mathcal{C} \quad]\end{array}$$

to the term until the term has one of the forms [N] or [$\lambda x.\mathcal{A}$].

If the term ends having form [N], then the term is an ur-term. If the term ends having form [$\lambda x.\mathcal{A}$] then the term is a function term. If the reduction never ends, then the term is an undecidable term. If a term can be reduced in more than one way, use "normal order" reduction [2].

A more elaborate definition of the trisection procedure is given in the following. The definition uses an auxiliary function [$\mathcal{R}(\mathcal{A})$] from terms to terms which performs zero or one reduction of [$\mathcal{A}$]. Let [$\mathcal{R}$] be the function from terms to terms given by:

$$\begin{array}{llll}[\,\mathcal{R}(x_i) & = & x_i &]\\ [\,\mathcal{R}(\mathsf{N}) & = & \mathsf{N} &]\\ [\,\mathcal{R}(\lambda x.\mathcal{A}) & = & \lambda x.\mathcal{A} &]\\ \\ [\,\mathcal{R}(\mathsf{N}\,\text{'}\,\mathcal{A}) & = & \mathsf{N} &]\\ [\,\mathcal{R}((\lambda x.\mathcal{A})\,\text{'}\,\mathcal{B}) & = & \langle \mathcal{A} \mid x{:=}\mathcal{B}\rangle &]\\ [\,\mathcal{R}(\mathrm{if}(\mathsf{N},\mathcal{B},\mathcal{C})) & = & \mathcal{B} &]\\ [\,\mathcal{R}(\mathrm{if}(\lambda x.\mathcal{A},\mathcal{B},\mathcal{C})) & = & \mathcal{C} &]\\ \\ [\,\mathcal{R}((\mathcal{A}\,\text{'}\,\mathcal{B})\,\text{'}\,\mathcal{C}) & = & \mathcal{R}(\mathcal{A}\,\text{'}\,\mathcal{B})\,\text{'}\,\mathcal{C} &]\\ [\,\mathcal{R}(\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C})\,\text{'}\,\mathcal{D}) & = & \mathcal{R}(\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C}))\,\text{'}\,\mathcal{D} &]\\ [\,\mathcal{R}(\mathrm{if}(\mathcal{A}\,\text{'}\,\mathcal{B},\mathcal{D},\mathcal{E})) & = & \mathrm{if}(\mathcal{R}(\mathcal{A}\,\text{'}\,\mathcal{B}),\mathcal{D},\mathcal{E}) &]\\ [\,\mathcal{R}(\mathrm{if}(\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D},\mathcal{E})) & = & \mathrm{if}(\mathcal{R}(\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C})),\mathcal{D},\mathcal{E}) &]\end{array}$$

If [$\mathcal{A}$] is closed then [$\mathcal{R}(\mathcal{A})$] is also closed. Let [$\mathcal{R}^n(\mathcal{A})$] denote

$$\left[\,\underbrace{\mathcal{R}(\mathcal{R}(\cdots\mathcal{R}(}_{n}\mathcal{A})\cdots))\,\right].$$

trisection procedure

A term is said to be in *root normal form* if it has one of the forms [N] or [$\lambda x.\mathcal{A}$]. The trisection procedure now reads: Given a term [$\mathcal{A}$], compute [$\mathcal{R}^n(\mathcal{A})$] for increasing [n] until [$\mathcal{R}^n(\mathcal{A})$] is in root normal form. If [$\mathcal{R}^n(\mathcal{A})$] is [N] then [$\mathcal{A}$] is an ur-term. if [$\mathcal{R}^n(\mathcal{A})$] has form [$\lambda x.\mathcal{A}$] then [$\mathcal{A}$] is a function term. If [$\mathcal{R}^n(\mathcal{A})$] never becomes a root normal form then [$\mathcal{A}$] is an undecidable term.

As an example,

$$[\ \mathrm{if}(\mathrm{if}(\mathsf{N}, \lambda x_0.x_0, \mathsf{N}), \mathsf{N}, \lambda x_1.x_1)\ ]$$

is a function term:

$$\begin{array}{ll} [& \mathrm{if}(\mathrm{if}(\mathsf{N}, \lambda x_0.x_0, \mathsf{N}), \mathsf{N}, \lambda x_1.x_1) \overset{+}{\to} \\ & \mathrm{if}(\lambda x_0.x_0, \mathsf{N}, \lambda x_1.x_1) \overset{+}{\to} \\ & \lambda x_1.x_1\ ] \end{array}$$

As another example, define

$$\left[\boxed{\perp} \doteq (\lambda x_0.x_0\,{}'\,x_0)\,{}'(\lambda x_0.x_0\,{}'\,x_0)\right].$$

[$\perp$] is an undecidable term.

2.9 Properties of [$x \equiv y$]

The relation [$x \equiv y$] has the following properties.

ZFC lemma 2.9.1 For all [$x \in \mathcal{V}$] and all [$\mathcal{A}, \mathcal{B}, \mathcal{C}, \dot{\mathcal{A}}, \dot{\mathcal{B}}, \dot{\mathcal{C}} \in \mathcal{T}$]:

$$\begin{array}{llll} [\ 0 = 0 & \Rightarrow & \mathcal{A} \equiv \mathcal{A} &] \\ [\ \mathcal{A} \equiv \mathcal{B} & \Rightarrow & \mathcal{B} \equiv \mathcal{A} &] \\ [\ \mathcal{A} \equiv \mathcal{B} \wedge \mathcal{B} \equiv \mathcal{C} & \Rightarrow & \mathcal{A} \equiv \mathcal{C} &] \\ [\ \mathcal{A} \equiv \dot{\mathcal{A}} & \Rightarrow & \lambda x.\mathcal{A} \equiv \lambda x.\dot{\mathcal{A}} &] \\ [\ \mathcal{A} \equiv \dot{\mathcal{A}} \wedge \mathcal{B} \equiv \dot{\mathcal{B}} & \Rightarrow & \mathcal{A}\,{}'\,\mathcal{B} \equiv \dot{\mathcal{A}}\,{}'\,\dot{\mathcal{B}} &] \\ [\ \mathcal{A} \equiv \dot{\mathcal{A}} \wedge \mathcal{B} \equiv \dot{\mathcal{B}} \wedge \mathcal{C} \equiv \dot{\mathcal{C}} & \Rightarrow & \mathrm{if}(\mathcal{A}, \mathcal{B}, \mathcal{C}) \equiv \mathrm{if}(\dot{\mathcal{A}}, \dot{\mathcal{B}}, \dot{\mathcal{C}}) &] \end{array}$$

ZFC lemma 2.9.2 For all [$x \in \mathcal{V}$] and all [$\mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathcal{T}$]:

$$\begin{array}{llll} [\ \mathsf{N}\,{}'\,\mathcal{B} & \equiv & \mathsf{N} &] \\ [\ (\lambda x.\mathcal{A})\,{}'\,\mathcal{B} & \equiv & \langle \mathcal{A} \mid x{:=}\mathcal{B} \rangle &] \\ [\ \perp{}'\,\mathcal{B} & \equiv & \perp &] \\ [\ \mathrm{if}(\mathsf{N}, \mathcal{B}, \mathcal{C}) & \equiv & \mathcal{B} &] \\ [\ \mathrm{if}(\lambda x.\mathcal{A}, \mathcal{B}, \mathcal{C}) & \equiv & \mathcal{C} &] \\ [\ \mathrm{if}(\perp, \mathcal{B}, \mathcal{C}) & \equiv & \perp &] \end{array}$$

ZFC lemma 2.9.3 If [$\mathcal{A}, \dot{\mathcal{A}} \in \mathcal{T}$] and [$\mathcal{A} \simeq \dot{\mathcal{A}}$] then [$\mathcal{A} \equiv \dot{\mathcal{A}}$].

As mentioned in Section 2.1, this paper does not prove lemmas stated in Chapter 2. To prove the lemmas above, prove that the reduction system in Section 2.8 has the Church-Rosser property [2].

root normal form

2.10 The universe of λN

For all closed terms [$\mathcal{A}$] define $\boxed{c(\mathcal{A})}$ by

$$[\ c(\mathcal{A}) \doteq \{\mathcal{B} \in \bar{\mathcal{T}} \mid \mathcal{B} \equiv \mathcal{A}\}\ ].$$

Define [$\mathcal{M}$] as the set of equivalence classes of [$\bar{\mathcal{T}}$] under [$\equiv$]:

$$[\ \mathcal{M} \doteq \{c(\mathcal{A}) \mid \mathcal{A} \in \bar{\mathcal{T}}\}\ ].$$

One may think of [$\mathcal{M}$] as the universe of λN. If [$\mathcal{A} \in \bar{\mathcal{T}}$] then [$c(\mathcal{A})$] is the map denoted by [$\mathcal{A}$].

[$\mathcal{M}$] is the underlying set of the term model of λN. This paper does not present the term model formally.

2.11 Extensionality

For all sets [S] let $\boxed{S^{<\omega}}$ denote the set of tuples [$\langle s_1, s_2, \ldots, s_n\rangle$] of elements of [S]. As a special case, the empty tuple $\boxed{\langle\,\rangle}$ belongs to [$S^{<\omega}$] for any set [S].

For all [$\mathcal{A} \in \mathcal{T}$] and all [$\mathcal{B} \in \mathcal{T}^{<\omega}$], define $\boxed{\mathcal{A}\text{''}\,\mathcal{B}}$ such that

$$[\ \mathcal{A}\text{''}\langle \mathcal{B}_1, \ldots, \mathcal{B}_n\rangle = \mathcal{A}\text{'}\,\mathcal{B}_1\text{'}\cdots\text{'}\,\mathcal{B}_n\ ]$$

for all [$n \in \omega$] and all [$\mathcal{A}, \mathcal{B}_1, \ldots, \mathcal{B}_n \in \mathcal{T}$]. As a special case,

$$[\ \mathcal{A}\text{''}\langle\,\rangle = \mathcal{A}\ ].$$

The trisection procedure of λN has the following property:

ZFC lemma 2.11.1 For all [$\mathcal{A}, \mathcal{B} \in \bar{\mathcal{T}}$],

$$[\ \forall \mathcal{C}{\in}\bar{\mathcal{T}}{:}\, \mathcal{C}\text{'}\,\mathcal{A} \approx \mathcal{C}\text{'}\,\mathcal{B} \Leftrightarrow \forall \mathcal{C}{\in}\bar{\mathcal{T}}^{<\omega}{:}\, \mathcal{A}\text{''}\,\mathcal{C} \approx \mathcal{B}\text{''}\,\mathcal{C}\ ].$$

This property is analogous to extensionality of ZFC which reads

$$[\ \forall z{:}\, (x \in z \Leftrightarrow y \in z) \Leftrightarrow \forall z{:}\, (z \in x \Leftrightarrow z \in y)\ ].$$

The definition of [$\mathcal{A} \equiv \mathcal{B}$] reads

$$[\ \mathcal{A} \equiv \mathcal{B} \Leftrightarrow \forall \mathcal{C}{\in}\bar{\mathcal{T}}{:}\, \mathcal{C}\text{'}\,\mathcal{A} \approx \mathcal{C}\text{'}\,\mathcal{B}\ ].$$

hence, the lemma may be stated

ZFC lemma 2.11.2 (Extensionality) For all [$\mathcal{A}, \mathcal{B} \in \bar{\mathcal{T}}$],

$$[\ \mathcal{A} \equiv \mathcal{B} \Leftrightarrow \forall \mathcal{C}{\in}\bar{\mathcal{T}}^{<\omega}{:}\, \mathcal{A}\text{''}\,\mathcal{C} \approx \mathcal{B}\text{''}\,\mathcal{C}\ ].$$

2.12 Graphical representation of maps

To develop an intuitive understanding of what a map is, we introduce a graphical representation of maps. The graphical representation is based on the following facts:

- Any closed term is either an ur-term, an undecidable term, or a function term.
- [$\mathcal{A} \equiv \mathsf{N}$] if and only if [$\mathcal{A}$] is an ur-term.
- [$\mathcal{A} \equiv \perp$] if and only if [$\mathcal{A}$] is an undecidable term.
- [$\mathcal{A} \equiv \mathcal{B} \Leftrightarrow \forall \mathcal{C} \in \bar{\mathcal{T}}^{<\omega} \colon \mathcal{A} \,\text{''}\, \mathcal{C} \approx \mathcal{B} \,\text{''}\, \mathcal{C}$] (Extensionality).

Maps will be represented by trees whose nodes are labelled [N], [$\perp$], or [λ] and whose edges are labelled by closed terms. The map denoted by a closed term [$\mathcal{A}$] is constructed thus: If [$\mathcal{A}$] is an ur-term, then the map denoted by [$\mathcal{A}$] is represented by a tree with a single node and no edges. The node is labelled [N]:

$$\boxed{\mathsf{N}}$$

Similarly, if [$\mathcal{A}$] is an undecidable term, then the map denoted by [$\mathcal{A}$] is represented thus:

$$\boxed{\perp}$$

If [$\mathcal{A}$] is a function term then the graphical representation of the map denoted by [$\mathcal{A}$] is constructed recursively as follows: Draw a node labelled [λ]. This node will be referred to as the *root node* of the graphical representation. For each closed term [$\mathcal{B}$], draw the graphical representation of the term [$\mathcal{A} \,\text{'}\, \mathcal{B}$] and connect it to the root node by an edge labelled by the term [$\mathcal{B}$]. As an example, the graphical representation of [$\lambda x.\mathsf{N}$] may be constructed thus. First, the root node is drawn:

$$\boxed{\lambda}$$

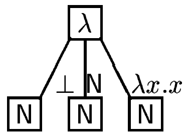
The term [$(\lambda x.\mathsf{N}) \,\text{'}\, \perp$] is an ur-term (i.e. [$(\lambda x.\mathsf{N}) \,\text{'}\, \perp \equiv \mathsf{N}$]), so a node labelled [N] is to be connected to the root node by an edge labelled [$\perp$]:

$$\begin{tikzcd}
\boxed{\lambda} \arrow[d, dash, "\perp"] \\
\boxed{\mathsf{N}}
\end{tikzcd}$$

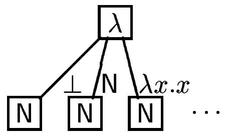
[$(\lambda x.\mathsf{N}) \,\text{'}\, \mathsf{N}$] and [$(\lambda x.\mathsf{N}) \,\text{'}(\lambda x.x)$] are also ur-terms:

---

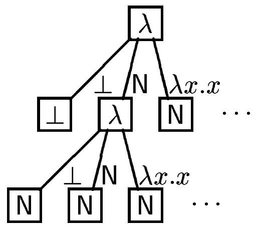
root node



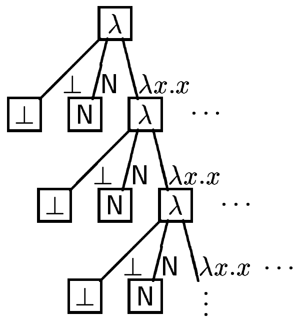
There are infinitely many closed terms, so it is impossible to complete the drawing. Nevertheless, the graphical representation may provide a useful mental picture of [$\lambda x.\mathsf{N}$]:



Here is a drawing of [$\lambda x.\mathrm{if}(x, \lambda x.\mathsf{N}, \mathsf{N})$] (incomplete, of course):



The graphical representation of a function is infinitely "wide" in the sense that infinitely many edges descend from the root. The graphical representation may also be infinitely "deep" like the one for [$\lambda x.x$]:



The graphical representation of maps of λN may be "countably wide" and "countably deep". The more general maps of MT may be "uncountably wide" but are still "countably deep".

2.13 Comparison with pure [λ]-calculus

The literature contains a vast body of results on pure lambda calculus. For that reason it is tempting to try to understand λN as a slight perturbation of pure lambda calculus and to try to carry results from pure lambda calculus to $\lambda\mathsf{N}$. That is not possible, however. Most of the results on pure lambda calculus that have ever been published are meaningless or irrelevant or trivial in $\lambda\mathsf{N}$. Section 2.13 compares pure lambda calculus and $\lambda\mathsf{N}$ in order to give an idea of what does carry over and what does not.

The notion of convertibility is central to the study of pure lambda calculus. As an example, let

$$[\, \mathcal{A} =_\beta \mathcal{B} \,]$$

denote that $[\,\mathcal{A}\,]$ and $[\,\mathcal{B}\,]$ are $[\,\beta\,]$-convertible. The central notion of $\lambda\mathsf{N}$ is the equality relation $[\,\mathcal{A} \equiv \mathcal{B}\,]$. The relation between convertibility and equality is:

ZFC lemma 2.13.1 If $[\,\mathcal{A}, \mathcal{B} \in \mathcal{T}\,]$ then

$$[\, \mathcal{A} =_\beta \mathcal{B} \Rightarrow \mathcal{A} \equiv \mathcal{B} \,].$$

As an example of use, if one defines

$$[\, \mathsf{Y} \doteq \lambda f.(\lambda x.f\,\text{'}(x\,\text{'}\,x))\,\text{'}(\lambda x.f\,\text{'}(x\,\text{'}\,x)) \,]$$

then it is well known from pure lambda calculus that

$$[\, f\,\text{'}(\mathsf{Y}\,\text{'}\,f) =_\beta \mathsf{Y}\,\text{'}\,f \,]$$

and, hence, by Lemma 2.13.1

$$[\, f\,\text{'}(\mathsf{Y}\,\text{'}\,f) \equiv \mathsf{Y}\,\text{'}\,f \,].$$

Hence, the existence of fixed points carries over from pure lambda calculus to $\lambda\mathsf{N}$.

As an example of a result that does not carry over, consider the following: If two terms are $[\,\beta\,]$-convertible, then they $[\,\beta\,]$-reduce to a common term:

$$[\, \mathcal{A} =_\beta \mathcal{B} \Rightarrow \exists \mathcal{C}\colon \mathcal{A} \to_\beta \mathcal{C} \wedge \mathcal{B} \to_\beta \mathcal{C} \,].$$

The closest one can get to this result in $\lambda\mathsf{N}$ is:

$$[\, \mathcal{A} \equiv \mathcal{B} \Rightarrow \exists \mathcal{C}\colon \mathcal{A} \equiv \mathcal{C} \wedge \mathcal{B} \equiv \mathcal{C} \,]$$

which is trivial. A statement something like

$$\left[\, \mathcal{A} \equiv \mathcal{B} \Rightarrow \exists \mathcal{C}\colon \mathcal{A} \overset{+}{\to} \mathcal{C} \wedge \mathcal{B} \overset{+}{\to} \mathcal{C} \,\right] \text{ fails.}$$

As an example of two terms that are equal without being $[\,\beta\,]$-convertible, consider

$$[\, a \doteq \mathsf{Y}\,\text{'}\,\lambda f.\lambda x.\lambda x.f \,], \text{ and}$$

$$[\, b \doteq \mathsf{Y}\,\text{'}\,\lambda f.\lambda x.\lambda x.\lambda x.f \,].$$

The two terms satisfy

[$a \equiv \lambda x.\lambda x.a$], and

[$b \equiv \lambda x.\lambda x.\lambda x.b$].

Hence, the terms satisfy

[$a \equiv \lambda x.\lambda x.\lambda x.\lambda x.\cdots$], and

[$b \equiv \lambda x.\lambda x.\lambda x.\lambda x.\cdots$].

The terms are equal by extensionality and one can prove [$a \equiv b$] in λN, but the terms are not in any way convertible. As another example, a function [f] that does shell-sort and a function [g] that does bubble-sort are equal in λN even though they are not even remotely connected by any sort of convertibility.

To sum up the differences between pure [λ]-calculus and λN one may say that the central concept of pure [λ]-calculus is convertibility whereas the central concept of λN is equality. Convertibility implies equality but equality does not imply convertibility. The ur-element also makes a difference, but the major difference is that between convertibility and equality.

One could make λN look more like pure [λ]-calculus by replacing equality by some sort of convertibility, but then λN seems to loose the power that allows it to generalize to a theory that can compete with ZFC.

One could also go the other way and try to port equality to pure [λ]-calculus. That is indeed possible, but not necessarily very useful. Inspection of the trisection procedure will show that if [$\mathcal{A}$] is a pure lambda term then [$\mathcal{R}(\mathcal{A})$] is also a pure lambda term. If the trisection procedure is applied to a closed, pure lambda term, then the procedure will either give a result in finite time or never give a result. If the procedure gives a result in finite time, it will invariably give the result that the term is a function term. Hence, the trisection procedure degenerates to a bisection procedure on pure lambda terms. The definition

[$\mathcal{A} \equiv \mathcal{B} \Leftrightarrow \forall \mathcal{C} \in \bar{\mathcal{T}} : \mathcal{C}\,'\,\mathcal{A} \approx \mathcal{C}\,'\,\mathcal{B}$]

defines an equivalence relation on closed, pure [λ]-terms. The resulting theory, call it [$\lambda\emptyset$], is consistent in the sense that one can find two [λ]-terms [$\mathcal{A}$] and [$\mathcal{B}$] for which [$\mathcal{A} \equiv \mathcal{B}$] does not hold. As an example, [$\lambda x.\bot \equiv \bot$] does not hold since the left hand side is a function term and the right hand side is undecidable.

If one uses [$\lambda\emptyset$] as a programming language, one will get a language in which the only thing that distinguishes one program from the other is whether or not it terminates for given input. It is rather cumbersome to get meaningful output from such a program. As an example, consider a computable function [f] which, given some input [x], is expected to loop indefinitely or answer "yes" or "no" in finite time. To implement [f] in [$\lambda\emptyset$] one will have to implement two functions, [f_{yes}] and [f_{no}] for which [f_{yes}] terminates on input [x] if and only if the answer is "yes" and similarly for [f_{no}]. One may then run the two functions in parallel until one of them, if any, terminates.

No attempt has been reported on trying to base a theory of ZFC power on [$\lambda\emptyset$].

The discourse on comparing [$\lambda\mathsf{N}$] with pure lambda calculus ends here and the presentation of [$\lambda\mathsf{N}$] continues.

2.14 Elementary axioms and inference rules

$\lambda\mathsf{N}$ has 13 axiom schemes and inference rules. Nine of these will be referred to as *elementary axiom schemes or inference rules* and the remaining ones will be referred to as *advanced inference rules*. Chapter 4 presents the axiom schemes and inference rules formally.

$\lambda\mathsf{N}$ has three elementary axiom schemes about selection, three elementary inference rules on equality, and three elementary axiom schemes about application. The elementary axiom schemes about selection read:

$$\begin{array}{lcll}
[\ \text{if}(\mathsf{N},\mathcal{B},\mathcal{C}) & \equiv & \mathcal{B} &] \\
[\ \text{if}(\lambda x.\mathcal{A},\mathcal{B},\mathcal{C}) & \equiv & \mathcal{C} &] \\
[\ \text{if}(\bot,\mathcal{B},\mathcal{C}) & \equiv & \bot &]
\end{array}$$

where [$x \in \mathcal{V}$] and [$\mathcal{A},\mathcal{B},\mathcal{C} \in \mathcal{T}$]. Chapter 1 used unary turnstyle [$\vdash z$] to denote that [z] is provable. This paper will avoid unary turnstyle from now on.

The elementary inference rules about equality read:

$$\begin{array}{lclcll}
[\ \mathcal{A}\equiv\mathcal{B} & \vdash & \mathcal{A}\equiv\mathcal{C} & \vdash & \mathcal{B}\equiv\mathcal{C} &] \\
[\ \mathcal{A}\equiv\mathcal{B} & & & \vdash & \lambda x.\mathcal{A}\equiv\lambda x.\mathcal{B} &] \\
[\ \mathcal{A}\equiv\mathcal{B} & & & \vdash & \mathcal{C}\,\text{'}\mathcal{A}\equiv\mathcal{C}\,\text{'}\mathcal{B} &]
\end{array}$$

Binary turnstyle $\left[\ \boxed{x \vdash y}\ \right]$ denotes that if [x] is provable then [y] is provable. As stated in Appendix C, [$x \vdash y$] is right associative such that [$x \vdash y \vdash z$] means [$x \vdash (y \vdash z)$]. As an example, [$\mathcal{A}\equiv\mathcal{B} \vdash \mathcal{A}\equiv\mathcal{C} \vdash \mathcal{B}\equiv\mathcal{C}$] says that if [$\mathcal{A}\equiv\mathcal{B}$] and [$\mathcal{A}\equiv\mathcal{C}$] are provable then [$\mathcal{B}\equiv\mathcal{C}$] is provable. In [$x \vdash y$], [x] and [y] will be referred to as *premise* and *conclusion*, respectively.

The elementary axiom schemes about application read

$$\begin{array}{lclcll}
[& & \mathsf{N}\,\text{'}\mathcal{B} & \equiv & \mathsf{N} &] \\
[\ \mathcal{C}\simeq\langle\mathcal{A}\mid x{:=}\mathcal{B}\rangle & \Vdash & (\lambda x.\mathcal{A})\,\text{'}\mathcal{B} & \equiv & \mathcal{C} &] \\
[& & \bot\,\text{'}\mathcal{B} & \equiv & \bot &]
\end{array}$$

elementary rules
advanced rules
premise
conclusion

$\left[\boxed{x \Vdash y}\right]$ states that if [x] is true then [y] is provable. In [$x \Vdash y$], [x] is a *side condition*. Side conditions must be computable by machine; otherwise the theories they occur in are non-axiomatic.

The elementary rules above correspond to the lemmas in Section 2.9.

The advanced rules will be referred to under the following names:

- Quartum Non Datur (QND)
- Extensionality
- Monotonicity
- Minimality

The inference rule of extensionality is presented in Section 7.1. It expresses the extensionality property stated in Lemma 2.11.2. The lemmas of ZFC that correspond to the other three advanced rules are stated in the following.

2.15 Quartum Non Datur (QND)

The trisection procedure of λN (c.f. Section 2.8) divides all closed terms of λN into ur-terms, function terms, and undecidable terms. The trisection satisfies the following lemma:

ZFC lemma 2.15.1 (QND) For all [$\mathcal{A} \in \bar{\mathcal{T}}$]:

[$\mathcal{A}$] is an ur-term	if and only if	[$\mathcal{A} \equiv \mathsf{N}$]
[$\mathcal{A}$] is a function term	if and only if	[$\mathcal{A} \equiv \lambda x.\mathcal{A}\,' x$]
[$\mathcal{A}$] is an undecidable term	if and only if	[$\mathcal{A} \equiv \bot$]

The *Quartum Non Datur* (*QND*) inference of λN says that every map [m] satisfies

[$m \equiv \mathsf{N}$] or [$m \equiv \lambda x.m\,' x$] or [$m \equiv \bot$]

there is no fourth possibility. QND is counter-intuitionistic. QND is stated as an inference rule in Section 6.1.

2.16 Ordering of maps

The inference rules of monotonicity and minimality express properties of a partial order [$\mathcal{A} \preceq \mathcal{B}$] on closed terms. That partial order is introduced in the following.

For all [$\mathcal{A}, \mathcal{B} \in \bar{\mathcal{T}}$] define $\boxed{\mathcal{A} \le_{\text{root}} \mathcal{B}}$ by

[$\mathcal{A} \le_{\text{root}} \mathcal{B} \Leftrightarrow \mathcal{A} \approx \bot \vee \mathcal{A} \approx \mathcal{B}$].

side condition
Quartum Non Datur
QND

ZFC lemma 2.16.1 For all $[\,\mathcal{A},\mathcal{B},\mathcal{C}\in\bar{\mathcal{T}}\,]$:

$$\begin{array}{ll} [\,\mathcal{A}\leq_{\mathrm{root}}\mathcal{A} &] \\ [\,\mathcal{A}\leq_{\mathrm{root}}\mathcal{B}\wedge\mathcal{B}\leq_{\mathrm{root}}\mathcal{A}\Leftrightarrow\mathcal{A}\approx\mathcal{B} &] \\ [\,\mathcal{A}\leq_{\mathrm{root}}\mathcal{B}\wedge\mathcal{B}\leq_{\mathrm{root}}\mathcal{C}\Rightarrow\mathcal{A}\leq_{\mathrm{root}}\mathcal{C} &] \end{array}$$

The partial order has a property which is similar to extensionality:

ZFC lemma 2.16.2 For all $[\,\mathcal{A},\mathcal{B}\in\bar{\mathcal{T}}\,]$:

$$[\,\forall\mathcal{C}{\in}\bar{\mathcal{T}}{:}\,\mathcal{C}\,\text{'}\,\mathcal{A}\leq_{\mathrm{root}}\mathcal{C}\,\text{'}\,\mathcal{B}\Leftrightarrow\forall\mathcal{C}{\in}\bar{\mathcal{T}}^{<\omega}{:}\,\mathcal{A}\,\text{"}\,\mathcal{C}\leq_{\mathrm{root}}\mathcal{B}\,\text{"}\,\mathcal{C}\,].$$

For all $[\,\mathcal{A},\mathcal{B}\in\bar{\mathcal{T}}\,]$ define $\boxed{\mathcal{A}\preceq\mathcal{B}}$ by

$$[\,\mathcal{A}\preceq\mathcal{B}\Leftrightarrow\forall\mathcal{C}{\in}\bar{\mathcal{T}}^{<\omega}{:}\,\mathcal{A}\,\text{"}\,\mathcal{C}\leq_{\mathrm{root}}\mathcal{B}\,\text{"}\,\mathcal{C}\,].$$

If $[\,\mathcal{A}\,]$ or $[\,\mathcal{B}\,]$ or both are not closed, then

$$[\,\mathcal{A}\preceq\mathcal{B}\Leftrightarrow\lambda v_1\ldots\lambda v_n.\mathcal{A}\preceq\lambda v_1\ldots\lambda v_n.\mathcal{B}\,]$$

where $[\,v_1,\ldots,v_n\,]$ are the free variables of $[\,\mathcal{A}\,]$ and $[\,\mathcal{B}\,]$.
$[\,\mathcal{A}\preceq\mathcal{B}\,]$ defines a partial ordering of all terms:

ZFC lemma 2.16.3 For all $[\,\mathcal{A},\mathcal{B},\mathcal{C}\in\mathcal{T}\,]$:

$$\begin{array}{ll} [\,\mathcal{A}\preceq\mathcal{A} &] \\ [\,\mathcal{A}\preceq\mathcal{B}\wedge\mathcal{B}\preceq\mathcal{A}\Leftrightarrow\mathcal{A}\equiv\mathcal{B} &] \\ [\,\mathcal{A}\preceq\mathcal{B}\wedge\mathcal{B}\preceq\mathcal{C}\Rightarrow\mathcal{A}\preceq\mathcal{C} &] \end{array}$$

The fundamental constructs of $\lambda\mathsf{N}$ are monotonic with respect to $[\,\mathcal{A}\preceq\mathcal{B}\,]$:

ZFC lemma 2.16.4 For all $[\,x\in\mathcal{V}\,]$ and $[\,\mathcal{A},\mathcal{B},\mathcal{C},\dot{\mathcal{A}},\dot{\mathcal{B}},\dot{\mathcal{C}}\in\mathcal{T}\,]$:

$$\begin{array}{llll} [\,\mathcal{A}\preceq\dot{\mathcal{A}} & \Rightarrow & \lambda x.\mathcal{A}\preceq\lambda x.\dot{\mathcal{A}} &] \\ [\,\mathcal{A}\preceq\dot{\mathcal{A}}\wedge\mathcal{B}\preceq\dot{\mathcal{B}} & \Rightarrow & \mathcal{A}\,\text{'}\,\mathcal{B}\preceq\dot{\mathcal{A}}\,\text{'}\,\dot{\mathcal{B}} &] \\ [\,\mathcal{A}\preceq\dot{\mathcal{A}}\wedge\mathcal{B}\preceq\dot{\mathcal{B}}\wedge\mathcal{C}\preceq\dot{\mathcal{C}} & \Rightarrow & \mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C})\preceq\mathrm{if}(\dot{\mathcal{A}},\dot{\mathcal{B}},\dot{\mathcal{C}}) &] \end{array}$$

$[\,\mathsf{N}\,]$ is maximal, $[\,\lambda x.\bot\,]$ is a bottom element among function terms, and $[\,\bot\,]$ is a bottom element among all terms with respect to $[\,\mathcal{A}\preceq\mathcal{B}\,]$:

ZFC lemma 2.16.5 If $[\,x\in\mathcal{V}\,]$, $[\,\mathcal{A}\in\mathcal{T}\,]$, and $[\,\mathrm{notfree}(x;\mathcal{A})\,]$:

$$\begin{array}{llll} [\,\mathsf{N}\preceq\mathcal{A} & \Leftrightarrow & \mathsf{N}\equiv\mathcal{A} &] \\ [\,\lambda x.\bot\preceq\mathcal{A} & \Leftrightarrow & \mathcal{A}\equiv\lambda x.\mathcal{A}\,\text{'}\,x &] \\ [\,\bot\preceq\mathcal{A} & \Leftrightarrow & 0=0 &] \end{array}$$

2.17 Monotonicity

The inference rule of monotonicity of λN expresses the following lemma:

ZFC lemma 2.17.1 (Monotonicity) For all terms $[\,\mathcal{A}\,]$, $[\,\mathcal{B}\,]$, and $[\,\mathcal{C}\,]$,

$$[\,\mathcal{A} \preceq \mathcal{B} \Rightarrow \mathcal{C}\,\text{'}\,\mathcal{A} \preceq \mathcal{C}\,\text{'}\,\mathcal{B}\,].$$

The inference rule of monotonicity is stated in Section 7.9.

2.18 Minimality

Define

$$\left[\,\boxed{\mathsf{Y}} \doteq \lambda f.(\lambda x.f\,\text{'}(x\,\text{'}\,x))\,\text{'}(\lambda x.f\,\text{'}(x\,\text{'}\,x))\,\right].$$

The following lemma follows from the lemmas of Section 2.9 (or the discussion in Section 2.13):

ZFC lemma 2.18.1 If $[\,f \in \bar{\mathcal{T}}\,]$ then

$$[\,f\,\text{'}(\mathsf{Y}\,\text{'}\,f) \equiv \mathsf{Y}\,\text{'}\,f\,].$$

Hence, for all $[\,f \in \bar{\mathcal{T}}\,]$, $[\,\mathsf{Y}\,\text{'}\,f\,]$ is a solution to

$$[\,f\,\text{'}\,x \equiv x\,].$$

Solutions to $[\,f\,\text{'}\,x \equiv x\,]$ are known as *fixed points* of $[\,f\,]$, and $[\,\mathsf{Y}\,]$ is known as the *fixed point operator*.

An equation like $[\,f\,\text{'}\,x \equiv x\,]$ may have several solutions. Among these solutions, $[\,\mathsf{Y}\,\text{'}\,f\,]$ is minimal:

ZFC lemma 2.18.2 If $[\,f, x \in \bar{\mathcal{T}}\,]$ then

$$[\,f\,\text{'}\,x \equiv x \Rightarrow \mathsf{Y}\,\text{'}\,f \preceq x\,].$$

The following generalization also holds:

ZFC lemma 2.18.3 (Minimality) If $[\,f, x \in \bar{\mathcal{T}}\,]$ then

$$[\,f\,\text{'}\,x \preceq x \Rightarrow \mathsf{Y}\,\text{'}\,f \preceq x\,].$$

The inference rule of minimality of λN expresses this lemma. The inference rule of minimality is stated in Section 7.11.

---

fixed point
fixed point operator

2.19 Continuity

λN has a continuity property. That property will not be stated as an axiom since it does not carry over to MT. The continuity property is stated here in Section 2.19 for completeness. The rest of Section 2.19 can be skipped without loss of continuity (no pun intended).

For all sets [S] let $\boxed{S^\omega}$ denote the set of infinite sequences [$\langle s_0, s_1, s_2, \ldots\rangle$] of elements of [S]. Let $\boxed{\langle s_i \mid i \in \omega\rangle}$ denote [$\langle s_0, s_1, s_2, \ldots\rangle$].

A sequence [$\langle \mathcal{B}_0, \mathcal{B}_1, \ldots\rangle \in \bar{\mathcal{T}}^\omega$] is *monotonic* if

$$[\, \mathcal{B}_0 \preceq \mathcal{B}_1 \preceq \mathcal{B}_2 \preceq \cdots \,].$$

It is possible to construct a Scott model [$\mathcal{M}_S$] of λN which is *complete* in the sense that any monotonic sequence [$\mathcal{B} \in \mathcal{M}_S^\omega$] has a least upper bound $\left[\,\boxed{\sup \mathcal{B}}\,\right]$. That least upper bound will be referred to as the *suppremum* of [$\mathcal{B}$]. The term model is incomplete.

The Scott model satisfies the following continuity property for all [$\mathcal{A} \in \mathcal{M}_S$] and all monotonic [$\langle \mathcal{B}_0, \mathcal{B}_1, \ldots\rangle \in \mathcal{M}_S^\omega$]:

$$[\, \mathcal{A}\,\text{'}\sup \langle \mathcal{B}_i \mid i \in \omega\rangle = \sup \langle \mathcal{A}\,\text{'}\,\mathcal{B}_i \mid i \in \omega\rangle \,].$$

The corresponding continuity property in models of MT reads

$$[\, \mathcal{A}\,\text{'}\sup \langle \mathcal{B}_i \mid i \in \kappa\rangle = \sup \langle \mathcal{A}\,\text{'}\,\mathcal{B}_i \mid i \in \kappa\rangle \,].$$

where [κ] is a cardinal greater than [ω] ([κ] is a regular cardinal greater than some inaccessible cardinal [σ]). The latter continuity property is referred to as [κ]-continuity [3].

2.20 Greatest lower bounds

From now on,

$$\left[x \begin{cases} y \\ z \end{cases} \right]$$

will be used as shorthand for

$$[\, \text{if}(x, y, z) \,].$$

Furthermore, let [$x \downarrow y$] be defined by the following recursive definition:

$$\left[x \downarrow y \doteq x \begin{cases} \text{if}(y, \mathsf{N}, \bot) \\ \text{if}(y, \bot, \lambda z. x\,\text{'}\,z \downarrow y\,\text{'}\,z) \end{cases} \right].$$

monotonic
complete
suppremum

A more pedantic, non-recursive, ZFC definition of $[\ x \downarrow y\ ]$ reads:

$$\left[\ \mathrm{Min} \doteq \lambda m.\lambda x.\lambda y.x \left\{ \begin{array}{l} \mathrm{if}(y, \mathsf{N}, \bot) \\ \mathrm{if}(y, \bot, \lambda z.m\,'(x\,'\,z)\,'(y\,'\,z)) \end{array}\right. \ \right]$$

$$[\ x \downarrow y \doteq \mathsf{Y}\,'\,\mathrm{Min}\,'\,x\,'\,y\ ]$$

In any case, for all $[\ x, y \in \bar{\mathcal{T}}\ ]$, $[\ x \downarrow y\ ]$ satisfies

$$\left[\ x \downarrow y \equiv x \left\{ \begin{array}{l} \mathrm{if}(y, \mathsf{N}, \bot) \\ \mathrm{if}(y, \bot, \lambda z.x\,'\,z \downarrow y\,'\,z) \end{array}\right. \ \right].$$

The construct has the following properties:

ZFC lemma 2.20.1 For all $[\ x, y, z \in \bar{\mathcal{T}}\ ]$:

$$\begin{array}{ll} [\ x \downarrow y \preceq x &] \\ [\ x \downarrow y \preceq y &] \\ [\ z \preceq x \wedge z \preceq y \Rightarrow z \preceq x \downarrow y &] \end{array}$$

In other words: $[\ x \downarrow y\ ]$ is a greatest lower bound of $[\ x\ ]$ and $[\ y\ ]$. If two terms $[\ u\ ]$ and $[\ v\ ]$ are both greatest lower bounds of $[\ x\ ]$ and $[\ y\ ]$ then $[\ u \equiv v\ ]$. Hence, the greatest lower bound of any two maps is unique up to equality of maps.

A lemma of particular interest reads:

ZFC lemma 2.20.2 For all $[\ x, y \in \bar{\mathcal{T}}\ ]$:

$$[\ x \preceq y \Leftrightarrow x \equiv x \downarrow y\ ].$$

This allows to define $[\ x \preceq y\ ]$ inside λN. Section 7.3 defines $[\ x \preceq y\ ]$ as shorthand for $[\ x \equiv x \downarrow y\ ]$. This is convenient since it allows to state the inference rules of monotonicity and minimality inside the language of λN.

Chapter 3

Programming in $\lambda\mathsf{N}$

3.1 Introduction

Chapter 3 presents $\lambda\mathsf{N}$ as a programming language. The chapter has three aims. First aim is to give a better understanding of what maps are and how they can be used. Second aim is to show how to represent quantities like natural numbers and truth values in $\lambda\mathsf{N}$. Third aim is to give a number of definitions within $\lambda\mathsf{N}$ which are going to be used in axioms, lemmas, and proofs in later chapters.

There exist tricks that allow computer implementations of $\lambda\mathsf{N}$ to perform input/output and to compete on efficiency with programming languages like C and even assembler, but such coding tricks are outside the scope of the present paper. In the present paper we shall merely consider $\lambda\mathsf{N}$ as a programming language from a theoretical point of view.

3.2 Overloading

Section 2.2 stated that in Chapter 2, [$\mathcal{A} \wedge \mathcal{B}$] stood for conjunction in ZFC. In this chapter, all terms are terms of $\lambda\mathsf{N}$. As an example, in this chapter, [$\mathcal{A} \wedge \mathcal{B}$] denotes conjunction in $\lambda\mathsf{N}$. [$\mathcal{A} \wedge \mathcal{B}$] is defined in Section 3.6.

3.3 The syntax of $\lambda\mathsf{N}$

The syntax of variables [$\mathcal{V}$] and terms [$\mathcal{T}$] of $\lambda\mathsf{N}$ reads:

$$\begin{array}{llll} [\,\mathcal{V} & ::= & x \mid y \mid z \mid \cdots &] \\ [\,\mathcal{T} & ::= & \mathcal{V} \mid \mathsf{N} \mid \lambda\mathcal{V}.\mathcal{T} \mid \mathcal{T}\,\text{'}\,\mathcal{T} \mid \mathrm{if}(\mathcal{T},\mathcal{T},\mathcal{T}) &] \end{array}$$

The definition of [$\mathcal{V}$] says that [x], [y], [z] and so on are variables. If $\lambda\mathsf{N}$ is implemented on a computer one of course has to be replace "and so on" with something more specific.

The definition of [$\mathcal{T}$] says that if [x] is a variable and [$\mathcal{A}$], [$\mathcal{B}$], and [$\mathcal{C}$] are terms, then [x], [N], [$\lambda x.\mathcal{A}$], [$\mathcal{A}\,\text{'}\,\mathcal{B}$], and [$\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C})$] are all terms.

3.4 The definition of $[\,\perp\,]$

The definition of $[\,\perp\,]$ reads

$$[\,\perp \doteq (\lambda x.x\,\text{'}\,x)\,\text{'}(\lambda x.x\,\text{'}\,x)\,].$$

3.5 Representation of truth values

Whenever a concept like truth values is going to be represented in $\lambda\mathsf{N}$, it is convenient to introduce two representations. The two representations will be referred to as the *canonical representation* and the *liberal representation*, respectively. For truth values, the following canonical representation is chosen in this paper:

$[\,\mathsf{N}\,]$	canonically represents truth and
$[\,\lambda x.\mathsf{N}\,]$	canonically represents falsehood

The following liberal representation is chosen in this paper:

the ur-element	liberally represents truth and
any function	liberally represents falsehood.

The two representations fit together in the sense that the canonical representation of truth is also a liberal representation of truth and similarly for falsehood.

In general, a canonical representation assigns exactly one map to each concept whereas a liberal representation may assign one or more.

The following two definitions express the canonical representation:

$$\begin{array}{llll} [\,\mathsf{T} & \doteq & \mathsf{N} &] \text{ and} \\ [\,\mathsf{F} & \doteq & \lambda x.\mathsf{N} &] \;. \end{array}$$

For any pair of representations, one may define the *retract* of the pair as follows: The retract is a function $[\,f(x)\,]$. If $[\,x\,]$ is a map and if $[\,x\,]$ is a liberal representation of some concept, then $[\,f(x)\,]$ must be the corresponding canonical representation of the concept. If $[\,x\,]$ is not the liberal representation of some concept, then $[\,f(x)\,]$ must equal $[\,\perp\,]$ (the map $[\,\perp\,]$ should never be used as the representation of anything). A retract $[\,f\,]$ must satisfy $[\,f(f(x)) \equiv f(x)\,]$ for all maps $[\,x\,]$. The retract of the representation of truth values will be named $[\,\boxed{\approx x}\,]$ and may be defined thus:

$$[\,\approx x \doteq \text{if}(x, \mathsf{T}, \mathsf{F})\,].$$

One should always choose representations such that the retract is expressible in the theory.

---

canonical representation
liberal representation
retract

In the following, ***representation*** will mean "liberal representation" unless otherwise noted. As an example, $[\ \lambda x.\mathsf{N}\ ]$ and $[\ \lambda x.x\ ]$ both represent falsehood. Only $[\ \lambda x.\mathsf{N}\ ]$ canonically represents falsehood.

In principle, the choice of representation of truth values is arbitrary. In practice, however, it turns out to be very convenient to let the ur-element represent truth, c.f. Section 3.13 and Section 5.11.

3.6 Logical connectives

As in Section 2.20 define:

$$\left[x\left\{ \begin{array}{c} y \\ z \end{array} \right. \doteq \text{if}(x,y,z) \right].$$

The usual logical connectives may be defined thus:

$$\begin{array}{lcll}
[\ \boxed{x \wedge y} & \doteq & x\left\{ \begin{array}{l} \text{if}(y,\mathsf{T},\mathsf{F}) \\ \text{if}(y,\mathsf{F},\mathsf{F}) \end{array} \right. &] \\
[\ \boxed{x \vee y} & \doteq & x\left\{ \begin{array}{l} \text{if}(y,\mathsf{T},\mathsf{T}) \\ \text{if}(y,\mathsf{T},\mathsf{F}) \end{array} \right. &] \\
[\ \boxed{x \Rightarrow y} & \doteq & x\left\{ \begin{array}{l} \text{if}(y,\mathsf{T},\mathsf{F}) \\ \text{if}(y,\mathsf{T},\mathsf{T}) \end{array} \right. &] \\
[\ \boxed{x \Leftarrow y} & \doteq & x\left\{ \begin{array}{l} \text{if}(y,\mathsf{T},\mathsf{T}) \\ \text{if}(y,\mathsf{F},\mathsf{T}) \end{array} \right. &] \\
[\ \boxed{x \Leftrightarrow y} & \doteq & x\left\{ \begin{array}{l} \text{if}(y,\mathsf{T},\mathsf{F}) \\ \text{if}(y,\mathsf{F},\mathsf{T}) \end{array} \right. &]
\end{array}$$

One may of course also define the logical connectives simpler:

$$\begin{array}{lcll}
[\ \boxed{x \mathbin{\tilde{\wedge}} y} & \doteq & \text{if}(x,y,\mathsf{F}) &] \\
[\ \boxed{x \mathbin{\tilde{\vee}} y} & \doteq & \text{if}(x,\mathsf{T},y) &] \\
[\ \boxed{x \mathrel{\tilde{\Rightarrow}} y} & \doteq & \text{if}(x,y,\mathsf{T}) &] \\
[\ \boxed{x \mathrel{\tilde{\Leftarrow}} y} & \doteq & y \mathrel{\tilde{\Rightarrow}} x &]
\end{array}$$

However, the more complicated definitions lead to nicer mathematical properties. As an example,

$$\begin{array}{ll}
[\ x \wedge y \equiv y \wedge x\ ] & \text{holds, whereas} \\
[\ x \mathbin{\tilde{\wedge}} y \equiv y \mathbin{\tilde{\wedge}} x\ ] & \text{fails.}
\end{array}$$

$[\ x \wedge y \equiv y \wedge x\ ]$ is proved in Section 6.3. $[\ x \mathbin{\tilde{\wedge}} y \equiv y \mathbin{\tilde{\wedge}} x\ ]$ fails e.g. for $[\ x \equiv \mathsf{F}\ ]$ and $[\ y \equiv \perp\ ]$.

representation

Logical negation is defined the obvious way:

$$[\ \boxed{\neg x} \doteq \text{if}(x, \mathsf{F}, \mathsf{T})\ ].$$

Logical negation has some properties which resemble the properties of negation in intuitionistic logic:

$[\ \neg\neg x \equiv x\ ]$ fails, whereas
$[\ \neg\neg\neg x \equiv \neg x\ ]$ holds.

$[\ \neg\neg x \equiv x\ ]$ fails e.g. for $[\ x \equiv \lambda z.z\ ]$. $[\ \neg\neg\neg x \equiv \neg x\ ]$ is proved in Section 6.4.

As proved in Section 6.4, the retract satisfies

$$[\ \approx x \equiv \neg\neg x\ ].$$

The construct $[\ !x\ ]$ equals $[\ \mathsf{T}\ ]$ if $[\ x\ ]$ represents a truth value and equals $[\ \bot\ ]$ otherwise:

$$\Big[\ \boxed{!x} \doteq \text{if}(x, \mathsf{T}, \mathsf{T})\ \Big].$$

As an example of use,

$[\ x \vee \neg x \equiv \mathsf{T}\ ]$ fails, whereas
$[\ x \vee \neg x \equiv\ !x\ ]$ holds.

$[\ x \vee \neg x \equiv \mathsf{T}\ ]$ fails for $[\ x \equiv \bot\ ]$. $[\ x \vee \neg x \equiv\ !x\ ]$ is proved in Section 6.4.

The construct $[\ ¡x\ ]$ equals $[\ \mathsf{F}\ ]$ if $[\ x\ ]$ represents a truth value and equals $[\ \bot\ ]$ otherwise:

$$\Big[\ \boxed{¡x} \doteq \text{if}(x, \mathsf{F}, \mathsf{F})\ \Big].$$

As an example of use,

$[\ x \wedge \neg x \equiv \mathsf{F}\ ]$ fails, whereas
$[\ x \wedge \neg x \equiv\ ¡x\ ]$ holds.

$[\ x \wedge \neg x \equiv \mathsf{F}\ ]$ fails for $[\ x \equiv \bot\ ]$. $[\ x \wedge \neg x \equiv\ ¡x\ ]$ is proved in Section 6.4.

3.7 Conclusion on truth values

Now truth values have been introduced in $\lambda\mathsf{N}$ in the sense that

- a canonical and a liberal representation have been chosen,
- the canonical representation has been implemented through the definitions of $[\ \mathsf{T}\ ]$ and $[\ \mathsf{F}\ ]$,
- the liberal representation has been implemented through the definition of $[\ \approx x\ ]$, and
- the most common operators like $[\ \neg x\ ]$, $[\ x \wedge y\ ]$, $[\ x \vee y\ ]$, $[\ x \Rightarrow y\ ]$, and $[\ x \Leftrightarrow y\ ]$ have been defined.

3.8 Recursive definitions

As in Section 2.18 define

$$[\, \mathsf{Y} \doteq \lambda f.(\lambda x.f\,\text{'}(x\,\text{'}\,x))\,\text{'}(\lambda x.f\,\text{'}(x\,\text{'}\,x))\,].$$

As mentioned in Section 2.18, [Y] satisfies

$$[\, \mathsf{Y}\,\text{'}\,f \equiv f\,\text{'}(\mathsf{Y}\,\text{'}\,f)\,].$$

Section 2.20 demonstrated how recursive definitions can be converted into non-recursive definitions using [Y]. From now on, recursive definitions are permitted in the λN programming language. One should consider recursive definitions as shorthand for non-recursive definitions that involve [Y].

3.9 Representation of natural numbers

Section 3.9 presents a representation of the natural numbers. The canonical representation of the natural number [n] is chosen to be:

$$\left[\, \underbrace{\lambda y.\lambda y.\cdots\lambda y.}_{n}\,\mathsf{N} \,\right].$$

Accordingly, the number [0] and the successor function [x^+] is defined thus:

$$[\, \boxed{0} \doteq \mathsf{N} \,]]$$

$$\left[\, \boxed{x^+} \doteq \lambda y.x \,\right]$$

The first few natural numbers are:

$$[\, 1 \doteq 0^+ \,]$$

$$[\, 2 \doteq 0^{++} \,]$$

$$[\, 3 \doteq 0^{+++} \,]$$

$$[\, 4 \doteq 0^{++++} \,]$$

$$[\, 5 \doteq 0^{+++++} \,]$$

Next step is to choose a liberal representation. To do so, we first define the predecessor function [x^-] as follows:

$$\left[\, \boxed{x^-} \doteq x\,\text{'}\,\mathsf{N} \,\right].$$

As an example of use, [$3^- \equiv 2$]:

$$[\, 3^- \equiv (\lambda y.\lambda y.\lambda y.\mathsf{N})\,\text{'}\,\mathsf{N} \equiv \lambda y.\lambda y.\mathsf{N} \equiv 2 \,].$$

A map is said to represent a natural number liberally if repeated application of the predecessor function eventually yields [0]. Hence, a map [x] liberally represents a natural number if an only if $\left[\tilde{\mathbf{N}}\text{'}x \equiv \mathsf{T} \right]$ where

$$\left[\boxed{\tilde{\mathbf{N}}} \equiv \lambda x.x \left\{ \begin{array}{l} \mathsf{T} \\ \tilde{\mathbf{N}}\text{'}(x^-) \end{array} \right. \right].$$

$[\, \tilde{\mathbf{N}} \,]$ is the *characteristic function* of the set [**N**] of natural numbers. Characteristic functions offer a primitive way of representing sets until Chapter 9 presents something better.

If $\left[\tilde{\mathbf{N}}\text{'}x \equiv \mathsf{T} \right]$ then [x] liberally represents the least natural number [n] for which

$$\left[x^{\overbrace{- \; - \; \cdots \; -}^{n}} \equiv 0 \right].$$

The retract [$\approx_{\mathbf{N}}$] of the representation reads

$$\left[\boxed{\approx_{\mathbf{N}}} \doteq \lambda x.x \left\{ \begin{array}{l} 0 \\ (\approx_{\mathbf{N}}\text{'}(x^-))^+ \end{array} \right. \right].$$

3.10 Natural numbers versus truth values

In the representations stated so far, [N] represents both truth and [0] and [$\lambda x.\mathsf{N}$] represents both falsehood and [1]. In practical programming, such "overloading" is inconvenient in the long run. See [6] for a more convenient but more complicated representation. The representations above, however, suffice for the present paper.

3.11 Equality, addition, and multiplication

The construct [$x = y$] equals [T] if [x] and [y] represent the same natural number and equals [F] if [x] and [y] represent different natural numbers. The construct equals [$\bot$] if [x] or [y] or both do not represent a natural number:

$$\left[x = y \doteq x \left\{ \begin{array}{l} y \left\{ \begin{array}{l} \mathsf{T} \\ \neg \tilde{\mathbf{N}}\text{'}y \end{array} \right. \\ \\ y \left\{ \begin{array}{l} \neg \tilde{\mathbf{N}}\text{'}x \\ x^- = y^- \end{array} \right. \end{array} \right. \right].$$

characteristic function

The construct [$x + y$] equals the canonical representation of the sum of [x] and [y] if both represent natural numbers and equal [$\perp$] otherwise:

$$\left[x + y \doteq x \begin{cases} \approx_{\mathbf{N}} \text{'}\, y \\ (x^- + y)^+ \end{cases} \right].$$

The construct [$x \cdot y$] equals the canonical representation of the product of [x] and [y] if both represent natural numbers and equal [$\perp$] otherwise:

$$\left[x \cdot y \doteq x \begin{cases} \tilde{\mathbf{N}} \text{'}\, y \\ x^- \cdot y + y \end{cases} \right].$$

One may of course also define the arithmetic operations simpler:

$$\left[x \mathrel{\tilde{\doteq}} y \doteq x \begin{cases} y \\ y \begin{cases} \mathsf{F} \\ x^- \mathrel{\tilde{\doteq}} y^- \end{cases} \end{cases} \right]$$

$$\left[x \mathbin{\tilde{+}} y \doteq x \begin{cases} y \\ (x^- \mathbin{\tilde{+}} y)^+ \end{cases} \right]$$

$$\left[x \mathbin{\tilde{\cdot}} y \doteq x \begin{cases} 0 \\ x^- \mathbin{\tilde{\cdot}} y + y \end{cases} \right]$$

However, like in Section 3.6 the more complicated definitions lead to nicer mathematical properties. As an example,

[$x + y \quad \equiv \quad y + x$] holds, whereas
[$x \mathbin{\tilde{+}} y \quad \equiv \quad y \mathbin{\tilde{+}} x$] fails.

3.12 Conclusion on natural numbers

Now natural numbers have been introduced in $\lambda\mathsf{N}$ in the sense that

- a canonical and a liberal representation have been chosen,
- the canonical representation has been implemented through the definitions of [0] and [x^+],
- the liberal representation has been implemented through the definition of [$\approx_{\mathbf{N}}$], and
- the most common operators like [x^-], [$x = y$], [$x + y$], and [$x \cdot y$] have been defined.

3.13 Readability of conditionals

The right hand side of

$$[\, x \mathbin{\tilde{+}} y \doteq \mathrm{if}(x, y, (x^- \mathbin{\tilde{+}} y)^+) \,]$$

consists of a conditional whose condition is

$$[\, x \,]$$

and whose two cases are

$$[\, y \,]$$

and

$$[\, (x^- \mathbin{\tilde{+}} y)^+ \,].$$

Conditionals tend to be most readable if the simple case is stated before the complicated one as is done above. As an example of that, an expression of form

$$\left[\begin{array}{l} \mathrm{if}(\mathcal{C}_1, \mathcal{A}_1, \\ \quad \mathrm{if}(\mathcal{C}_2, \mathcal{A}_2, \\ \quad\quad \mathrm{if}(\mathcal{C}_3, \mathcal{A}_3, \\ \quad\quad\quad \ddots \\ \quad\quad\quad\quad \mathrm{if}(\mathcal{C}_{n-1}, \mathcal{A}_{n-1}, \\ \quad\quad\quad\quad\quad \mathrm{if}(\mathcal{C}_n, \mathcal{A}_n, \mathcal{B}))\cdots))) \end{array}\right]$$

is easier to read than an expression of form

$$\left[\begin{array}{l} \mathrm{if}(\mathcal{C}_1, \\ \quad \mathrm{if}(\mathcal{C}_2, \\ \quad\quad \mathrm{if}(\mathcal{C}_3, \\ \quad\quad\quad \vdots \\ \quad\quad\quad\quad \mathrm{if}(\mathcal{C}_{n-1}, \\ \quad\quad\quad\quad\quad \mathrm{if}(\mathcal{C}_n, \mathcal{B}, \mathcal{A}_n), \\ \quad\quad\quad\quad \mathcal{A}_{n-1}), \\ \quad\quad\quad \vdots \\ \quad\quad \mathcal{A}_3), \\ \quad \mathcal{A}_2), \\ \mathcal{A}_1) \end{array}\right]$$

The former statement has form $[\, \mathrm{if}(\mathcal{C}_1, \mathcal{A}_1, \mathcal{Z}) \,]$ where $[\, \mathcal{A}_1 \,]$ is simple and $[\, \mathcal{Z} \,]$ is complicated, which leads to high readability. The latter statement has form $[\, \mathrm{if}(\mathcal{C}_1, \mathcal{Z}, \mathcal{A}_1) \,]$ which leads to low readability.

The conditional

$$[\, \mathrm{if}(x, y, z) \,]$$

tests whether [x] is the ur-element or a function. The conditional equals [y] if [x] is the ur-element and equals [z] if [x] is a function. Therefore, [x], [y], and [z] will be referred to as the "condition", the "ur-case", and the "function case", respectively.

In recursive definitions, the function case normally is more complicated than the ur-case, and that is why the order

$$[\ \mathrm{if}(\mathrm{condition}, \mathrm{ur\text{-}case}, \mathrm{function\ case})\ ]$$

has been chosen for [$\mathrm{if}(x, y, z)$]. From a mathematical point of view, the order of arguments is irrelevant, but from a readability point of view, the order is important.

In conditions, however, it is customary (and convenient) to put the true case before the false case as in

$$[\ \mathrm{if}(\mathrm{condition}, \mathrm{true\ case}, \mathrm{false\ case})\ ].$$

Hence, to obtain readability and stick to convention at the same time, it is convenient to let the ur-term represent truth and to let functions represent falsehood. Hence, there is at least one good reason why ur-terms are chosen to represent truth and functions are chosen to represent falsehood. Another reason is stated in Section 5.11.

3.14 Pairs and tuples

Pairs may be introduced thus:

$$\left[\ \boxed{x :: y} \doteq \lambda z.\mathrm{if}(z, x, y)\ \right]$$

$$\left[\ \boxed{\mathbf{head}(x)} \doteq x\ \text{'}\ \mathsf{N}\ \right]$$

$$\left[\ \boxed{\mathbf{tail}(x)} \doteq x\ \text{'}\ \lambda x.\mathsf{N}\ \right]$$

Reduction of [$\mathbf{head}(x :: y)$] proceeds thus:

$$\begin{array}{ll} [& \mathbf{head}(x :: y) \simeq \\ & (\lambda z.\mathrm{if}(z, x, y))\ \text{'}\ \mathsf{N} \overset{+}{\to} \\ & \mathrm{if}(\mathsf{N}, x, y) \overset{+}{\to} \\ & x\ ] \end{array}$$

In conclusion,

$$[\ \mathbf{head}(x :: y) \equiv x\ ].$$

Likewise,

$$[\ \mathbf{tail}(x :: y) \equiv y\ ].$$

Tuples may be introduced thus:

$$\left[\ \boxed{\langle\,\rangle} \doteq \mathsf{N}\ \right], \text{ and}$$

$$\left[\ \boxed{\langle x_1, x_2, \ldots, x_n\rangle} \doteq x_1 :: x_2 :: \cdots :: x_n :: \mathsf{N}\ \right].$$

Tuples satisfy the following:

$$\begin{array}{lcl}
[\ \mathbf{head}(\langle\,\rangle) & \equiv & \langle\,\rangle\ ] \\
[\ \mathbf{tail}(\langle\,\rangle) & \equiv & \langle\,\rangle\ ] \\
[\ \mathbf{head}(\langle x_1, \ldots, x_n\rangle) & \equiv & x_1\ ] \\
[\ \mathbf{tail}(\langle x_1, \ldots, x_n\rangle) & \equiv & \langle x_2, \ldots, x_n\rangle\ ]
\end{array}$$

Now define

$$\left[\ \boxed{n_1(x)} \doteq \mathbf{head}(x)\ \right],$$

$$\left[\ \boxed{n_2(x)} \doteq \mathbf{head}(\mathbf{tail}(x))\ \right], \text{ and}$$

$$\left[\ \boxed{n_3(x)} \doteq \mathbf{head}(\mathbf{tail}(\mathbf{tail}(x)))\ \right].$$

Tuples satisfy the following:

$$\begin{array}{lcl}
[\ n_1(\langle x, y\rangle) & \equiv & x\ ] \\
[\ n_2(\langle x, y\rangle) & \equiv & y\ ] \\
[\ n_1(\langle x, y, z\rangle) & \equiv & x\ ] \\
[\ n_2(\langle x, y, z\rangle) & \equiv & y\ ] \\
[\ n_3(\langle x, y, z\rangle) & \equiv & z\ ]
\end{array}$$

Chapter 4

Presentation of $\lambda\mathsf{N}$

4.1 Introduction

Chapter 4 presents $\lambda\mathsf{N}$ as an axiomatic theory. The reader is assumed to have an intuitive understanding of what a map is from Chapter 2 and Chapter 3, so inference rules and axiom schemes are stated with little or no explanation. Rather, focus will be on proof techniques within $\lambda\mathsf{N}$.

$\lambda\mathsf{N}$ is a subtheory of full MT, and all proof techniques presented in Chapter 4 carry over to MT.

Chapter 4 also presents the meta-language in which axiomatic theories are presented in this paper.

4.2 Overloading

Section 2.2 stated that in Chapter 2, $[\,\mathcal{A} \wedge \mathcal{B}\,]$ stood for conjunction in ZFC. In this chapter, all terms and formulas (i.e. well-formed formulas) are terms and formulas, respectively, of $\lambda\mathsf{N}$ unless otherwise noted. As an example, in this chapter, $[\,\mathcal{A} \wedge \mathcal{B}\,]$ denotes conjunction in $\lambda\mathsf{N}$. $[\,\mathcal{A} \wedge \mathcal{B}\,]$ was defined in Section 3.6.

4.3 The syntax of $\lambda\mathsf{N}$

Let $\left[\,\boxed{\mathcal{V}}\,\right]$, $\left[\,\boxed{\mathcal{T}}\,\right]$, and $\left[\,\boxed{\mathcal{F}}\,\right]$ denote the syntax classes of variables, terms, and formulas, respectively, of $\lambda\mathsf{N}$. The statement

$$\left[\,\boxed{\text{syntaxclass}}\,\mathcal{V};\mathcal{T};\mathcal{F}\,\right]^*$$

states that $[\,\mathcal{V}\,]$, $[\,\mathcal{T}\,]$, and $[\,\mathcal{F}\,]$ are syntax classes. All formulas that are enclosed in square brackets and decorated with an asterisk are intended to be rigorous to the degree where they make sense to a mechanical proof checker. In contrast, [syntaxclass $\mathcal{W}$] does not introduce [$\mathcal{W}$] as a syntax class because of the missing asterisk.

The statement

$$[\ \mathcal{V} \boxed{::=} x; y; z\ ]^*$$

says that [x], [y], and [z] belong to the syntax class [$\mathcal{V}$]. In other words, the statement classifies [x], [y], and [z] as variables of $\lambda\mathsf{N}$. Variables of $\lambda\mathsf{N}$ will be referred to as *concrete variables* as opposed to metavariables that are introduced in Section 4.7. The statement

$$[\ \mathcal{V} ::= u; v; w\ ]^*$$

adds [u], [v], and [w] to the syntax class [$\mathcal{V}$]. There is no limit on the number of variables.

The syntax class [$\mathcal{T}$] of terms reads:

$$[\ \mathcal{T} ::= \mathcal{V}; \mathsf{N}; \lambda\mathcal{V}.\mathcal{T}; \mathcal{T}\,\text{'}\,\mathcal{T}; \text{if}(\mathcal{T},\mathcal{T},\mathcal{T})\ ]^*.$$

The definitions says: Every variable is a term; [N] is a term; if [x] is a variable and [y] is a term then [$\lambda x.y$] is a term and so on.

The syntax class [$\mathcal{F}$] of formulas reads:

$$[\ \mathcal{F} ::= \mathcal{T} \equiv \mathcal{T}\ ]^*.$$

4.4 Relation to MT

The terms of MT form a superset of the terms of $\lambda\mathsf{N}$ in that MT has two more fundamental constructs. Those two constructs will be added to [$\mathcal{F}$] in Chapter 9. Otherwise, the syntax of $\lambda\mathsf{N}$ and MT is the same.

The inference rules and axiom schemes of MT form a superset of the inference rules and axiom schemes of $\lambda\mathsf{N}$ in that MT has eight more axiom schemes (c.f. Section 2.14).

In consequence, all terms of $\lambda\mathsf{N}$ are also terms of MT, all formulas of $\lambda\mathsf{N}$ are also formulas of MT, and all theorems of $\lambda\mathsf{N}$ are also theorems of MT.

4.5 Definitions

As in Section 3.4, the definition of [$\bot$] reads

$$[\ \bot \doteq (\lambda x.x\,\text{'}\,x)\,\text{'}(\lambda x.x\,\text{'}\,x)\ ]^*.$$

A definition like the one above indicates that [$\bot$] is shorthand for [$(\lambda x.x\,\text{'}\,x)\,\text{'}\,(\lambda x.x\,\text{'}\,x)$]. In theorems and proofs, one has to replace all occurrences of [$\bot$] by [$(\lambda x.x\,\text{'}\,x)\,\text{'}(\lambda x.x\,\text{'}\,x)$] before checking the correctness of the proof.

concrete variable

In mechanical proof checkers it may be more efficient to treat definitions as new axioms, but the convention that definitions introduce shorthand notation suffice for the present paper.

The definitions in Chapter 3 of $[\, x \wedge y \,]$, $[\, x + y \,]$, and so on are in effect here in Chapter 4.

4.6 Axiomatic systems

From now on, the word *rule* will be used as a common name for axiom schemes and inference rules. Until further, we shall use $\left[\, \boxed{\text{Map}} \,\right]$ to denote the collection of rules of λN. From Chapter 9, [Map] will be extended to cover MT. In any case, [Map] is an axiomatic system:

$$\left[\, \boxed{\text{Axiomatic system}}\, \text{Map} \,\right]^*$$

[Map] is shorthand for $[\, \mathcal{R}_1; \cdots; \mathcal{R}_n \,]$ where $[\, \mathcal{R}_1; \cdots; \mathcal{R}_n \,]$ are the rules that are stated in the following.

4.7 Metaquantifiers

The rule

$$[\, \langle \mathcal{B}, \mathcal{C} \in \mathcal{T} \rangle \text{if}(\mathsf{N}, \mathcal{B}, \mathcal{C}) \equiv \mathcal{B} \,]$$

states that $[\, \text{if}(\mathsf{N}, \mathcal{B}, \mathcal{C}) \,]$ equals $[\, \mathcal{B} \,]$ for all terms $[\, \mathcal{B} \,]$ and $[\, \mathcal{C} \,]$. In general, the *metaquantifier* $\left[\, \boxed{\langle x \in y \rangle z} \,\right]$ states that $[\, z \,]$ holds whenever $[\, x \,]$ belongs to the syntax class $[\, y \,]$. $[\, \langle u, v \in w \rangle z \,]$ is shorthand for $[\, \langle u \in w \rangle \langle v \in w \rangle z \,]$ and $[\, \langle u \in v; x \in y \rangle z \,]$ is shorthand for $[\, \langle u \in v \rangle \langle x \in y \rangle z \,]$. The construct $[\, \langle x \in y \rangle z \,]$ temporarily classifies $[\, x \,]$ as a *metavariable* regardless of what meaning $[\, x \,]$ has outside the scope of the metaquantifier. As an example,

$$[\, \langle x \in \mathcal{V}; \mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathcal{T} \rangle \text{if}(\lambda x.\mathcal{A}, \mathcal{B}, \mathcal{C}) \equiv \mathcal{C} \,]$$

temporarily classifies $[\, x \,]$ as a metavariable that can denote any concrete variable. Outside the scope of the metaquantifier, $[\, x \,]$ denotes a specific, concrete variable.

4.8 Labels

$\left[\, \boxed{x\colon y} \,\right]$ means the same as $[\, y \,]$ but also introduces $[\, x \,]$ as shorthand for $[\, y \,]$. Hence, as an example,

$$[\, \text{IfNil}\colon \langle \mathcal{B}, \mathcal{C} \in \mathcal{T} \rangle \text{if}(\mathsf{N}, \mathcal{B}, \mathcal{C}) \equiv \mathcal{B} \,]$$

introduces [IfNil] is shorthand for $[\, \langle \mathcal{B}, \mathcal{C} \in \mathcal{T} \rangle \text{if}(\mathsf{N}, \mathcal{B}, \mathcal{C}) \equiv \mathcal{B} \,]$.

rule
metaquantifier
metavariable

4.9 Elementary rules about selection

As mentioned in Section 2.14, the elementary rules about selection read:

[Map rule

IfNil:	$\langle \mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle$	$\mathsf{if}(\mathsf{N},\mathcal{B},\mathcal{C})$	$\equiv$	$\mathcal{B}$	;
IfLambda:	$\langle x{\in}\mathcal{V};\mathcal{A},\mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle$	$\mathsf{if}(\lambda x.\mathcal{A},\mathcal{B},\mathcal{C})$	$\equiv$	$\mathcal{C}$	;
IfBottom:	$\langle \mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle$	$\mathsf{if}(\bot,\mathcal{B},\mathcal{C})$	$\equiv$	$\bot$	]*

$\left[\boxed{x \text{ rule } y}\right]$ states that the rules listed in [y] are rules of [x]. The rules are separated by semicolons. The statement above says that [Map] is shorthand for [$\langle \mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle\mathsf{if}(\mathsf{N},\mathcal{B},\mathcal{C}) \equiv \mathcal{B}; \langle x{\in}\mathcal{V};\mathcal{A},\mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle\mathsf{if}(\lambda x.\mathcal{A},\mathcal{B},\mathcal{C}) \equiv \mathcal{C}; \langle \mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle\mathsf{if}(\bot,\mathcal{B},\mathcal{C}) \equiv \bot; \cdots$] where the dots [$\cdots$] denote the remaining rules stated throughout the rest of this paper.

4.10 Elementary rules about equality

The elementary rules about equality read:

[Map rule

Trans:	$\langle \mathcal{A},\mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle$	$\mathcal{A}\equiv\mathcal{B}$	$\vdash$	$\mathcal{A}\equiv\mathcal{C}$	$\vdash$	$\mathcal{B}\equiv\mathcal{C}$	;
SubLambda:	$\langle x{\in}\mathcal{V};\mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle$			$\mathcal{A}\equiv\mathcal{B}$	$\vdash$	$\lambda x.\mathcal{A}\equiv\lambda x.\mathcal{B}$	;
SubApply:	$\langle \mathcal{A},\mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle$			$\mathcal{A}\equiv\mathcal{B}$	$\vdash$	$\mathcal{C}\,'\mathcal{A}\equiv\mathcal{C}\,'\mathcal{B}$	]*

$\left[\boxed{x \vdash y}\right]$ states that if there exists a proof of [x] then there exists a proof of [y]. [$x \vdash y$] is right associative so that [$x \vdash y \vdash z$] means [$x \vdash (y \vdash z)$]. [$x \vdash y$] has higher priority than [$\langle x{\in}y\rangle z$] so that e.g. [$\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle\mathcal{A} \equiv \mathcal{B} \vdash \mathcal{B} \equiv \mathcal{A}$] means [$\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle(\mathcal{A} \equiv \mathcal{B} \vdash \mathcal{B} \equiv \mathcal{A})$].

4.11 Elementary rules about application

The elementary rules about application read:

[Map rule

ApplyNil:	$\langle \mathcal{B}{\in}\mathcal{T}\rangle$	$\mathsf{N}\,'\mathcal{B}\equiv\mathsf{N}$	;
ApplyLambda:	$\langle x{\in}\mathcal{V};\mathcal{A},\mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle\mathcal{C}{\simeq}\langle\mathcal{A} \mid x{:=}\mathcal{B}\rangle \Vdash$	$(\lambda x.\mathcal{A})\,'\mathcal{B}\equiv\mathcal{C}$	;
ApplyBottom:	$\langle \mathcal{B}{\in}\mathcal{T}\rangle$	$\bot\,'\mathcal{B}\equiv\bot$	]*

If [x] is a computable term and [y] is a formula, then $\left[\boxed{x \Vdash y}\right]$ means that [y] is provable if [x] is true. In [$x \Vdash y$], [x] will be referred to as a *side condition*.

As mentioned in Section 2.6, $\left[\boxed{x \simeq y}\right]$ is true if [x] and [y] are Gödel-numbers of terms that are equal except for naming of bound variables (for some, suitable Gödel-numbering). As an example,

$$\underbrace{[\ \ulcorner\lambda y.\lambda z.y\urcorner \simeq \ulcorner\lambda v.\lambda w.v\urcorner\ ]}_{\text{side condition}}$$

is true where $\left[\boxed{\ulcorner\mathcal{A}\urcorner}\right]$ denotes the Gödel-number of [$\mathcal{A}$]. $\left[\boxed{\langle x \mid y{:=}z\rangle}\right]$ also operates on Gödel-numbers.

In addition to the elementary rules, λN has four advanced rules. They will be stated in Section 6.1, Section 7.1, Section 7.9, and Section 7.11.

In addition to the rules of λN, MT has eight more rules. They will be stated in Section 9.4 and Section 9.5. Until further, [Map] denotes the collection of rules of λN. From Chapter 9, [Map] will denote MT.

4.12 Lemmas of λN

The following is an example of a lemma of λN:

[Map lemma L4.12.1$_{55}$: $\lambda y.\lambda z.y \equiv \lambda v.\lambda w.v$]*

A lemma has exactly the same form as a rule except that "rule" is replaced by "lemma". In $\left[\boxed{x \text{ lemma } y}\right]$, [y] is a list of statements separated by semicolons which are claimed to be provable in the axiomatic system [x]. The lemma above claims that [$\lambda y.\lambda z.y \equiv \lambda v.\lambda w.v$] is provable in [Map] and introduces [L4.12.1] as shorthand for [$\lambda y.\lambda z.y \equiv \lambda v.\lambda w.v$]. In the lemma above, the list of statements that are claimed to be true merely contains one statement. Therefore, the list contains no semicolons.

$\left[\boxed{x_y{:}\,z}\right]$ has the same meaning as [$x{:}\,z$]. The subscript should be ignored by mechanical proof checkers. In lemmas like the one above, the subscript indicates the page number on which the proof of the lemma can be found.

4.13 Argumentations

The following is an example of an *argumentation*:

$$
\begin{array}{llll}
[\ \mathsf{L1}: & \text{ApplyLambda} \gg & (\lambda u.u)\,\text{'}(\lambda y.\lambda z.y) \equiv \lambda y.\lambda z.y & ; \\
\ \ \mathsf{L2}: & \text{ApplyLambda} \gg & (\lambda u.u)\,\text{'}(\lambda y.\lambda z.y) \equiv \lambda v.\lambda w.v & ; \\
\ \ \mathsf{L3}: & \text{Trans} \rhd \mathsf{L1} \rhd \mathsf{L2} \gg & \lambda y.\lambda z.y \equiv \lambda v.\lambda w.v &]^*
\end{array}
$$

The argumentation has three *proof lines*. Line [L1] uses [ApplyLambda] to conclude [$(\lambda u.u)\,\text{'}(\lambda y.\lambda z.y) \equiv \lambda y.\lambda z.y$]. Line [L2] uses the same rule to conclude [$(\lambda u.u)\,\text{'}(\lambda y.\lambda z.y) \equiv \lambda v.\lambda w.v$] (recall that ApplyLambda allows renaming of bound variables). Line [L3] uses [Trans] to conclude [$\lambda y.\lambda z.y \equiv \lambda v.\lambda w.v$] from Line [L1] and Line [L2].

The argumentation proves [$\lambda y.\lambda z.y \equiv \lambda v.\lambda w.v$] within [Map]. If an argumentation [$\mathcal{A}$] proves a statement [$\mathcal{S}$] within an axiomatic theory [$\mathcal{Z}$] then [$\mathcal{S}$] will be referred to as the *conclusion* of [$\mathcal{A}$] within [$\mathcal{Z}$] and will be denoted $\left[\boxed{\mathcal{A}/\mathcal{Z}}\right]$. [$\mathcal{A}/\mathcal{Z}$] equals the special constant $\left[\boxed{\text{Failure}}\right]$ if [$\mathcal{A}$] contains a flaw.

argumentation
proof line
conclusion

The operations $[\ \boxed{x \gg y}\ ]$, $[\ \boxed{x \rhd y}\ ]$, $[\ \boxed{x; y}\ ]$, and $[\ \boxed{x{:}\, y}\ ]$ allow to build up argumentations.

If $[\ u\ ]$ does not have one of the forms $[\ x \gg y\ ]$, $[\ x{\rhd}y\ ]$, $[\ x; y\ ]$, or $[\ x{:}\, y\ ]$, then $[\ u/\mathcal{Z}\ ]$ equals $[\ u\ ]$ if $[\ u\ ]$ belongs to the list $[\ \mathcal{Z}\ ]$ of rules and equals $[\ \text{Failure}\ ]$ otherwise.

$[\ x{:}\, y\ ]$ means the same as $[\ y\ ]$ except that it introduces $[\ x\ ]$ as shorthand for the conclusion of $[\ y\ ]$. Hence, locally in the proof, $[\ \mathsf{L1}\ ]$ is shorthand for $[\ (\lambda u.u)\,\text{'}(\lambda y.\lambda z.y) \equiv \lambda y.\lambda z.y\ ]$. $[\ \mathsf{L2}\ ]$ and $[\ \mathsf{L3}\ ]$ are defined similarly.

$[\ (x; y)/\mathcal{Z}\ ]$ equals $[\ y/\mathcal{Z}'\ ]$ where $[\ \mathcal{Z}'\ ]$ is $[\ \mathcal{Z}\ ]$ extended with the conclusion of $[\ x\ ]$. More precisely, $[\ (x; y)/\mathcal{Z}\ ]$ equals $[\ \text{Failure}\ ]$ if $[\ x/\mathcal{Z}\ ]$ equals $[\ \text{Failure}\ ]$. Otherwise, $[\ (x; y)/\mathcal{Z}\ ]$ equals $[\ y/(\mathcal{Z}; (x/\mathcal{Z}))\ ]$.

4.14 Instantiation

$[\ x \gg y\ ]$ states that $[\ y\ ]$ is an instance of $[\ x\ ]$. $[\ (x \gg y)/\mathcal{Z}\ ]$ equals $[\ y/\mathcal{Z}\ ]$ if $[\ y/\mathcal{Z}\ ]$ is an instance of $[\ x/\mathcal{Z}\ ]$ and satisfies all side conditions in $[\ x/\mathcal{Z}\ ]$. Otherwise, $[\ (x \gg y)/\mathcal{Z}\ ]$ equals $[\ \text{Failure}\ ]$.

The instantiation operator $[\ x \gg y\ ]$ has lower priority than $[\ x \vdash y\ ]$ and $[\ x \Vdash y\ ]$ so that e.g. $[\ x \Vdash y \gg z\ ]$ means $[\ (x \Vdash y) \gg z\ ]$. Here is an example:

$$[\ \mathcal{C} \simeq \langle \mathcal{A} \mid x{:=}\mathcal{B}\rangle \Vdash (\lambda x.\mathcal{A})\,\text{'}\,\mathcal{B} \equiv \mathcal{C} \gg (\lambda u.u)\,\text{'}(\lambda y.\lambda z.y) \equiv \lambda v.\lambda w.v\ ]$$

instantiates $[\ x\ ]$ to $[\ u\ ]$, $[\ \mathcal{A}\ ]$ to $[\ u\ ]$, $[\ \mathcal{B}\ ]$ to $[\ \lambda y.\lambda z.y\ ]$, and $[\ \mathcal{C}\ ]$ to $[\ \lambda v.\lambda w.v\ ]$. This instantiates the side condition to

$$[\ \lceil \lambda v.\lambda w.v \rceil \simeq \langle \lceil u \rceil \mid \lceil u \rceil{:=}\lceil \lambda y.\lambda z.y \rceil\rangle\ ].$$

The side condition evaluates to $[\ \mathsf{T}\ ]$ which makes the side condition disappear. Hence, the instantiation succeeds and yields

$$[\ (\lambda u.u)\,\text{'}(\lambda y.\lambda z.y) \equiv \lambda v.\lambda w.v\ ].$$

4.15 Application of rules to premises

$[\ x \rhd y\ ]$ applies a rule to a premise yielding a conclusion: Suppose $[\ x/\mathcal{Z}\ ]$ has form $[\ \mathcal{A} \vdash \mathcal{B}\ ]$ (possibly prepended with metaquantifiers and side conditions). Suppose $[\ y/\mathcal{Z}\ ]$ has form $[\ \mathcal{A}'\ ]$ (possibly prepended with metaquantifiers and side conditions). If $[\ \mathcal{A}\ ]$ and $[\ \mathcal{A}'\ ]$ can be unified, then $[\ (x{\rhd}y)/\mathcal{Z}\ ]$ equals the result of applying the unification to $[\ \mathcal{B}\ ]$ (prepended with suitable metaquantifiers and side conditions). $[\ x \rhd y\ ]$ never evaluates side conditions.

$[\ x \rhd y\ ]$ is left associative so that $[\ x \rhd y \rhd z\ ]$ means $[\ (x \rhd y) \rhd z\ ]$. $[\ x \rhd y\ ]$ has lower priority than $[\ x \vdash y\ ]$ and $[\ x \Vdash y\ ]$ so that e.g. $[\ x \vdash y \rhd z\ ]$ means $[\ (x \vdash y) \rhd z\ ]$. When $[\ x \rhd y\ ]$ is applied to two rules $[\ x\ ]$ and $[\ y\ ]$, the result is either a new rule or the value $[\ \text{Failure}\ ]$. Here is an example:

$$[\ \langle \mathcal{A}, \mathcal{B}, \mathcal{C} {\in} \mathcal{T}\rangle \mathcal{A} \equiv \mathcal{B} \vdash \mathcal{A} \equiv \mathcal{C} \vdash \mathcal{B} \equiv \mathcal{C} \rhd (\lambda z.z)\,\text{'}(\lambda x.\lambda y.x) \equiv \lambda x.\lambda y.x\ ]$$

unifies [$\mathcal{A}$] with [$(\lambda z.z)\,\text{'}(\lambda x.\lambda y.x)$] and [$\mathcal{B}$] with [$\lambda x.\lambda y.x$], which yields the rule

$$[\ \langle \mathcal{C}{\in}\mathcal{T}\rangle (\lambda z.z)\,\text{'}(\lambda x.\lambda y.x) \equiv \mathcal{C} \vdash \lambda x.\lambda y.x \equiv \mathcal{C}\ ].$$

Line [$\mathsf{L3}$] in Section 4.13 reads:

$$[\ \mathsf{L3}\text{: Trans} \rhd \mathsf{L1} \rhd \mathsf{L2} \gg \lambda y.\lambda z.y \equiv \lambda v.\lambda w.v\ ].$$

After replacing [Trans], [$\mathsf{L1}$], and [$\mathsf{L2}$] with their definitions and performing the unifications and instantiations, the conclusion of Line [$\mathsf{L3}$] reads

$$[\ \lambda y.\lambda z.y \equiv \lambda v.\lambda w.v\ ].$$

4.16 Proofs in $\lambda\mathsf{N}$

The following is an example of a *proof* in $\lambda\mathsf{N}$:

```
[ The Map proof of L4.12.1 reads
  L1:  ApplyLambda ≫       (λu.u) '(λy.λz.y) ≡ λy.λz.y     ;
  L2:  ApplyLambda ≫       (λu.u) '(λy.λz.y) ≡ λv.λw.v     ;
  L3:  Trans ▷ L1 ▷ L2 ≫   λy.λz.y ≡ λv.λw.v               ]*
```

$\left[\ \boxed{\text{A } \mathcal{Z} \text{ proof of } x \text{ reads } y}\ \right]$ states that [y] proves [x] within the axiomatic system [$\mathcal{Z}$]. In other words, [A $\mathcal{Z}$ proof of x reads y] states that [x] equals [$y/\mathcal{Z}$] and that [$y/\mathcal{Z}$] differs from [Failure].

4.17 Relative correctness

In $\lambda\mathsf{N}$, there are no axioms and inference rules that define the behavior of [$x \vdash y$], [$x \gg y$], and so on. Rather, [$x \vdash y$], [$x \gg y$], and so on are fundamental constructs of a meta-language for stating axioms and inference rules. These fundamental constructs can be implemented in any programming language yielding a program that can distinguish a mathematically correct text from an incorrect one.

A mathematical text will be said to be *absolutely correct* if every lemma in the text has a correct proof. Absolutely correct texts are rare because most texts leave out large, trivial proofs.

A mathematical text will be said to be *relatively correct* if every lemma in the text either has a correct proof or no proof. The present text aims at relative correctness. In a relatively correct text, any lemma can be used in any proof, even lemmas for which no proof is given.

proof
absolutely correct
relatively correct

Chapter 5

Proof techniques of $\lambda\mathsf{N}$

5.1 Equality

The following *meta-lemmas* state that [$x \equiv y$] is an equivalence relation:

[Map lemma

Reflexivity$_{57}$:	$\langle \mathcal{A} {\in} \mathcal{T} \rangle$	$\mathcal{A} \equiv \mathcal{A}$					;
Commutativity$_{58}$:	$\langle \mathcal{A}, \mathcal{B} {\in} \mathcal{T} \rangle$	$\mathcal{A} \equiv \mathcal{B}$	$\vdash$	$\mathcal{B} \equiv \mathcal{A}$			;
Transitivity$_{58}$:	$\langle \mathcal{A}, \mathcal{B}, \mathcal{C} {\in} \mathcal{T} \rangle$	$\mathcal{A} \equiv \mathcal{B}$	$\vdash$	$\mathcal{B} \equiv \mathcal{C}$	$\vdash$	$\mathcal{A} \equiv \mathcal{C}$	]*

[Reflexivity] is a meta-lemma rather than a *concrete lemma* because it contains a meta-quantifier. In general, lemmas that contain the constructs [$\langle x {\in} y \rangle z$], [$x \vdash y$], or [$x \Vdash y$] are meta-lemmas. The *meta-proof* of [Reflexivity] reads:

[The Map proof of Reflexivity reads

L1:	IfNil $\gg$	$\mathrm{if}(\mathsf{N}, \mathcal{A}, \mathsf{N}) \equiv \mathcal{A}$	;
L2:	IfNil $\gg$	$\mathrm{if}(\mathsf{N}, \mathcal{A}, \mathsf{N}) \equiv \mathcal{A}$	;
L3:	Trans $\rhd$ L1 $\rhd$ L2 $\gg$	$\mathcal{A} \equiv \mathcal{A}$	]*

The proof has two identical lines, so it may be stated one line shorter.

There are two different ways to treat meta-proofs like the one above. The first approach is to regard the meta-proof as a proof procedure which takes a term [$\mathcal{A}$] as input and produces a proof of [$\mathcal{A} \equiv \mathcal{A}$] as output. With this approach, a mechanical proof checker will have to execute the proof procedure each time [Reflexivity] is used in another proof. The second approach is to program the proof checker such that it checks the meta-proof once and for all. For some of the meta-proofs stated later on, only the first of the two approaches is possible.

The next proof contains a [$\boxed{\text{Premise}}$].

meta-lemma
concrete lemma
meta-proof

[The Map proof of Commutativity reads
L1: Premise $\gg$ $\mathcal{A} \equiv \mathcal{B}$;
L2: Reflexivity $\gg$ $\mathcal{A} \equiv \mathcal{A}$;
L3: Trans $\rhd$ L1 $\rhd$ L2 $\gg$ $\mathcal{B} \equiv \mathcal{A}$]*

[Commutativity] says that if [$\mathcal{A}$] and [$\mathcal{B}$] are terms and if there exists a proof of [$\mathcal{A} \equiv \mathcal{B}$], then there also exists a proof of [$\mathcal{B} \equiv \mathcal{A}$]. The meta-proof above indicates how to construct such a proof of [$\mathcal{B} \equiv \mathcal{A}$] given a proof of [$\mathcal{A} \equiv \mathcal{B}$]: Replace [$\mathcal{A}$] and [$\mathcal{B}$] by the actual terms, and replace line 1 of the meta-proof by the given proof of [$\mathcal{A} \equiv \mathcal{B}$].

There are two different ways to treat meta-proofs with premises. The first approach is to regard the meta-proof as a proof procedure which takes a proof of [$\mathcal{A} \equiv \mathcal{B}$] as input and produces a proof of [$\mathcal{B} \equiv \mathcal{A}$] as output. With this a approach, [Premise] declares a parameter of the proof procedure. The second approach is to program the proof checker such that it checks all of the proof except the premises. With this approach, a mechanical proof checker merely has to check the proof of [Commutativity] once.

[The Map proof of Transitivity reads
L1: Premise $\gg$ $\mathcal{A} \equiv \mathcal{B}$;
L2: Premise $\gg$ $\mathcal{B} \equiv \mathcal{C}$;
L3: Commutativity $\rhd$ L1 $\gg$ $\mathcal{B} \equiv \mathcal{A}$;
L4: Trans $\rhd$ L3 $\rhd$ L2 $\gg$ $\mathcal{A} \equiv \mathcal{C}$]*

5.2 Side conditions

The following lemma has a side condition:

[Map lemma Rename$_{58}$: $\langle \mathcal{A}, \mathcal{B} \in \mathcal{T} \rangle \mathcal{A} \simeq \mathcal{B} \Vdash \mathcal{A} \equiv \mathcal{B}$]*

[The Map proof of Rename reads
L1: ApplyLambda $\gg$ $(\lambda x.x)\,'\,\mathcal{A} \equiv \mathcal{A}$;
L2: ApplyLambda $\gg$ $(\lambda x.x)\,'\,\mathcal{A} \equiv \mathcal{B}$;
L3: Trans $\rhd$ L1 $\rhd$ L2 $\gg$ $\mathcal{A} \equiv \mathcal{B}$]*

Mechanical proof checkers are unable to check proofs of lemmas that contain side conditions. Rather, one is forced to regard the proof above as a proof procedure which is expected to generate a proof of [$\mathcal{A} \equiv \mathcal{B}$] whenever [$\mathcal{A} \simeq \mathcal{B}$]. Whenever a mechanical proof checker meets a proof line of form [Rename $\gg$ $\mathcal{X} \equiv \mathcal{Y}$] it should first evaluate [$\mathcal{X} \simeq \mathcal{Y}$] and reject the proof line if [$\mathcal{X} \simeq \mathcal{Y}$] fails. After that it should replace [$\mathcal{A}$] and [$\mathcal{B}$] in the proof of [Rename] by [$\mathcal{X}$] and [$\mathcal{Y}$], respectively, and insert the result in place of [Rename $\gg$ $\mathcal{X} \equiv \mathcal{Y}$]. Finally, the proof checker should check these new proof lines in the given context.

5.3 Repetition

The following peculiar metatheorem turns out to be handy:

[Map lemma Repetition$_{59}$: $\mathcal{A} \equiv \mathcal{B} \vdash \mathcal{A} \equiv \mathcal{B}$]*

[The Map proof of Repetition reads

L1:	Premise $\gg$	$\mathcal{A} \equiv \mathcal{B}$	;
L2:	Commutativity $\rhd$ L1 $\gg$	$\mathcal{B} \equiv \mathcal{A}$	;
L3:	Commutativity $\rhd$ L2 $\gg$	$\mathcal{A} \equiv \mathcal{B}$	]*

The metatheorem allows to repeat a previous proof line, which may enhance the readability of a proof.

5.4 Algebraic blocks

Proofs of form [$\mathcal{A}_0 \equiv \mathcal{A}_1 \vdash \mathcal{A}_1 \equiv \mathcal{A}_2 \vdash \cdots \vdash \mathcal{A}_{n-1} \equiv \mathcal{A}_n \vdash \mathcal{A}_0 \equiv \mathcal{A}_n$] are very common. For that reason, the construct

[a:	Block $\gg$	Begin	;
b_0:	Algebra $\gg$	$\mathcal{A}_0$	;
b_1:	$\mathcal{R}_1 \gg$	$\mathcal{A}_1$	;
$\vdots$	$\vdots$	$\vdots$	;
b_n:	$\mathcal{R}_n \gg$	$\mathcal{A}_n$	;
c:	Block $\gg$	End	]

is introduced as shorthand for

[b_0:	Reflexivity $\gg$	$\mathcal{A}_0 \equiv \mathcal{A}_0$	;
b_1:	Transitivity $\rhd b_0 \rhd \mathcal{R}_1 \gg$	$\mathcal{A}_0 \equiv \mathcal{A}_1$	;
$\vdots$	$\vdots$	$\vdots$	;
a: c: b_n:	Transitivity $\rhd b_{n-1} \rhd \mathcal{R}_n \gg$	$\mathcal{A}_0 \equiv \mathcal{A}_n$	]

For [$i = 1, \ldots, n$], unification requires [$\mathcal{R}_i$] to conclude [$\mathcal{A}_{i-1} \equiv \mathcal{A}_i$].

Proof lines inside an algebraic block contain terms whereas lines outside algebraic blocks contain formulas. Algebraic blocks cannot contain other blocks.

5.5 Substitution

Let $\left[\boxed{\mathrm{Rep}(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})} \right]$ be defined such that [Rep($\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$)] is true iff [$\mathcal{D}$] arises from [$\mathcal{C}$] by replacing [$\mathcal{A}$] by [$\mathcal{B}$] or [$\mathcal{B}$] by [$\mathcal{A}$] zero, one or more times. As an example,

$$[\ \mathrm{Rep}(\lceil 2 \cdot 3 \rceil, \lceil 6 \rceil, \lceil 2 \cdot 3 + 9 \rceil, \lceil 6 + 9 \rceil)\ ]$$

is true. The following lemma allows substitution of equals:

[Map lemma

Replace:	$\mathrm{Rep}(\mathcal{A}, \dot{\mathcal{A}}, \mathcal{B}, \dot{\mathcal{B}}) \Vdash \mathcal{A} \equiv \dot{\mathcal{A}} \vdash \mathcal{B} \equiv \dot{\mathcal{B}}$	;
Replace'$_{123}$:	$\mathrm{Rep}(\mathcal{A}, \dot{\mathcal{A}}, \mathcal{B}, \dot{\mathcal{B}}) \wedge \mathrm{Rep}(\mathcal{A}, \dot{\mathcal{A}}, \mathcal{C}, \dot{\mathcal{C}}) \Vdash$ $\mathcal{A} \equiv \dot{\mathcal{A}} \vdash \mathcal{B} \equiv \mathcal{C} \vdash \dot{\mathcal{B}} \equiv \dot{\mathcal{C}}$	]*

It is easy to prove [Replace'] on basis of [Replace].

[Replace] follows by structural induction from the lemmas below. More precisely, [Replace] is a meta-lemma whose proof procedure constructs a proof of [$\mathcal{B} \equiv \mathcal{C}$] by recursion on the structure of [$\mathcal{B}$] and [$\mathcal{C}$].

This paper does not introduce a notation for proofs that require structural inductions. Therefore, [Replace] has no formal proof in this paper, c.f. the notes on relative correctness in Section 4.17. The auxiliary lemmas for the structural induction read:

[Map lemma

L5.5.1$_{123}$:	$\langle a, \underline{a} \in \mathcal{T} \rangle a \equiv \underline{a} \vdash$	$\lambda x.a \equiv \lambda x.\underline{a}$	;
L5.5.2$_{123}$:	$\langle a, \underline{a}, b, \underline{b} \in \mathcal{T} \rangle a \equiv \underline{a} \vdash b \equiv \underline{b} \vdash$	$a\,'\,b \equiv \underline{a}\,'\,\underline{b}$	;
L5.5.3$_{124}$:	$\langle a, \underline{a}, b, \underline{b}, c, \underline{c} \in \mathcal{T} \rangle a \equiv \underline{a} \vdash b \equiv \underline{b} \vdash c \equiv \underline{c} \vdash$	$\mathrm{if}(a, b, c) \equiv \mathrm{if}(\underline{a}, \underline{b}, \underline{c})$	]*

5.6 Parameterized lemmas

When a meta-lemma is used in a proof, one may use the procedure of unification to deduce what the meta-variables in premises and conclusions stand for, but unification cannot deduce what meta-variables in side conditions stand for. Hence, if a meta-variable merely occurs in a side condition, then that variable has to be instantiated explicitly each time the lemma is used in a proof. Such variables will be referred to a *parameters* of the lemma. The following is an example of a parameterized lemma:

[Map lemma L5.6.1$(x, \mathcal{C})_{60}$: $\langle \mathcal{A}, \dot{\mathcal{A}}, \mathcal{B}, \dot{\mathcal{B}} \in \mathcal{T} \rangle \dot{\mathcal{A}} \simeq \langle \mathcal{A} \mid x{:=}\mathcal{C} \rangle \wedge \dot{\mathcal{B}} \simeq \langle \mathcal{B} \mid x{:=}\mathcal{C} \rangle \Vdash \mathcal{A} \equiv \mathcal{B} \vdash \dot{\mathcal{A}} \equiv \dot{\mathcal{B}}$]*

[The Map proof of L5.6.1$(x, \mathcal{C})$ reads

L1:	Premise ≫	$\mathcal{A} \equiv \mathcal{B}$	;
L2:	Block ≫	Begin	;
L3:	Algebra ≫	$\dot{\mathcal{A}}$	;
L4:	Commutativity ▷ ApplyLambda ≫	$(\lambda x.\mathcal{A})\,'\,\mathcal{C}$	;
L5:	Replace ▷ L1 ≫	$(\lambda x.\mathcal{B})\,'\,\mathcal{C}$	;
L6:	ApplyLambda ≫	$\dot{\mathcal{B}}$	;
L7:	Block ≫	End	]*

The proof above illustrates three things:

First, the proof is parameterized by [x] and [$\mathcal{C}$]. Each time [L5.6.1$(x, \mathcal{C})$] is used in a proof, a mechanical proof checker needs to know what [x] and [$\mathcal{C}$] stand for in order to expand the proof above.

Second, Line [L4] shows how it is possible to combine [Commutativity] with [ApplyLambda] to show $\left[\, \dot{\mathcal{A}} \equiv (\lambda x.\mathcal{A})\,'\,\mathcal{C} \,\right]$. In Line [L4], unification forces [ApplyLambda] to verify $\left[\, (\lambda x.\mathcal{A})\,'\,\mathcal{C} \equiv \dot{\mathcal{A}} \,\right]$.

parameters

Third, the proof illustrates that a proof can stop inside an algebraic block. When expanding the algebraic block as specified in Section 5.4, Line [L7] disappears and Line [L6] becomes $[\, \dot{\mathcal{A}} \equiv \dot{\mathcal{B}} \,]$, so $[\, \dot{\mathcal{A}} \equiv \dot{\mathcal{B}} \,]$ becomes the conclusion of the proof as required.

Now let

$$\left[\, \boxed{\langle \mathcal{A} \mid z_1 := \mathcal{B}_1 \mid \cdots \mid z_n := \mathcal{B}_n \rangle} \,\right]$$

be shorthand for

$$[\, \langle \cdots \langle \mathcal{A} \mid z_1{:=}\mathcal{B}_1 \rangle \cdots \mid z_n{:=}\mathcal{B}_n \rangle \,].$$

The following lemma illustrates a use of [L5.6.1$(x,\mathcal{C})$]:

[Map lemma L5.6.2$(x_1,\mathcal{C}_1,x_2,\mathcal{C}_2,x_3,\mathcal{C}_3)_{61}$: $\dot{\mathcal{A}} \simeq \langle \mathcal{A} \mid x_1 := \mathcal{C}_1 \mid x_2 := \mathcal{C}_2 \mid x_3 := \mathcal{C}_3 \rangle \wedge \dot{\mathcal{B}} \simeq \langle \mathcal{B} \mid x_1 := \mathcal{C}_1 \mid x_2 := \mathcal{C}_2 \mid x_3 := \mathcal{C}_3 \rangle \Vdash \mathcal{A} \equiv \mathcal{B} \vdash \dot{\mathcal{A}} \equiv \dot{\mathcal{B}}$]*

[The Map proof of L5.6.2$(x_1,\mathcal{C}_1,x_2,\mathcal{C}_2,x_3,\mathcal{C}_3)$ reads

L1:	Premise $\gg$	$\mathcal{A} \equiv \mathcal{B}$	;
L2:	L5.6.1$(x_1,\mathcal{C}_1) \rhd$ L1 $\gg$	$\langle \mathcal{A} \vert x_1{:=}\mathcal{C}_1 \rangle \equiv \langle \mathcal{B} \vert x_1{:=}\mathcal{C}_1 \rangle$	;
L3:	L5.6.1$(x_2,\mathcal{C}_2) \rhd$ L2 $\gg$	$\langle \mathcal{A} \vert x_1{:=}\mathcal{C}_1 \vert x_2{:=}\mathcal{C}_2 \rangle \equiv \langle \mathcal{B} \vert x_1{:=}\mathcal{C}_1 \vert x_2{:=}\mathcal{C}_2 \rangle$	;
L4:	L5.6.1$(x_3,\mathcal{C}_3) \rhd$ L3 $\gg$	$\dot{\mathcal{A}} \equiv \dot{\mathcal{B}}$	]*

5.7 Instantiation

If [$\mathcal{A} \equiv \mathcal{B}$] is a formula, if [$z_1,\ldots,z_n$] are variables, if [$\mathcal{C}_1,\ldots \mathcal{C}_n$] are terms, and if

$$[\, \dot{\mathcal{A}} \simeq \langle \mathcal{A} \mid z_1 := \mathcal{C}_1 \mid \cdots \mid z_n := \mathcal{C}_n \rangle \,] \text{ and}$$

$$[\, \dot{\mathcal{B}} \simeq \langle \mathcal{B} \mid z_1 := \mathcal{C}_1 \mid \cdots \mid z_n := \mathcal{C}_n \rangle \,]$$

then $[\, \dot{\mathcal{A}} \equiv \dot{\mathcal{B}} \,]$ will be said to be an *instance* of [$\mathcal{A} \equiv \mathcal{B}$]. Let

$$\left[\, \boxed{\text{Instance}(\mathcal{A},\mathcal{B},\dot{\mathcal{A}},\dot{\mathcal{B}})} \,\right]$$

denote that $[\, \dot{\mathcal{A}} \equiv \dot{\mathcal{B}} \,]$ is an instance of [$\mathcal{A} \equiv \mathcal{B}$].

[Map lemma Instantiation: Instance$(\mathcal{A},\mathcal{B},\dot{\mathcal{A}},\dot{\mathcal{B}}) \Vdash \mathcal{A} \equiv \mathcal{B} \vdash \dot{\mathcal{A}} \equiv \dot{\mathcal{B}}$]*

A proof procedure for [Instantiation] must extract [$z_1,\mathcal{C}_1,\ldots,z_n,\mathcal{C}_n$] from [$\mathcal{A}$], [$\mathcal{B}$], $[\, \dot{\mathcal{A}} \,]$, and $[\, \dot{\mathcal{B}} \,]$ and then apply [L5.6.1] repeatedly. The proof procedure will not be stated in this paper.

instance

5.8 Reduction

Section 2.8 introduced a reduction system with four reduction rules. In this section we consider a reduction system with two more rules:

$$\begin{array}{llll}
[\, \mathsf{N}\,\text{'}\,\mathcal{B} & \overset{+}{\to} & \mathsf{N} &] \\
[\, (\lambda x.\mathcal{A})\,\text{'}\,\mathcal{B} & \overset{+}{\to} & \langle \mathcal{A} \mid x{:=}\mathcal{B} \rangle &] \\
[\, \bot\,\text{'}\,\mathcal{B} & \overset{+}{\to} & \bot &] \\
[\, \text{if}(\mathsf{N},\mathcal{B},\mathcal{C}) & \overset{+}{\to} & \mathcal{B} &] \\
[\, \text{if}(\lambda x.\mathcal{A},\mathcal{B},\mathcal{C}) & \overset{+}{\to} & \mathcal{C} &] \\
[\, \text{if}(\bot,\mathcal{B},\mathcal{C}) & \overset{+}{\to} & \bot &]
\end{array}$$

Section 2.8 considered normal order reduction. In contrast, this section considers arbitrary reduction.

Let $\left[\boxed{\text{Reduction}(\mathcal{A},\mathcal{B})} \right]$ be true iff [$\mathcal{A}$] and [$\mathcal{B}$] reduce to terms that are identical except for naming of bound variables in the reduction system above.

[Map lemma Reduction: Reduction$(\mathcal{A},\mathcal{B}) \Vdash \mathcal{A} \equiv \mathcal{B}$]*

[Reduction] Follows from [Reflexivity], [Commutativity], [Transitivity] and [Replace]. No formal proof procedure will be stated.

Note that the side condition in Reduction is not decidable. If decidability is required, one has e.g. to put an upper bound on the number of reductions needed to reach a common term. Lack of decidability may cause a mechanical proof checker to loop indefinitely when applied to an incorrect proof.

5.9 Pairs

As an example of use of Reduction, recall the following definitions from Section 3.14:

$$\left[\boxed{x :: y} \doteq \lambda z.\text{if}(z,x,y) \right]^*,$$

$$\left[\boxed{\mathbf{head}(x)} \doteq x\,\text{'}\,\mathsf{N} \right]^*, \text{ and}$$

$$\left[\boxed{\mathbf{tail}(x)} \doteq x\,\text{'}\,\lambda x.\mathsf{N} \right]^*.$$

In [$\mathcal{A} :: \mathcal{B}$] it is understood that [z] in the definition of [::] denotes some variable that is not free in [$\mathcal{A}$] and [$\mathcal{B}$]. The main lemmas about pairs read:

[Map lemma

$$\begin{array}{llll}
\text{Head}_{63}: & \mathbf{head}(x :: y) & \equiv & x \quad ; \\
\text{Tail}_{63}: & \mathbf{tail}(x :: y) & \equiv & y \quad]^*
\end{array}$$

Reduction of [$\mathbf{head}(x :: y)$] proceeds thus:

$$
\begin{array}{lll}
[& \mathbf{head}(x :: y) & \simeq \\
 & (\lambda z.\mathrm{if}(z,x,y))\text{'}\,\mathsf{N} & \xrightarrow{+} \\
 & \mathrm{if}(\mathsf{N},x,y) & \xrightarrow{+} \\
 & x &]
\end{array}
$$

The first step above is not a reduction step; it is just a repetition of the same term written in two ways. The term [x] is irreducible, so the reduction sequence for [x] merely contains the term [x] itself. Hence, [$\mathbf{head}(x :: y)$] and [x] reduce to the common term [x]. This allows to prove [Head] by reduction:

[The Map proof of Head reads
L1: Reduction $\gg$ $\mathbf{head}(x :: y) \equiv x$]*

The treatment of [Tail] is similar:

[The Map proof of Tail reads
L1: Reduction $\gg$ $\mathbf{tail}(x :: y) \equiv y$]*

Note that [x] and [y] in [Head] are concrete variables. Applications of [Head] typically have the form [Instantiation $\rhd$ Head] where Instantiation allows to replace [x] and [y] with arbitrary terms.

The definition

$$[\, x :: y \doteq \lambda z.\mathrm{if}(z,x,y) \,]^*$$

means that [$\mathcal{A} :: \mathcal{B}$] is shorthand for [$\lambda\mathcal{C}.\mathrm{if}(\mathcal{C},\mathcal{A},\mathcal{B})$] where [$\mathcal{C}$] is chosen in some, deterministic way such that [$\mathcal{C}$] is not free in [$\mathcal{A}$] and [$\mathcal{B}$]. A general proof of [$\mathbf{head}(\mathcal{A} :: \mathcal{B}) \equiv \mathcal{A}$] cannot assume that [$\mathbf{head}(\mathcal{A} :: \mathcal{B})$] is shorthand for [$\lambda z.\mathrm{if}(z,\mathcal{A},\mathcal{B})$].

5.10 The fixed point operator [Y]

Recall the following definition from Section 3.8:

$$\left[\boxed{\mathsf{Y}} \doteq \lambda f.(\lambda x.f\text{'}(x\text{'}x))\text{'}(\lambda x.f\text{'}(x\text{'}x))\right].$$

[Map lemma FixedPoint$_{63}$: $\mathsf{Y}\text{'}f \equiv f\text{'}(\mathsf{Y}\text{'}f)$]*

[The Map proof of FixedPoint reads
L1: Reduction $\gg$ $\mathsf{Y}\text{'}f \equiv f\text{'}(\mathsf{Y}\text{'}f)$]*

A reduction sequence of [$\mathsf{Y}\text{'}f$] reads:

$$
\begin{array}{lll}
[& \mathsf{Y}\text{'}f & \simeq \\
 & (\lambda f.(\lambda x.f\text{'}(x\text{'}x))\text{'}(\lambda x.f\text{'}(x\text{'}x)))\text{'}f & \xrightarrow{+} \\
 & (\lambda x.f\text{'}(x\text{'}x))\text{'}(\lambda x.f\text{'}(x\text{'}x)) & \xrightarrow{+} \\
 & f\text{'}((\lambda x.f\text{'}(x\text{'}x))\text{'}(\lambda x.f\text{'}(x\text{'}x))) &]
\end{array}
$$

A reduction sequence of [$f\,\text{'}(\mathsf{Y}\,\text{'}\,f)$] reads:

$$\begin{array}{lll} [& f\,\text{'}(\mathsf{Y}\,\text{'}\,f) & \simeq \\ & f\,\text{'}((\lambda f.(\lambda x.f\,\text{'}(x\,\text{'}\,x))\,\text{'}(\lambda x.f\,\text{'}(x\,\text{'}\,x)))\,\text{'}\,f) & \overset{+}{\rightarrow} \\ & f\,\text{'}((\lambda x.f\,\text{'}(x\,\text{'}\,x))\,\text{'}(\lambda x.f\,\text{'}(x\,\text{'}\,x))) &] \end{array}$$

This verifies that [$\mathsf{Y}\,\text{'}\,f$] and [$f\,\text{'}(\mathsf{Y}\,\text{'}\,f)$] reduce to a common term.

5.11 Terms that represent formulas

Whenever a term [$\mathcal{A}$] occurs in a position where a formula is expected, [$\mathcal{A}$] is shorthand for [$\mathcal{A} \equiv \mathsf{T}$]. As an example, if a lemma reads [$2+3=5$], then that lemma states that [$(2+3=5) \equiv \mathsf{T}$]:

[Map lemma L5.11.1$_{64}$: $2+3=5$]*

[The Map proof of L5.11.1 reads
L1: Reduction ≫ $2+3=5$]*

The lemma says [$(2+3=5) \equiv \mathsf{T}$] and Line [L1] says the same.

Statements about terms being true are frequent, so it is convenient to have a simple convention like the one above for stating that a term is true.

As mentioned in Section 3.5 it is convenient to let the ur-element represent truth and let functions represent falsehood. One benefit of this choice is that truth merely has one representation so that the convention above makes sense. If truth had more than one representation, one would need a more complicated convention than the one above. Another benefit of letting the ur-element represent truth was stated in Section 3.13.

Chapter 6

Quartum Non Datur (QND)

6.1 Presentation of QND

As mentioned in Section 2.15, the *Quartum Non Datur* (*QND*) inference of $\lambda\mathsf{N}$ says that every map [$\mathcal{C}$] satisfies

[$\mathcal{C} \equiv \mathsf{N}$] or [$\mathcal{C} \equiv \lambda y.\mathcal{C}\,'\,y$] or [$\mathcal{C} \equiv \bot$]

there is no fourth possibility. Another formulation reads: Every map [$\mathcal{C}$] satisfies

[$\mathcal{C} \equiv \mathsf{N}$] or [$\mathcal{C} \equiv \Lambda\mathcal{C}$] or [$\mathcal{C} \equiv \bot$]

where

[$\Lambda\mathcal{C} \doteq (\lambda x.\lambda y.x\,'\,y)\,'\,\mathcal{C}$].

The latter formulation is the most convenient one to formalize because it allows [$\mathcal{C}$] to be a term with arbitrary free variables:

[Map rule QND: $\langle\mathcal{A},\mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle \mathcal{A}\,'\,\mathsf{N} \equiv \mathcal{B}\,'\,\mathsf{N} \vdash \mathcal{A}\,'\,\Lambda\mathcal{C} \equiv \mathcal{B}\,'\,\Lambda\mathcal{C} \vdash \mathcal{A}\,'\,\bot \equiv \mathcal{B}\,'\,\bot \vdash \mathcal{A}\,'\,\mathcal{C} \equiv \mathcal{B}\,'\,\mathcal{C}$]*

Here is an example of use:

[Map lemma L6.1.1$_{66}$: $\neg\neg\neg x \equiv \neg x$]*

Quartum Non Datur
QND

[The Map proof of L6.1.1 reads

L1:	Reduction $\gg$	$(\lambda x.\neg\neg\neg x)\,{}'\,\mathsf{N} \equiv (\lambda x.\neg x)\,{}'\,\mathsf{N}$	;
L2:	Reduction $\gg$	$(\lambda x.\neg\neg\neg x)\,{}'\,\Lambda x \equiv (\lambda x.\neg x)\,{}'\,\Lambda x$	;
L3:	Reduction $\gg$	$(\lambda x.\neg\neg\neg x)\,{}'\,\bot \equiv (\lambda x.\neg x)\,{}'\,\bot$	;
L4:	QND $\rhd$ L1 $\rhd$ L2 $\rhd$ L3 $\gg$	$(\lambda x.\neg\neg\neg x)\,{}'\,x \equiv (\lambda x.\neg x)\,{}'\,x$	;
L5:	Transitivity $\rhd$ L4 $\rhd$ ApplyLambda $\gg$	$(\lambda x.\neg\neg\neg x)\,{}'\,x \equiv \neg x$	;
L6:	Trans $\rhd$ ApplyLambda $\rhd$ L5 $\gg$	$\neg\neg\neg x\,{}'\,x \equiv \neg x$	]*

6.2 QND applied to a variable

The following version of [QND] turns out to be handy:

[Map lemma Q$(x)_{124}$: $\langle \mathcal{A},\mathcal{B}\in\mathcal{T}\rangle(\lambda x.\mathcal{A})\,{}'\,\mathsf{N} \equiv (\lambda x.\mathcal{B})\,{}'\,\mathsf{N} \vdash (\lambda x.\mathcal{A})\,{}'\,\Lambda x \equiv (\lambda x.\mathcal{B})\,{}'\,\Lambda x \vdash (\lambda x.\mathcal{A})\,{}'\,\bot \equiv (\lambda x.\mathcal{B})\,{}'\,\bot \vdash \mathcal{A} \equiv \mathcal{B}$]*

The proof resembles that of [L6.1.1]. [Q(x)] allows to state the proof of [L6.1.1] a bit shorter:

[The Map proof number 2 of L6.1.1 reads

L1:	Reduction $\gg$	$(\lambda x.\neg\neg\neg x)\,{}'\,\mathsf{N} \equiv (\lambda x.\neg x)\,{}'\,\mathsf{N}$	;
L2:	Reduction $\gg$	$(\lambda x.\neg\neg\neg x)\,{}'\,\Lambda x \equiv (\lambda x.\neg x)\,{}'\,\Lambda x$	;
L3:	Reduction $\gg$	$(\lambda x.\neg\neg\neg x)\,{}'\,\bot \equiv (\lambda x.\neg x)\,{}'\,\bot$	;
L4:	Q(x) $\rhd$ L1 $\rhd$ L2 $\rhd$ L3 $\gg$	$\neg\neg\neg x \equiv \neg x$	]*

Now define

$$\left[\,\boxed{\bar{\mathcal{R}}} \doteq \text{Reduction}\,\right]^{*}.$$

This definition allows to give a rather short proof of [L6.1.1]:

[The Map proof number 3 of L6.1.1 reads

L1: Q(x) $\rhd \bar{\mathcal{R}} \rhd \bar{\mathcal{R}} \rhd \bar{\mathcal{R}} \gg$ $\neg\neg\neg x \equiv \neg x$]*

The parameter [x] was included in [Q(x)] in order to allow terse proofs like the one-liner above. A mechanical proof checker is able to construct the three premises of [Q(x)] from the conclusion and this parameter. When the premises are given explicitly like in the four line proof above, the parameter serves no purpose.

6.3 Two levels of QND

The following lemma illustrates that [QND] can be used in two levels:

[Map lemma L6.3.1$_{67}$: $x \wedge y \equiv y \wedge x$]*

[The Map proof of L6.3.1 reads

$$\begin{array}{rlll}
a: & \bar{\mathcal{R}} \gg & (\lambda y.(\lambda x.x\wedge y)\text{'}\ \mathsf{N})\text{'}\ \mathsf{N} & \equiv (\lambda y.(\lambda x.y\wedge x)\text{'}\ \mathsf{N})\text{'}\ \mathsf{N} \quad ; \\
b: & \bar{\mathcal{R}} \gg & (\lambda y.(\lambda x.x\wedge y)\text{'}\ \mathsf{N})\text{'}\Lambda y & \equiv (\lambda y.(\lambda x.y\wedge x)\text{'}\ \mathsf{N})\text{'}\Lambda y \quad ; \\
c: & \bar{\mathcal{R}} \gg & (\lambda y.(\lambda x.x\wedge y)\text{'}\ \mathsf{N})\text{'}\ \bot & \equiv (\lambda y.(\lambda x.y\wedge x)\text{'}\ \mathsf{N})\text{'}\ \bot \quad ; \\
d: & \bar{\mathcal{R}} \gg & (\lambda y.(\lambda x.x\wedge y)\text{'}\Lambda x)\text{'}\ \mathsf{N} & \equiv (\lambda y.(\lambda x.y\wedge x)\text{'}\Lambda x)\text{'}\ \mathsf{N} \quad ; \\
e: & \bar{\mathcal{R}} \gg & (\lambda y.(\lambda x.x\wedge y)\text{'}\Lambda x)\text{'}\Lambda y & \equiv (\lambda y.(\lambda x.y\wedge x)\text{'}\Lambda x)\text{'}\Lambda y \quad ; \\
f: & \bar{\mathcal{R}} \gg & (\lambda y.(\lambda x.x\wedge y)\text{'}\Lambda x)\text{'}\ \bot & \equiv (\lambda y.(\lambda x.y\wedge x)\text{'}\Lambda x)\text{'}\ \bot \quad ; \\
g: & \bar{\mathcal{R}} \gg & (\lambda y.(\lambda x.x\wedge y)\text{'}\ \bot)\text{'}\ \mathsf{N} & \equiv (\lambda y.(\lambda x.y\wedge x)\text{'}\ \bot)\text{'}\ \mathsf{N} \quad ; \\
h: & \bar{\mathcal{R}} \gg & (\lambda y.(\lambda x.x\wedge y)\text{'}\ \bot)\text{'}\Lambda y & \equiv (\lambda y.(\lambda x.y\wedge x)\text{'}\ \bot)\text{'}\Lambda y \quad ; \\
i: & \bar{\mathcal{R}} \gg & (\lambda y.(\lambda x.x\wedge y)\text{'}\ \bot)\text{'}\ \bot & \equiv (\lambda y.(\lambda x.y\wedge x)\text{'}\ \bot)\text{'}\ \bot \quad ; \\
j: & \mathsf{Q}(y)\rhd a\rhd b\rhd c\gg & (\lambda x.x\wedge y)\text{'}\ \mathsf{N} & \equiv (\lambda x.y\wedge x)\text{'}\ \mathsf{N} \quad ; \\
k: & \mathsf{Q}(y)\rhd d\rhd e\rhd f\gg & (\lambda x.x\wedge y)\text{'}\Lambda x & \equiv (\lambda x.y\wedge x)\text{'}\Lambda x \quad ; \\
l: & \mathsf{Q}(y)\rhd g\rhd h\rhd i\gg & (\lambda x.x\wedge y)\text{'}\ \bot & \equiv (\lambda x.y\wedge x)\text{'}\ \bot \quad ; \\
m: & \mathsf{Q}(x)\rhd j\rhd k\rhd l\gg & x\wedge y & \equiv y\wedge x \quad]^*
\end{array}$$

A shorter version reads:

[The Map proof number 2 of L6.3.1 reads

$$\begin{array}{rllcll}
\mathsf{L1}: & \mathsf{Q}(y) \rhd \bar{\mathcal{R}} \rhd \bar{\mathcal{R}} \rhd \bar{\mathcal{R}} \gg & (\lambda x.x \wedge y)\text{'}\,\mathsf{N} & \equiv & (\lambda x.y \wedge x)\text{'}\,\mathsf{N} & ; \\
\mathsf{L2}: & \mathsf{Q}(y) \rhd \bar{\mathcal{R}} \rhd \bar{\mathcal{R}} \rhd \bar{\mathcal{R}} \gg & (\lambda x.x \wedge y)\text{'}\,\Lambda x & \equiv & (\lambda x.y \wedge x)\text{'}\,\Lambda x & ; \\
\mathsf{L3}: & \mathsf{Q}(y) \rhd \bar{\mathcal{R}} \rhd \bar{\mathcal{R}} \rhd \bar{\mathcal{R}} \gg & (\lambda x.x \wedge y)\text{'}\,\bot & \equiv & (\lambda x.y \wedge x)\text{'}\,\bot & ; \\
\mathsf{L4}: & \mathsf{Q}(x) \rhd \mathsf{L1} \rhd \mathsf{L2} \rhd \mathsf{L3} \gg & x \wedge y & \equiv & y \wedge x &]^*
\end{array}$$

The proof may also be stated as a one-liner:

[The Map proof number 3 of L6.3.1 reads

$$\mathsf{L1}: \quad \mathsf{Q}(x) \rhd \bar{\mathsf{Q}}(y) \rhd \bar{\mathsf{Q}}(y) \rhd \bar{\mathsf{Q}}(y) \gg \quad x \wedge y \equiv y \wedge x \quad]^*$$

where

$$[\ \bar{\mathsf{Q}}(u) \doteq \mathsf{Q}(u) \rhd \bar{\mathcal{R}} \rhd \bar{\mathcal{R}} \rhd \bar{\mathcal{R}}\ ]^*.$$

The definition

$$[\ \bar{\mathsf{Q}}(u,v) \doteq \mathsf{Q}(u) \rhd \bar{\mathsf{Q}}(v) \rhd \bar{\mathsf{Q}}(v) \rhd \bar{\mathsf{Q}}(v)\ ]^*$$

allows to state the proof even shorter:

[The Map proof number 4 of L6.3.1 reads $\bar{\mathsf{Q}}(x,y) \gg x \wedge y \equiv y \wedge x$]*

6.4 Tautologies

Let [$\mathcal{A} \equiv \mathcal{B}$] be a formula whose free variables are exactly [$z_1, \ldots, z_n$]. A logical instance of [$\mathcal{A} \equiv \mathcal{B}$] is a formula

$$[\ \langle \mathcal{A} \mid z_1 := \mathcal{C}_1 \mid \cdots \mid z_n := \mathcal{C}_n \rangle \equiv \langle \mathcal{B} \mid z_1 := \mathcal{C}_1 \mid \cdots \mid z_n := \mathcal{C}_n \rangle\ ]$$

where each [$\mathcal{C}_i$] is one of the terms [N], [Λz_i], or [$\bot$]. As can be seen, [$\mathcal{A} \equiv \mathcal{B}$] has exactly [3^n] logical instances. Repeated application of the QND inference gives the following result: If all logical instances of a formula [$\mathcal{A} \equiv \mathcal{B}$] are provable then [$\mathcal{A} \equiv \mathcal{B}$] itself is provable.

A formula [$\mathcal{A} \equiv \mathcal{B}$] is a ***simple tautology*** if each logical instance is provable by [$\bar{\mathcal{R}}$]. Hence, all simple tautologies are provable. [$\bar{\mathrm{Q}}(x)$] and [$\bar{\mathrm{Q}}(x,y)$] are special cases of this result which allow to prove simple tautologies with one and two free variables, respectively. Some examples of simple tautologies read:

[$x \wedge x \equiv \approx x$]	[$x \vee x \equiv \approx x$]
[$x \wedge y \equiv y \wedge x$]	[$x \vee y \equiv y \vee x$]
[$(x \wedge y) \wedge z \equiv x \wedge (y \wedge z)$]	[$(x \vee y) \vee z \equiv x \vee (y \vee z)$]
[$x \wedge \neg x \equiv$ ¡x]	[$x \vee \neg x \equiv\, !x$]
[$\neg(x \wedge y) \equiv \neg x \vee \neg y$]	[$\neg(x \vee y) \equiv \neg x \wedge \neg y$]
[$\approx x \equiv \neg\neg x$]	[$\neg\neg\neg x \equiv \neg x$]
[$\approx\approx x \equiv \approx x$]	

A formula [$\mathcal{A} \equiv \mathcal{B}$] is a *tautology* if it is an instance of a simple tautology. The simple tautologies form a proper subset of the tautologies. As an example,

[$\neg\neg\neg(x = y) \equiv \neg(x = y)$]

is a tautology but not a simple tautology. Let $\left[\,\boxed{\mathrm{Tautology}(x)}\,\right]$ denote that [x] is a tautology. The following lemma will be stated without proof:

[Map lemma Logic: $\langle \mathcal{A}, \mathcal{B} \in \mathcal{T} \rangle \mathrm{Tautology}(\mathcal{A} \equiv \mathcal{B}) \Vdash \mathcal{A} \equiv \mathcal{B}$]*

6.5 Non-monotonic implication

Equality of $\lambda\mathsf{N}$ is non-monotonic in the sense that a map [f] would be non-monotonic if it satisfied [$f\,'\,x\,'\,y \equiv \mathsf{T}$] if and only if [$x \equiv y$]. To see this, note that [$\bot \equiv \bot$], so [$f\,'\bot\,'\bot \equiv \mathsf{T}$]. If [f] were monotonic, then [$\mathsf{T} \equiv f\,'\,\bot\,'\,\bot \preceq f\,'\,x\,'\,y$] which implies [$f\,'\,x\,'\,y \equiv \mathsf{T}$] for all [x] and [y]. Hence, no monotonic function [f] can satisfy [$f\,'\,x\,'\,y \equiv \mathsf{T}$] if and only if [$x \equiv y$].

Consequently, no map [f] satisfies [$f\,'\,x\,'\,y \equiv \mathsf{T}$] if and only if [$x \equiv y$] since all maps are monotonic (c.f. Section 2.17).

Put another way, there are things that formulas can express which terms cannot express. Among other, the non-monotonicity of equality allows to express an implication concept [$x \rightarrow y$] that is more powerful than [$x \Rightarrow y$]. [$x \Rightarrow y$] is a term, so it is monotonic in [x] and [y].

For all terms [$\mathcal{A}$] and for all formulas [$\mathcal{S}$], define $\boxed{\mathcal{A} \rightarrow \mathcal{S}}$ by the following convention:

[$\mathcal{A} \rightarrow (\mathcal{B} \equiv \mathcal{C})$] is shorthand for [$\mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{B} \equiv \mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{C}$].

Let [$\mathcal{A} \equiv \mathcal{B}$] have higher priority than [$\mathcal{A} \rightarrow \mathcal{S}$] so that

[$\mathcal{A} \rightarrow \mathcal{B} \equiv \mathcal{C}$] is shorthand for [$\mathcal{A} \rightarrow (\mathcal{B} \equiv \mathcal{C})$].

simple tautology
tautology

Furthermore, let [$\mathcal{A} \to \mathcal{S}$] be right associative so that

[$\mathcal{A} \to \mathcal{B} \to \mathcal{S}$] is shorthand for [$\mathcal{A} \to (\mathcal{B} \to \mathcal{S})$].

As stated in Section 5.11, if a term [$\mathcal{B}$] occurs in a position where a formula is expected then [$\mathcal{B}$] is shorthand for [$\mathcal{B} \equiv \mathsf{T}$]. Hence, if [$\mathcal{A}$] and [$\mathcal{B}$] are terms, then

[$\mathcal{A} \to \mathcal{B}$] is shorthand for [$\mathcal{A} \to \mathcal{B} \equiv \mathsf{T}$].

Informally, if [$\mathcal{A}$], [$\mathcal{B}$], and [$\mathcal{C}$] are terms, then [$\mathcal{A} \to \mathcal{B} \equiv \mathcal{C}$] expresses "if [$\mathcal{A} \equiv \mathsf{T}$] then [$\mathcal{B} \equiv \mathcal{C}$]". To see this informally (or in the framework of ZFC), consider the following three cases:

1. If [$\mathcal{A} \equiv \mathsf{T}$] holds and [$\mathcal{B} \equiv \mathcal{C}$] holds, then [$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{B} \equiv \mathcal{C} \equiv \mathcal{A} \tilde{\wedge} \mathcal{C}$] so [$\mathcal{A} \to \mathcal{B} \equiv \mathcal{C}$] holds.
2. If [$\mathcal{A} \equiv \mathsf{T}$] holds and [$\mathcal{B} \equiv \mathcal{C}$] fails, then [$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{B} \not\equiv \mathcal{C} \equiv \mathcal{A} \tilde{\wedge} \mathcal{C}$] so [$\mathcal{A} \to \mathcal{B} \equiv \mathcal{C}$] fails.
3. If [$\mathcal{A} \equiv \mathsf{T}$] fails then [$\mathcal{A}$] is either [$\perp$] or a function. In the former case, [$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \perp$] and [$\mathcal{A} \tilde{\wedge} \mathcal{C} \equiv \perp$]. In the latter case, [$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathsf{F}$] and [$\mathcal{A} \tilde{\wedge} \mathcal{C} \equiv \mathsf{F}$]. In both cases, [$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{A} \tilde{\wedge} \mathcal{C}$] holds.

Due to the right associativity of [$\to$], [$\mathcal{A}_1 \to \cdots \to \mathcal{A}_n \to \mathcal{B} \equiv \mathcal{C}$] expresses "if [$\mathcal{A}_1, \cdots, \mathcal{A}_n$] are all true then [$\mathcal{B} \equiv \mathcal{C}$]".

6.6 Modus Ponens

The theorem of Modus Ponens reads:

[Map lemma ModusPonens$_{69}$: $\langle \mathcal{A}, \mathcal{B} \in \mathcal{T} \rangle \mathcal{A} \vdash \mathcal{A} \to \mathcal{B} \equiv \mathcal{C} \vdash \mathcal{B} \equiv \mathcal{C}$]*

To save space, [Rev] is introduced as shorthand for [Commutativity]:

$\left[\boxed{\text{Rev}} \doteq \text{Commutativity} \right]^*$.

[The Map proof of ModusPonens reads

L1:	Premise $\gg$	$\mathcal{A}$	;
L2:	Premise $\gg$	$\mathcal{A} \to \mathcal{B} \equiv \mathcal{C}$	;
L3:	Repetition $\rhd$ L1 $\gg$	$\mathcal{A} \equiv \mathsf{T}$	;
L4:	Repetition $\rhd$ L2 $\gg$	$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{A} \tilde{\wedge} \mathcal{C}$	;
L5:	Repetition $\rhd$ L4 $\gg$	$\text{if}(\mathcal{A}, \mathcal{B}, \mathsf{F}) \equiv \text{if}(\mathcal{A}, \mathcal{C}, \mathsf{F})$	;
L6:	Block $\gg$	Begin	;
L7:	Algebra $\gg$	$\mathcal{B}$	;
L8:	Rev $\rhd$ IfNil $\gg$	$\text{if}(\mathsf{T}, \mathcal{B}, \mathsf{F})$	;
L9:	Rev $\rhd$ (Replace $\rhd$ L3) $\gg$	$\text{if}(\mathcal{A}, \mathcal{B}, \mathsf{F})$	;
L10:	Repetition $\rhd$ L5 $\gg$	$\text{if}(\mathcal{A}, \mathcal{C}, \mathsf{F})$	;
L11:	Replace $\rhd$ L3 $\gg$	$\text{if}(\mathsf{T}, \mathcal{C}, \mathsf{F})$	;
L12:	IfNil $\gg$	$\mathcal{C}$	;
L13:	Block $\gg$	End	]*

In Line [$\mathsf{L1}$] above, the term [$\mathcal{A}$] occurs in a position where a formula is expected, so Line [$\mathsf{L1}$] is shorthand for [$\mathcal{A} \equiv \mathsf{T}$]. To enhance the readability of the proof, Line [$\mathsf{L1}$] is repeated in Line [$\mathsf{L3}$], but this time the line is written out. Likewise, Line [$\mathsf{L2}$] is shorthand for the formula in Line [$\mathsf{L4}$] which in turn is shorthand for the formula in Line [$\mathsf{L5}$]. A mechanical proof checker would see Line [$\mathsf{L1}$] and Line [$\mathsf{L3}$] as identical after elimination of shorthand notation and similarly for Line [$\mathsf{L2}$], [$\mathsf{L4}$], and [$\mathsf{L5}$].

Line [$\mathsf{L3}$], [$\mathsf{L4}$], and [$\mathsf{L5}$] are superfluous in principle, but may enhance the readability of the proof.

Note that

$$[\, \mathcal{A} \vdash \mathcal{A} \to \mathcal{B} \vdash \mathcal{B} \,]$$

is a special case of [ModusPonens]. It is the special case where [$\mathcal{C}$] is [T].

6.7 Logical consequences

The following definition may enhance the readability of proofs:

$$\Big[\, \boxed{x \trianglerighteq y} \doteq \text{ModusPonens} \rhd y \rhd x \,\Big]^{*}.$$

As an example of use, consider the following:

[Map lemma L6.7.1$_{70}$: $\langle x, y{\in}\mathcal{T}\rangle x \wedge y \vdash \neg y \vdash x$]*

[The Map proof of L6.7.1 reads

L1:	Premise $\gg$	$x \wedge y$	;
L2:	Premise $\gg$	$\neg y$	;
L3:	Logic $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	x	]*

In Line [$\mathsf{L3}$], unification forces [Logic] to verify

$$[\, x \wedge y \to \neg y \to x \,]$$

which happens to be a tautology.

If [$\mathcal{A}_1 \to \cdots \to \mathcal{A}_n \to \mathcal{B} \equiv \mathcal{C}$] is a tautology then [$\mathcal{B} \equiv \mathcal{C}$] will be said to *follow logically* from [$\mathcal{A}_1, \ldots, \mathcal{A}_n$]. As shown above, [Logic] allows to conclude [$\mathcal{B} \equiv \mathcal{C}$] from [$\mathcal{A}_1, \ldots, \mathcal{A}_n$] if the former follows logically from the latter.

6.8 Deduction

The deduction theorem of classical mathematics more or less says that if [$\mathcal{A} \vdash \mathcal{B}$] then [$\mathcal{A} \Rightarrow \mathcal{B}$]. The corresponding theorem of $\lambda\mathsf{N}$ (and MT) more or less says that if [$\mathcal{A} \vdash \mathcal{S}$] then [$\mathcal{A} \to \mathcal{S}$]. Or, written out in more detail, if

follow logically

[$\mathcal{A} \equiv \mathsf{T} \vdash \mathcal{B} \equiv \mathcal{C}$] then [$\mathcal{A} \to \mathcal{B} \equiv \mathcal{C}$]. This section illustrates the notation that will be used when deduction is used in proofs.

[Map lemma CommuteHyp$_{71}$: $\langle \mathcal{A}, \mathcal{B}, \mathcal{C} {\in} \mathcal{T} \rangle \mathcal{A} \to \mathcal{B} \to \mathcal{C} \vdash \mathcal{B} \to \mathcal{A} \to \mathcal{C}$]*

[The Map proof of CommuteHyp reads

L1:	Premise $\gg$	$\mathcal{A} \to \mathcal{B} \to \mathcal{C}$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\mathcal{B}$	;
L4:	Block $\gg$	Begin	;
L5:	Hypothesis $\gg$	$\mathcal{A}$	;
L6:	L1 $\trianglerighteq$ L5 $\gg$	$\mathcal{B} \to \mathcal{C}$	;
L7:	L6 $\trianglerighteq$ L3 $\gg$	$\mathcal{C}$	;
L8:	Block $\gg$	End	;
L9:	Repetition $\rhd$ L7 $\gg$	$\mathcal{A} \to \mathcal{C}$	;
L10:	Block $\gg$	End	;
L11:	Repetition $\rhd$ L9 $\gg$	$\mathcal{B} \to \mathcal{A} \to \mathcal{C}$	]*

In the proof, [$\mathcal{B} \equiv \mathsf{T}$] is a deduction hypothesis; it is assumed from Line 2 to 10. The deduction hypothesis [$\mathcal{A} \equiv \mathsf{T}$] is assumed from Line 4 to 8. The premise [$\mathcal{A} \to \mathcal{B} \to \mathcal{C}$] is assumed throughout the proof and is unrelated to deduction.

The proof above is shorthand for a formal proof that merely uses the premise [$\mathcal{A} \to \mathcal{B} \to \mathcal{C}$] of [CommuteHyp] and the axioms and inferences of $\lambda\mathsf{N}$. Section 6.11 describes how to transform shorthand proofs into formal proofs.

The transformation defined in Section 6.11 will be referred to as *shorthand elimination*. Shorthand elimination applied to the proof above produces a proof with the following skeleton:

[The Map proof of CommuteHyp reads

L1:	Premise $\gg$	$\mathcal{A} \to \mathcal{B} \to \mathcal{C}$	;
$\vdots$	$\vdots$	$\vdots$	;
L3:	$\cdots \gg$	$\mathcal{B} \to \mathcal{B}$	;
$\vdots$	$\vdots$	$\vdots$	;
L5:	$\cdots \gg$	$\mathcal{B} \to \mathcal{A} \to \mathcal{A}$	;
$\vdots$	$\vdots$	$\vdots$	;
L6:	$\cdots \gg$	$\mathcal{B} \to \mathcal{A} \to \mathcal{B} \to \mathcal{C}$	;
$\vdots$	$\vdots$	$\vdots$	;
L7:	$\cdots \gg$	$\mathcal{B} \to \mathcal{A} \to \mathcal{C}$	;
L9:	Repetition $\rhd$ L7 $\gg$	$\mathcal{B} \to \mathcal{A} \to \mathcal{C}$	;
L11:	Repetition $\rhd$ L9 $\gg$	$\mathcal{B} \to \mathcal{A} \to \mathcal{C}$	]

Shorthand elimination deletes lines that say “Block Begin” and “Block End”. Furthermore, shorthand elimination prepends each remaining proof line with

shorthand elimination

all deduction hypotheses that are in effect at the given line. Finally, shorthand elimination adds extra proof lines and expands metatheorems like Repetition. To save space, Repetition is not expanded in the skeleton above.

The skeleton above shows why the use of Repetition in Line 9 and 11 are legal.

6.9 A false deduction proof

Like in classical logic, the deduction theorem of $\lambda\mathsf{N}$ imposes a restriction on the use of some of the inferences: In each proof line, the deduction theorem forbids use of SubLambda and Extensionality on a variable that is free in any hypothesis that is in effect in the given proof line. As an example, the following false proof of [$\lambda x.x \equiv \lambda x.\mathsf{T}$] illustrates abuse of deduction:

[Map lemma FalseLemma: $\lambda x.x \equiv \lambda x.\mathsf{T}$]

[The Map proof of FalseLemma reads

L1:	Block $\gg$	Begin	;
L2:	Hypothesis $\gg$	x	;
L3:	Repetition $\rhd$ L2 $\gg$	$x \equiv \mathsf{T}$	;
L4:	SubLambda $\rhd$ L3 $\gg$	$\lambda x.x \equiv \lambda x.\mathsf{T}$ ERROR!	;
L5:	Block $\gg$	End	;
L6:	Repetition $\rhd$ L4 $\gg$	$x \rightarrow \lambda x.x \equiv \lambda x.\mathsf{T}$	;
L7:	Instantiation $\rhd$ L6 $\gg$	$\mathsf{T} \rightarrow \lambda x.x \equiv \lambda x.\mathsf{T}$	;
L8:	Reflexivity $\gg$	$\mathsf{T} \equiv \mathsf{T}$	;
L9:	Repetition $\rhd$ L8 $\gg$	T	;
L10:	L7 $\trianglerighteq$ L9 $\gg$	$\lambda x.x \equiv \lambda x.\mathsf{T}$	]

Line 4 is in error because

1. [x] occurs free in the hypothesis in Line 2,
2. the hypothesis in Line 2 is in effect in Line 4, and
3. SubLambda(3) in Line 4 is used on [x].

In classical logic, the deduction theorem merely puts a restriction on one of the inference rules. The deduction theorem of $\lambda\mathsf{N}$ puts a restriction on both [SubLambda] and [Extensionality]. If it is desirable that the deduction theorem of $\lambda\mathsf{N}$ merely puts a restriction on one of the inference rules rather than two, then one has to reformulate [Extensionality].

6.10 Lemmas for shorthand elimination

Shorthand elimination as described in Section 6.11 makes use of two lemmas which will be referred to as [Ded:Hyp] and [Ded:Import]. In addition, shorthand elimination makes use of one lemma for each inference rule of $\lambda\mathsf{N}$. The lemmas that correspond to [Trans], [SubLambda], [SubApply], and [QND] will be referred to as [Ded:Trans], [Ded:SubLambda], [Ded:SubApply], and [Ded:QND], respectively. The lemmas read:

$$\begin{array}{lll}
[\ \text{Map lemma} & & \\
\ \text{Ded:Hyp}_{124}\text{:} & \langle \mathcal{H}{\in}\mathcal{T}\rangle \mathcal{H} \rightarrow \mathcal{H} & ; \\
\ \text{Ded:Import}_{124}\text{:} & \langle \mathcal{A},\mathcal{B},\mathcal{H}{\in}\mathcal{T}\rangle \mathcal{A} \equiv \mathcal{B} \vdash \mathcal{H} \rightarrow \mathcal{A} \equiv \mathcal{B} & ; \\
\ \text{Ded:Trans}_{124}\text{:} & \langle \mathcal{A},\mathcal{B},\mathcal{C},\mathcal{H}{\in}\mathcal{T}\rangle & \\
 & \mathcal{H} \rightarrow \mathcal{A} \equiv \mathcal{B} & \vdash \\
 & \mathcal{H} \rightarrow \mathcal{A} \equiv \mathcal{C} & \vdash \\
 & \mathcal{H} \rightarrow \mathcal{B} \equiv \mathcal{C} & ; \\
\ \text{Ded:SubLambda}_{125}\text{:} & \langle \mathcal{A},\mathcal{B},\mathcal{H}{\in}\mathcal{T}\rangle \neg\text{free}(x,\mathcal{H}) & \Vdash \\
 & \mathcal{H} \rightarrow \mathcal{A} \equiv \mathcal{B} & \vdash \\
 & \mathcal{H} \rightarrow \lambda x.\mathcal{A} \equiv \lambda x.\mathcal{B} & ; \\
\ \text{Ded:SubApply}_{125}\text{:} & \langle \mathcal{A},\mathcal{B},\mathcal{C},\mathcal{H}{\in}\mathcal{T}\rangle & \\
 & \mathcal{H} \rightarrow \mathcal{A} \equiv \mathcal{B} & \vdash \\
 & \mathcal{H} \rightarrow \mathcal{C}\,'\mathcal{A} \equiv \mathcal{C}\,'\mathcal{B} & ; \\
\ \text{Ded:QND}_{126}\text{:} & \langle \mathcal{A},\mathcal{B},\mathcal{H}{\in}\mathcal{T}\rangle & \\
 & \mathcal{H} \rightarrow \mathcal{A}\,'\mathsf{N} \equiv \mathcal{B}\,'\mathsf{N} & \vdash \\
 & \mathcal{H} \rightarrow \mathcal{A}\,'\Lambda\mathcal{C} \equiv \mathcal{B}\,'\Lambda\mathcal{C} & \vdash \\
 & \mathcal{H} \rightarrow \mathcal{A}\,'\bot \equiv \mathcal{B}\,'\bot & \vdash \\
 & \mathcal{H} \rightarrow \mathcal{A}\,'\mathcal{C} \equiv \mathcal{B}\,'\mathcal{C} &]^{*}
\end{array}$$

The lemmas that correspond to the remaining inferences of $\lambda\mathsf{N}$ are stated in Section 7.1, Section 7.9, and Section 7.11, respectively.

6.11 Shorthand elimination

Proofs that do use deduction are shorthand for proofs that do not. The following procedure specifies how to translate shorthand proofs into pure formal proofs. The procedure will be referred to as *shorthand elimination*.

The procedure given in the following is simple but the number of proof lines it produces is exponential in the number of nested blocks. For actual use with mechanical proof checkers one has to modify the procedure slightly in order to reduce the number of produced lines.

Shorthand elimination is a recursive procedure. Given a shorthand proof, shorthand elimination does as follows:

Step 1

Expand all shorthand notation (algebraic blocks, defined concepts, lemmas, and so one) except deduction blocks. The resulting proof uses nothing but axioms, inference rules, and deduction blocks. If the resulting proof has no deductions blocks, stop. Otherwise, proceed with Step 2.

shorthand elimination

Step 2

Identify the last occurrence of "Block End" and the matching occurrence of "Block Begin". Apply shorthand elimination to the lines before "Block Begin" (eliminating deduction blocks in this area). Then apply shorthand elimination to the lines between "Block Begin" and "Block End" (eliminating deduction blocks in this area). The resulting proof uses axioms and inference rules and contains exactly one deduction block.

Step 3

The remaining deduction block has form

$$\begin{array}{rlll}
[\,a: & \text{Block} \gg & \text{Begin} & ; \\
b_0: & \text{Hypothesis} \gg & \mathcal{A} & ; \\
b_1: & \mathcal{Z}_1 \gg & \mathcal{B}_1 \equiv \mathcal{C}_1 & ; \\
\vdots & \vdots & \vdots & ; \\
b_n: & \mathcal{Z}_n \gg & \mathcal{B}_n \equiv \mathcal{C}_n & ; \\
c: & \text{Block} \gg & \text{End} &]
\end{array}$$

Within the block, do the following and then proceed with Step 1: Delete line [a] and [c]. Prepend [$\mathcal{A} \rightarrow$] to the lines [$b_0, \ldots, b_n$]. Replace [Hypothesis] by [Ded:Hyp]. In [$\mathcal{Z}_i$], replace references [z] to proof lines outside the deduction block with [Ded:Import $\rhd z$]. In [$\mathcal{Z}_i$], replace references [z] to axiom schemes with [Ded:Import $\rhd z$]. In [$\mathcal{Z}_i$], replace references to [Trans] by [Ded:Trans] and similarly for the other inference rules.

6.12 References in and out of blocks

Shorthand elimination converts the following shorthand proof into a correct, formal proof. The example illustrates references in and out of blocks.

[Map lemma L6.12.1$_{74}$: $\langle \mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D} \in \mathcal{T} \rangle \mathcal{A} \rightarrow \mathcal{B} \rightarrow \mathcal{C} \rightarrow \mathcal{D} \rightarrow \mathcal{C}$]*

[The Map proof of L6.12.1 reads

L1:	Block $\gg$	Begin	;
L2:	Hypothesis $\gg$	$\quad\mathcal{C}$	;
L3:	Block $\gg$	$\quad$Begin	;
L4:	Hypothesis $\gg$	$\qquad\mathcal{D}$	;
L5:	Repetition $\rhd$ L2 $\gg$	$\qquad\mathcal{C}$	;
L6:	Block $\gg$	$\quad$End	;
L7:	Block $\gg$	End	;
L8:	Block $\gg$	Begin	;
L9:	Hypothesis $\gg$	$\quad\mathcal{A}$	;
L10:	Block $\gg$	$\quad$Begin	;
L11:	Hypothesis $\gg$	$\qquad\mathcal{B}$	;
L12:	Repetition $\rhd$ L5 $\gg$	$\qquad\mathcal{C} \to \mathcal{D} \to \mathcal{C}$	;
L13:	Block $\gg$	$\quad$End	;
L14:	Block $\gg$	End	;
L15:	Repetition $\rhd$ L12 $\gg$	$\mathcal{A} \to \mathcal{B} \to \mathcal{C} \to \mathcal{D} \to \mathcal{C}$	]*

6.13 Proofs that end inside a block

A proof may end inside a block:

[Map lemma InsideBlock$_{75}$: $\langle \mathcal{A}, \mathcal{B} \in \mathcal{T} \rangle \mathcal{A} \to \mathcal{B} \to \mathcal{A}$]*

[The Map proof of InsideBlock reads

L1:	Block $\gg$	Begin	;
L2:	Hypothesis $\gg$	$\quad\mathcal{A}$	;
L3:	Block $\gg$	$\quad$Begin	;
L4:	Hypothesis $\gg$	$\qquad\mathcal{B}$	;
L5:	Repetition $\rhd$ L2 $\gg$	$\qquad\mathcal{A}$	;
L6:	Block $\gg$	$\quad$End	;
L7:	Block $\gg$	End	]*

Shorthand elimination will transform the proof above into one in which Line 5 is the last line. After shorthand elimination, Line 5 will read [$\mathcal{A} \to \mathcal{B} \to \mathcal{A}$]. Hence, the proof proves InsideBlock.

6.14 Blocks with several hypotheses

Until further, each hypothesis has occurred as the first line of a block. From now on, whenever a hypothesis occurs in any other position, then it is understood that there is a block that starts right before the hypothesis and ends at the end of the smallest enclosing block. If there is no enclosing block, then the understood block ends at the end of the proof. With these conventions, the proof of InsideBlock can be stated the following two ways:

[The Map proof number 2 of InsideBlock reads

L1:	Block $\gg$	Begin	;
L2:	Hypothesis $\gg$	$\mathcal{A}$	;
L3:	Hypothesis $\gg$	$\mathcal{B}$	;
L4:	Repetition $\rhd$ L2 $\gg$	$\mathcal{A}$	;
L5:	Block $\gg$	End	]*

[The Map proof number 3 of InsideBlock reads

L1:	Hypothesis $\gg$	$\mathcal{A}$	;
L2:	Hypothesis $\gg$	$\mathcal{B}$	;
L3:	Repetition $\rhd$ L1 $\gg$	$\mathcal{A}$	]*

6.15 Blocks with no hypotheses

Blocks with no hypotheses have no effect except that they delimit the scope of local definitions (c.f. Section A.3) made within them.

6.16 QND again

The following lemma follows immediately from [Q].

[Map lemma Q'$(x)_{126}$: $\langle \mathcal{A}, \mathcal{A}', \mathcal{A}'', \mathcal{A}''', \mathcal{B}, \mathcal{B}', \mathcal{B}'', \mathcal{B}''' \in \mathcal{T} \rangle$
$\mathcal{A}' \simeq \langle \mathcal{A} \mid x{:=}\mathsf{T} \rangle \wedge \mathcal{B}' \simeq \langle \mathcal{B} \mid x{:=}\mathsf{T} \rangle \wedge$
$\mathcal{A}'' \simeq \langle \mathcal{A} \mid x{:=}\Lambda x \rangle \wedge \mathcal{B}'' \simeq \langle \mathcal{B} \mid x{:=}\Lambda x \rangle \wedge$
$\mathcal{A}''' \simeq \langle \mathcal{A} \mid x{:=}\bot \rangle \wedge \mathcal{B}''' \simeq \langle \mathcal{B} \mid x{:=}\bot \rangle \Vdash$
$\mathcal{A}' \equiv \mathcal{B}' \vdash \mathcal{A}'' \equiv \mathcal{B}'' \vdash \mathcal{A}''' \equiv \mathcal{B}''' \vdash \mathcal{A} \equiv \mathcal{B}$]*

6.17 Tertium Non Datur (TND)

The following theorem expresses Tertium Non Datur: If [$\mathcal{A}$] is a truth value, then [$\mathcal{A} \to \mathcal{B} \equiv \mathcal{C}$] and [$\neg\mathcal{A} \to \mathcal{B} \equiv \mathcal{C}$] together imply [$\mathcal{B} \equiv \mathcal{C}$].

[Map lemma TND$_{126}$: $\langle \mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathcal{T} \rangle ! \mathcal{A} \vdash \mathcal{A} \to \mathcal{B} \equiv \mathcal{C} \vdash \neg\mathcal{A} \to \mathcal{B} \equiv \mathcal{C} \vdash \mathcal{B} \equiv \mathcal{C}$]*

6.18 Indirect proofs

The following theorems justify the method of indirect proof:

[Map lemma

Indirect$_{127}$:	$\langle \mathcal{A} \in \mathcal{T} \rangle ! \mathcal{A} \vdash \neg\mathcal{A} \to \mathsf{F} \vdash \mathcal{A}$	;
Indirect'$_{127}$:	$\langle \mathcal{A}, \mathcal{B} \in \mathcal{T} \rangle ! \mathcal{A} \to ! \mathcal{B} \vdash \neg\mathcal{B} \to \neg\mathcal{A} \vdash \mathcal{A} \to \mathcal{B}$	]*

6.19 Classical deduction

The deduction theorem of classical logic more or less says that if [$\mathcal{A} \vdash \mathcal{B}$] then [$\mathcal{A} \Rightarrow \mathcal{B}$]. That theorem does not hold in $\lambda\mathsf{N}$ since [$\mathcal{A}$] or [$\mathcal{B}$] or both may equal [$\bot$]. However, [CDeduction] combined with the deduction theorem of $\lambda\mathsf{N}$ allows to prove formulas of form [$\mathcal{A} \Rightarrow \mathcal{B}$] (i.e. formulas of form [$(\mathcal{A} \Rightarrow \mathcal{B}) \equiv \mathsf{T}$]). [Deduction"] and [IntroIff] present two further kinds of deduction.

[Map lemma
CDeduction$_{127}$: $\langle \mathcal{A}, \mathcal{B} \in \mathcal{T} \rangle !\mathcal{A} \vdash !\mathcal{B} \vdash \mathcal{A} \rightarrow \mathcal{B} \vdash \mathcal{A} \Rightarrow \mathcal{B}$;
Deduction"$_{128}$: $\langle \mathcal{A}, \mathcal{B} \in \mathcal{T} \rangle !\mathcal{A} \vdash \mathcal{A} \rightarrow \mathcal{B} \vdash \mathcal{A} \Rightarrow \mathcal{B}$;
IntroIff$_{128}$: $\langle \mathcal{A}, \mathcal{B} \in \mathcal{T} \rangle !\mathcal{A} \vdash !\mathcal{B} \vdash \mathcal{A} \rightarrow \mathcal{B} \vdash \mathcal{B} \rightarrow \mathcal{A} \vdash \mathcal{A} \Leftrightarrow \mathcal{B}$]*

6.20 Anti-lemmas and anti-proofs

Like any other axiomatic theory, $\lambda\mathsf{N}$ and MT have a number of rules. In addition, $\lambda\mathsf{N}$ and MT have an *anti-rule* which reads:

[Map anti rule Contradiction: $\bot \equiv \mathsf{T}$]*

[$\bot \equiv \mathsf{T}$] is an anti-rule in the sense that $\lambda\mathsf{N}$ and MT are considered inconsistent if the anti-rule turns out to be provable from the rules.

A statement [$\mathcal{R}$] is an *anti-lemma*, written $\left[\,\boxed{\nvdash}\,\mathcal{R}\,\right]$, if the formula has an *anti-proof*. An anti-proof of a statement [$\mathcal{R}$] is a proof of [$\mathcal{R} \vdash \mathcal{Z}$] where [$\mathcal{Z}$] denotes the anti-rule.

In $\lambda\mathsf{N}$ and MT, an anti-proof of [$\mathcal{R}$] is a proof of [$\mathcal{R} \vdash \bot$]. Note that in [$\mathcal{R} \vdash \bot$], the term [$\bot$] occurs in a position where a formula is expected, so [$\bot$] is shorthand for [$\bot \equiv \mathsf{T}$] which is the anti-rule. Hence, in $\lambda\mathsf{N}$ and MT, an anti-proof of a statement [$\mathcal{R}$] is a proof of [$\mathcal{R} \vdash \bot \equiv \mathsf{T}$]. Here is an example:

[Map anti lemma L6.20.1$_{77}$: $2 + 2 \equiv 5$]*

[The Map proof of L6.20.1 reads

L1:	Premise $\gg$	$2 + 2 \equiv 5$	;
L2:	Block $\gg$	Begin	;
L3:	Algebra $\gg$	$\bot$	;
L4:	Reduction $\gg$	$\mathrm{if}(2 + 2 = 5, \mathsf{T}, \bot)$	;
L5:	Replace $\rhd$ L1 $\gg$	$\mathrm{if}(5 = 5, \mathsf{T}, \bot)$	;
L6:	Reduction $\gg$	T	;
L7:	Block $\gg$	End	;
L8:	Repetition $\rhd$ L6 $\gg$	$\bot$	]*

The formula [$2+x \equiv 5$] is also an anti-lemma because [$2+x \equiv 5$] is disprovable for at least one [x]:

anti-rule
anti-lemma
anti-proof

[Map anti lemma L6.20.2$_{78}$: $2 + x \equiv 5$]*

[The Map proof of L6.20.2 reads

L1:	Premise $\gg$	$2 + x \equiv 5$	;
L2:	Instantiation $\rhd$ L1 $\gg$	$2 + 2 \equiv 5$	;
L3:	L6.20.1 $\rhd$ L2 $\gg$	$\perp$	]*

Anti-lemmas are convenient in connection with mechanical proof assistants. Working mathematicians typically state many conjectures and succeed to prove some of them. Invariably, working mathematicians state conjectures that turn out to fail. If the mathematician deletes wrong conjectures, then the mathematician or the mechanical proof assistant may repeat the failure and proclaim the same false conjecture later on. Hence, it may be more convenient to keep false conjectures in the form of anti-lemmas. In other words, the anti-lemma formalism above is a formalism that allows to learn from failure.

Chapter 7

Further advanced rules

7.1 Extensionality

Recall Lemma 2.11.2: For all $[\ \mathcal{A}, \mathcal{B} \in \bar{\mathcal{T}}\ ]$,

$$[\ \mathcal{A} \equiv \mathcal{B} \Leftrightarrow \forall \mathcal{X} {\in} \bar{\mathcal{T}}^{<\omega}{:}\, \mathcal{A}\text{''}\, \mathcal{X} \approx \mathcal{B}\text{''}\, \mathcal{X}\ ].$$

It is non-trivial to formulate this as an inference rule. To formulate extensionality, first note that

$$[\ \mathcal{A} \equiv \mathcal{B} \Rightarrow \forall \mathcal{X} {\in} \bar{\mathcal{T}}^{<\omega}{:}\, \mathcal{A}\text{''}\, \mathcal{X} \approx \mathcal{B}\text{''}\, \mathcal{X}\ ]$$

follows from [Replace]. Hence, the inference of extensionality merely has to express

$$[\ \forall \mathcal{X} {\in} \bar{\mathcal{T}}^{<\omega}{:}\, \mathcal{A}\text{''}\, \mathcal{X} \approx \mathcal{B}\text{''}\, \mathcal{X} \Rightarrow \mathcal{A} \equiv \mathcal{B}\ ].$$

The following turns out to be a reasonable formulation:

[Map rule Extensionality: $\langle u, v {\in} \mathcal{V}; \mathcal{A}, \mathcal{B}, \mathcal{C} {\in} \mathcal{T} \rangle \mathrm{notfree}(u, v; \mathcal{A}, \mathcal{B}, \mathcal{C}) \Vdash$
$\approx(\mathcal{A}\text{'}\, u) \equiv \approx(\mathcal{B}\text{'}\, u) \vdash \mathcal{A}\text{'}\, u\text{'}\, v \equiv \mathcal{A}\text{'}(\mathcal{C}\text{'}\, u\text{'}\, v) \vdash \mathcal{B}\text{'}\, u\text{'}\, v \equiv \mathcal{B}\text{'}(\mathcal{C}\text{'}\, u\text{'}\, v) \vdash$
$\mathcal{A}\text{'}\, u \equiv \mathcal{B}\text{'}\, u$]*

The side condition of [Extensionality] requires [u] and [v] to be distinct variables that are not free in [$\mathcal{A}$], [$\mathcal{B}$], and [$\mathcal{C}$]. In general, if [$x_1, \ldots, x_m$] are variables and [$\mathcal{A}_1, \ldots, \mathcal{A}_n$] are terms, then

$$\left[\ \boxed{\mathrm{notfree}(x_1, \ldots, x_m; \mathcal{A}_1, \ldots, \mathcal{A}_n)}\ \right]$$

states that the variables are distinct and that none of the variables occurs free in any of the terms.

To see how [Extensionality] works, suppose [$\mathcal{A}$], [$\mathcal{B}$], and [$\mathcal{C}$] satisfy the side conditions and premises of [Extensionality], and suppose [u], [v_1], [v_2], and [v_3] are distinct variables that are not free in [$\mathcal{A}$], [$\mathcal{B}$], and [$\mathcal{C}$]. Define [u_0], [u_1], [u_2], and [u_3] thus:

$$\begin{array}{llll}
[& u_0 & \equiv & u &] \\
[& u_1 & \equiv & \mathcal{C}\,'\,u_0\,'\,v_1 &] \\
[& u_2 & \equiv & \mathcal{C}\,'\,u_1\,'\,v_2 &] \\
[& u_3 & \equiv & \mathcal{C}\,'\,u_2\,'\,v_3 &]
\end{array}$$

According to the three premises, the following holds:

$$\begin{array}{lll}
[& \approx(\mathcal{A}\,'\,u\,'\,v_1\,'\,v_2\,'\,v_3) & \equiv \\
 & \approx(\mathcal{A}\,'\,u_0\,'\,v_1\,'\,v_2\,'\,v_3) & \equiv \\
 & \approx(\mathcal{A}\,'\,u_1\,'\,v_2\,'\,v_3) & \equiv \\
 & \approx(\mathcal{A}\,'\,u_2\,'\,v_3) & \equiv \\
 & \approx(\mathcal{A}\,'\,u_3) & \equiv \\
 & \approx(\mathcal{B}\,'\,u_3) & \equiv \\
 & \approx(\mathcal{B}\,'\,u_2\,'\,v_3) & \equiv \\
 & \approx(\mathcal{B}\,'\,u_1\,'\,v_2\,'\,v_3) & \equiv \\
 & \approx(\mathcal{B}\,'\,u_0\,'\,v_1\,'\,v_2\,'\,v_3) & \equiv \\
 & \approx(\mathcal{B}\,'\,u\,'\,v_1\,'\,v_2\,'\,v_3) &]
\end{array}$$

Hence, the premises infer

$$[\ \approx(\mathcal{A}\,'\,u\,'\,v_1\,'\,v_2\,'\,v_3) \equiv \approx(\mathcal{B}\,'\,u\,'\,v_1\,'\,v_2\,'\,v_3)\ ].$$

In general, the premises infer

$$[\ \approx(\mathcal{A}\,'\,u\,'\,v_1\,'\cdots'\,v_n) \equiv \approx(\mathcal{B}\,'\,u\,'\,v_1\,'\cdots'\,v_n)\ ]$$

for all [$n \geq 0$], which entails

$$[\ \mathcal{A}\,'\,u \equiv \mathcal{B}\,'\,u\ ]$$

by Lemma 2.11.2. This is what [Extensionality] says.

7.2 Extensionality and deduction

Shorthand elimination depends on one lemma per inference rule. For Extensionality, the lemma reads:

[Map lemma Ded:Extensionality$_{132}$: notfree$(u, v; \mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{H}) \Vdash$
$\mathcal{H} \to \approx(\mathcal{A}\,'\,u) \equiv \approx(\mathcal{B}\,'\,u) \vdash$
$\mathcal{H} \to \mathcal{A}\,'\,u\,'\,v \equiv \mathcal{A}\,'(\mathcal{C}\,'\,u\,'\,v) \vdash$
$\mathcal{H} \to \mathcal{B}\,'\,u\,'\,v \equiv \mathcal{B}\,'(\mathcal{C}\,'\,u\,'\,v) \vdash$
$\mathcal{H} \to \mathcal{A}\,'\,u \equiv \mathcal{B}\,'\,u$]*

7.3 Greatest lower bounds

As in Section 2.20 define

$$\left[x \downarrow y \doteq x \left\{ \begin{array}{l} \mathrm{if}(y, \mathsf{N}, \bot) \\ \mathrm{if}(y, \bot, \lambda z . x \,\text{'}\, z \downarrow y \,\text{'}\, z) \end{array} \right. \right]^* .$$

[$x \downarrow y$] denotes the greatest lower bound of [x] and [y]. [$x \downarrow y$] allows to define [$x \preceq y$] thus:

$$[\, x \preceq y \doteq x \equiv x \downarrow y \,]^* .$$

The definition above defines [$x \preceq y$] as shorthand for the formula [$x \equiv x \downarrow y$]. The definition of [$x \preceq y$] above allows to state the inference rules of [Monotonicity] and [Minimality] in Section 7.9 and Section 7.11, respectively.

In this paper, [Extensionality] is merely used to prove four lemmas. One of the lemmas is [Ded:Extensionality] which was stated in Section 7.2 to allow use of [Extensionality] inside deduction blocks. The other three lemmas read:

[Map lemma

L7.3.1$_{128}$:	$x \equiv x \downarrow x$	;
L7.3.2$_{130}$:	$x \downarrow y \equiv y \downarrow x$	;
L7.3.3$_{130}$:	$x \downarrow (y \downarrow z) \equiv (x \downarrow y) \downarrow z$	]*

The three lemmas above allow to prove that [$x \preceq y$] is a partial order and to prove a few more lemmas above [$x \preceq y$].

7.4 Partial ordering

The following lemma states that [$x \preceq y$] is a *partial order* (*p.o.*).

[Map lemma

L7.4.1$_{81}$:	$\langle \mathcal{A} {\in} \mathcal{T} \rangle$	$\mathcal{A} \preceq \mathcal{A}$	;
L7.4.2$_{82}$:	$\langle \mathcal{A}, \mathcal{B} {\in} \mathcal{T} \rangle$	$\mathcal{A} \preceq \mathcal{B} \vdash \mathcal{B} \preceq \mathcal{A} \vdash \mathcal{A} \equiv \mathcal{B}$	;
L7.4.3$_{82}$:	$\langle \mathcal{A}, \mathcal{B}, \mathcal{C} {\in} \mathcal{T} \rangle$	$\mathcal{A} \preceq \mathcal{B} \vdash \mathcal{B} \preceq \mathcal{C} \vdash \mathcal{A} \preceq \mathcal{C}$	]*

The lemmas follow directly from L7.3.1, L7.3.2, and L7.3.3:

[The Map proof of L7.4.1 reads

L1:	Instantiation ▷ L7.3.1 ≫	$\mathcal{A} \equiv \mathcal{A} \downarrow \mathcal{A}$	;
L2:	Repetition ▷ L1 ≫	$\mathcal{A} \preceq \mathcal{A}$	]*

partial order
p.o.

[The Map proof of L7.4.2 reads

L1:	Premise $\gg$	$\mathcal{A} \preceq \mathcal{B}$	;
L2:	Premise $\gg$	$\mathcal{B} \preceq \mathcal{A}$	;
L3:	Block $\gg$	Begin	;
L4:	Algebra $\gg$	$\mathcal{A}$	;
L5:	Repetition $\rhd$ L1 $\gg$	$\mathcal{A} \downarrow \mathcal{B}$	;
L6:	Instantiation $\rhd$ L7.3.2 $\gg$	$\mathcal{B} \downarrow \mathcal{A}$	;
L7:	Commutativity $\rhd$ L2 $\gg$	$\mathcal{B}$	;
L8:	Block $\gg$	End	]*

[The Map proof of L7.4.3 reads

L1:	Premise $\gg$	$\mathcal{A} \preceq \mathcal{B}$	;
L2:	Premise $\gg$	$\mathcal{B} \preceq \mathcal{C}$	;
L3:	Block $\gg$	Begin	;
L4:	Algebra $\gg$	$\mathcal{A}$	;
L5:	Repetition $\rhd$ L1 $\gg$	$\mathcal{A} \downarrow \mathcal{B}$	;
L6:	Replace $\rhd$ L2 $\gg$	$\mathcal{A} \downarrow (\mathcal{B} \downarrow \mathcal{C})$	;
L7:	Instantiation $\rhd$ L7.3.3 $\gg$	$(\mathcal{A} \downarrow \mathcal{B}) \downarrow \mathcal{C}$	;
L8:	Rev $\rhd$ (Replace $\rhd$ L1) $\gg$	$\mathcal{A} \downarrow \mathcal{C}$	;
L9:	Block $\gg$	End	]*

7.5 Monotonicity of abstraction

Later on, Monotonicity states that application [$x\,'\,y$] is monotonic. That abstraction is monotonic does not depend on Monotonicity; it merely depends on the definition of [$x \preceq y$] and rules that are already stated:

[Map lemma L7.5.1$_{133}$: $\langle x{\in}\mathcal{V}; \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle \mathcal{A} \preceq \mathcal{B} \vdash \lambda x.\mathcal{A} \preceq \lambda x.\mathcal{B}$]*

7.6 Some minimal and maximal elements

The following lemma states that, with respect to [$x \preceq y$], [T] is a maximal element, [$\bot$] is the unique bottom element, and [$\lambda x.\bot$] is the unique bottom element among functions.

[Map lemma

L7.6.1$_{133}$:	$\langle \mathcal{A}{\in}\mathcal{T}\rangle \bot \preceq \mathcal{A}$	;
L7.6.2$_{133}$:	$\langle \mathcal{A}{\in}\mathcal{T}\rangle \mathcal{A} \preceq \bot \vdash \mathcal{A} \equiv \bot$	;
L7.6.3$_{133}$:	$\langle \mathcal{A}{\in}\mathcal{T}\rangle \mathsf{T} \preceq \mathcal{A} \vdash \mathcal{A} \equiv \mathsf{T}$	;
L7.6.4$_{133}$:	$\langle x{\in}\mathcal{V}; \mathcal{A}{\in}\mathcal{T}\rangle \mathrm{notfree}(x,\mathcal{A}) \Vdash \lambda x.\bot \preceq \mathcal{A} \vdash \mathcal{A} \equiv \lambda x.\mathcal{A}\,'\,x$	;
L7.6.5$_{133}$:	$\langle x{\in}\mathcal{V}; \mathcal{A}{\in}\mathcal{T}\rangle \mathrm{notfree}(x,\mathcal{A}) \Vdash \mathcal{A} \equiv \lambda x.\mathcal{A}\,'\,x \vdash \lambda x.\bot \preceq \mathcal{A}$	]*

[L7.6.3] allows to prove a new form for Modus Ponens:

[Map lemma MP'$_{133}$: $\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle \mathcal{A} \vdash \mathcal{A} \preceq \mathcal{B} \vdash \mathcal{B}$]*

7.7 Greatest lower bounds again

The following lemma states that [$x \downarrow y$] actually is the greatest lower bound of [x] and [y] with respect to [$x \preceq y$]:

[Map lemma
L7.7.1$_{134}$: $\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle \mathcal{A} \downarrow \mathcal{B} \preceq \mathcal{A}$;
L7.7.2$_{134}$: $\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle \mathcal{A} \downarrow \mathcal{B} \preceq \mathcal{B}$;
L7.7.3$_{134}$: $\langle \mathcal{A},\mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle \mathcal{A} \preceq \mathcal{B}; \mathcal{A} \preceq \mathcal{C} \vdash \mathcal{A} \preceq \mathcal{B} \downarrow \mathcal{C}$]*

7.8 Order and implication

Let [$?x$] be given by

$$\left[\boxed{?x} \doteq x\left\{\begin{array}{c}\mathsf{T}\\ \bot\end{array}\right.\right].$$

[$?x$] equals [T] if [x] equals [T] and [$?x$] equals [$\bot$] otherwise. The following theorem states some properties that relate order and implication:

[Map lemma
L7.8.1$_{134}$: $\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle \mathcal{A} \rightarrow \mathcal{B} \vdash ?\mathcal{A} \preceq ?\mathcal{B}$;
L7.8.2$_{134}$: $\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle \mathcal{A} \rightarrow \mathcal{B} \vdash ?\mathcal{A} \preceq \mathcal{B}$;
L7.8.3$_{135}$: $\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle ?\mathcal{A} \preceq ?\mathcal{B} \vdash \mathcal{A} \rightarrow \mathcal{B}$;
L7.8.4$_{135}$: $\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle \mathcal{A} \rightarrow \mathcal{B}; \neg\mathcal{A} \rightarrow \neg\mathcal{B} \vdash \approx\mathcal{A} \preceq \approx\mathcal{B}$;
L7.8.5$_{135}$: $\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle \approx\mathcal{A} \preceq \approx\mathcal{B} \vdash \mathcal{A} \rightarrow \mathcal{B}$;
L7.8.6$_{135}$: $\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle \approx\mathcal{A} \preceq \approx\mathcal{B} \vdash \neg\mathcal{A} \rightarrow \neg\mathcal{B}$;
L7.8.7$_{135}$: $\langle \mathcal{A},\mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle \mathcal{A} \rightarrow \mathcal{B} \preceq \mathcal{C} \vdash \mathcal{A} \tilde{\wedge} \mathcal{B} \preceq \mathcal{A} \tilde{\wedge} \mathcal{C}$;
L7.8.8$_{135}$: $\langle \mathcal{A},\mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle \mathcal{A} \tilde{\wedge} \mathcal{B} \preceq \mathcal{A} \tilde{\wedge} \mathcal{C} \vdash \mathcal{A} \rightarrow \mathcal{B} \preceq \mathcal{C}$]*

7.9 Monotonicity

[Monotonicity] expresses Lemma 2.17.1:

[Map rule Monotonicity: $\langle \mathcal{A},\mathcal{B},\mathcal{C}{\in}\mathcal{T}\rangle \mathcal{A} \preceq \mathcal{B} \vdash \mathcal{C}\,\text{'}\,\mathcal{A} \preceq \mathcal{C}\,\text{'}\,\mathcal{B}$]*

The following generalization of Monotonicity states that all constructs of $\lambda\mathsf{N}$ are monotonic:

[Map lemma Mono: $\langle \mathcal{A},\dot{\mathcal{A}},\mathcal{B},\dot{\mathcal{B}}{\in}\mathcal{T}\rangle \mathrm{Rep}(\mathcal{A},\dot{\mathcal{A}},\mathcal{B},\dot{\mathcal{B}}) \Vdash \mathcal{A} \preceq \dot{\mathcal{A}} \vdash \mathcal{B} \preceq \dot{\mathcal{B}}$]*

No formal proof of [Mono] will be given. [Mono] follows by structural induction in [$\mathcal{B}$] from the lemmas below.

[Map lemma
L7.9.1$_{137}$: $\langle b{\in}\mathcal{T}\rangle$ $b \preceq b$;
L7.9.2$_{137}$: $\langle x{\in}\mathcal{V}; a, \underline{a}{\in}\mathcal{T}\rangle$ $a \preceq \underline{a} \vdash \lambda x.a{\preceq}\lambda x.\underline{a}$;
L7.9.3$_{137}$: $\langle a,\underline{a},b,\underline{b}{\in}\mathcal{T}\rangle$ $a{\preceq}\underline{a} \vdash b{\preceq}\underline{b} \vdash a\,\text{'}\,b{\preceq}\underline{a}\,\text{'}\,\underline{b}$;
L7.9.4$_{137}$: $\langle a,\underline{a},b,\underline{b},c,\underline{c}{\in}\mathcal{T}\rangle$ $a{\preceq}\underline{a} \vdash b{\preceq}\underline{b} \vdash c{\preceq}\underline{c} \vdash \mathrm{if}(a,b,c){\preceq}\mathrm{if}(\underline{a},\underline{b},\underline{c})$]*

7.10 Monotonicity and deduction

Shorthand elimination depends on one meta-theorem per inference rule. For Monotonicity, the meta-theorem reads:

[Map lemma Ded:Monotonicity$_{137}$: $\langle a,b,c,h{\in}\mathcal{T}\rangle h \to a \preceq b \vdash h \to c\,\text{'}\,a \preceq c\,\text{'}\,b$]*

7.11 Minimality

[Minimality] expresses Lemma 2.18.3:

[Map rule Minimality: $\langle a,b{\in}\mathcal{T}\rangle a\,\text{'}\,b \preceq b \vdash \mathsf{Y}\,\text{'}\,a \preceq b$]*

7.12 Minimality and deduction

Shorthand elimination depends on one metatheorem per inference rule. For [Minimality], the metatheorem reads:

[Map lemma Ded:Minimality$_{138}$: $\langle a,b,h{\in}\mathcal{T}\rangle h \to a\,\text{'}\,b \preceq b \vdash h \to \mathsf{Y}\,\text{'}\,a \preceq b$]*

7.13 Peano induction

In $\lambda\mathsf{N}$ and MT, induction theorems are proved using Minimality. As an example of that [Peano] below states the principle of Peano induction.

It is possible to develop quite a lot of Peano arithmetic in $\lambda\mathsf{N}$, but the development is limited by the absence of quantifiers. Furthermore, the development is rather tedious.

MT has quantifiers and allows to develop all of the Peano arithmetic (and all of ZFC set theory, for that matter). In Section 9.13, the theorem of transfinite induction, [Transfinite], is proved using Minimality.

The natural numbers are represented as in Section 3.9, and Section 3.11. In particular, $\left[\,\tilde{\mathbf{N}}\,\text{'}\,x\,\right]$ equals [T] if [x] liberally represents a natural number and equals [$\perp$] otherwise. Section 3.9 states a recursive definition of $\left[\,\tilde{\mathbf{N}}\,\right]$. A more careful definition that uses [Y] instead of recursion reads:

$$\left[\,\boxed{\tilde{\mathbf{N}}} \doteq \mathsf{Y}\,\text{'}\,\hat{\mathbf{N}}\,\right]^{*} \text{ where}$$

$$\left[\,\boxed{\hat{\mathbf{N}}} \doteq \lambda f.\lambda x.\text{if}(x,\mathsf{T},f\,\text{'}(x^{-}))\,\right]^{*}.$$

The principle of Peano induction reads:

[Map lemma Peano$_{85}$: $\langle x{\in}\mathcal{V};\mathcal{A}{\in}\mathcal{T}\rangle\text{notfree}(x,\mathcal{A}) \Vdash$
$\mathcal{A}\,\text{'}\,0 \vdash \neg x \to \mathcal{A}\,\text{'}(x^{-}) \to \mathcal{A}\,\text{'}\,x \vdash \tilde{\mathbf{N}}\,\text{'}\,x \to \mathcal{A}\,\text{'}\,x$]*

The conclusion of [Peano] says that [$\mathcal{A}\,\text{'}\,x$] is true for all liberal representations of natural numbers. The first premise says that [$\mathcal{A}\,\text{'}\,x$] is true for all [x] that represent zero (there is only one such [x], namely the ur-element). The second premise has form

$$[\, \neg x \to \mathcal{A}\text{'}(x^-) \to \mathcal{A}\text{'}\, x \,].$$

That premise says that for all non-zero [x], [$\mathcal{A}$'(x^-)] implies [$\mathcal{A}$' x]. That [x] is non-zero is expressed as [$\neg x$]. This is legal but may be seen as abuse of notation because [$\neg$] is normally used on truth values.

Note that if the second premise was stated as

$$[\, \mathcal{A}\text{'}\, x \to \mathcal{A}\text{'}(x^+) \,]$$

then the premises would merely ensure that [$\mathcal{A}$' x] was true for all canonical representations of natural numbers.

It is possible to prove similar induction theorems for many other sets of premises. As an example, it is still possible to prove Peano if the second premise is replaced by

$$[\, \tilde{\mathbf{N}}\text{'}\, x \to \neg x \to \mathcal{A}\text{'}(x^-) \to \mathcal{A}\text{'}\, x \,].$$

However, a proof of [Peano] suffices to show how to prove induction theorems from Minimality.

The proof of [Peano] uses two trivial auxiliary lemmas:

[Map lemma

L7.13.1$_{138}$:	$\langle x{\in}\mathcal{V}; \mathcal{A}{\in}\mathcal{T}\rangle \mathrm{notfree}(x,\mathcal{A}) \Vdash \mathcal{A}\text{'}0 \vdash \neg x \to \mathcal{A}\text{'}(x^-) \to \mathcal{A}\text{'}x \vdash$ $\mathrm{if}(x, \mathsf{T}, ?\mathcal{A}\text{'}(x^-)) \preceq ?\mathcal{A}\text{'}\, x$	;
L7.13.2$_{138}$:	$\langle \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle \mathcal{A} \preceq ?\mathcal{B} \vdash \mathcal{A} \to \mathcal{B}$	]*

A more interesting proof is that of [Peano]:

[The Map proof of Peano reads

L1:	Premise ≫	$\mathcal{A}\text{'}\,0$	;
L2:	Premise ≫	$\neg x \to \mathcal{A}\text{'}(x^-) \to \mathcal{A}\text{'}\, x$	;
L3:	L7.13.1 ⊳ L1 ⊳ L2 ≫	$\mathrm{if}(x, \mathsf{T}, ?\mathcal{A}\text{'}(x^-)) \preceq ?\mathcal{A}\text{'}\, x$	;
L4:	Mono ⊳ L3 ≫	$\lambda x.\mathrm{if}(x, \mathsf{T}, ?\mathcal{A}\text{'}(x^-)) \preceq \lambda x.?\mathcal{A}\text{'}\, x$	;
L5:	Replace' ⊳ $\bar{\mathcal{R}}$ ⊳ L4 ≫	$\hat{\mathbf{N}}\text{'}\, \lambda x.?\mathcal{A}\text{'}\, x \preceq \lambda x.?\mathcal{A}\text{'}\, x$	;
L6:	Minimality ⊳ L5 ≫	$\tilde{\mathbf{N}} \preceq \lambda x.?\mathcal{A}\text{'}\, x$	;
L7:	Mono ⊳ L6 ≫	$\tilde{\mathbf{N}}\text{'}\, x \preceq (\lambda x.?\mathcal{A}\text{'}\, x)\text{'}\, x$	;
L8:	Replace' ⊳ $\bar{\mathcal{R}}$ ⊳ L7 ≫	$\tilde{\mathbf{N}}\text{'}\, x \preceq ?\mathcal{A}\text{'}\, x$	;
L9:	L7.13.2 ⊳ L8 ≫	$\tilde{\mathbf{N}}\text{'}\, x \to \mathcal{A}\text{'}\, x$	]*

The crucial step is in Line [L6] where [Minimality] is used to infer [$\tilde{\mathbf{N}} \preceq \lambda x.?\mathcal{A}\text{'}\, x$] from [$\hat{\mathbf{N}}\text{'}\, \lambda x.?\mathcal{A}\text{'}\, x \preceq \lambda x.?\mathcal{A}\text{'}\, x$].

Chapter 8

Presentation of MT in ZFC+SI

8.1 Introduction

Chapter 8 presents MT inside the framework of set theory. The presentation will be kept informal.

To construct a model of MT inside set theory one has to assert axiom SI which says that there exists a strongly inaccessible ordinal. This is why MT is presented in "ZFC+SI" rather than just "ZFC". No knowledge of inaccessible ordinals is needed to follow the exposition, however.

All terms and formulas of this chapter are terms and formulas of ZFC+SI. (ZFC+SI has the same terms and formulas as ZFC). All lemmas in this chapter are lemmas of ZFC+SI. We shall not prove the lemmas stated in this chapter as the purpose of this chapter merely is to give an intuitive understanding of MT.

The definitions in Chapter 2 are in effect here in Chapter 8.

8.2 The syntax of MT

As mentioned in Section 4.4, MT has two more constructs than $\lambda\mathsf{N}$:

$$[\,\mathcal{T} ::= \varepsilon\mathcal{T}; \mathsf{E}\mathcal{T}\,]^{*}.$$

The definition above continues the definitions of $[\,\mathcal{T}\,]$ in Section 2.4 and Section 4.3. The effective definition of $[\,\mathcal{T}\,]$ reads:

$$[\,\mathcal{T} ::= \mathcal{V}; \mathsf{N}; \lambda\mathcal{V}.\mathcal{T}; \mathcal{T}\,\text{'}\,\mathcal{T}; \mathrm{if}(\mathcal{T},\mathcal{T},\mathcal{T}); \varepsilon\mathcal{T}; \mathsf{E}\mathcal{T}\,].$$

8.3 The intuitive meaning of [$\mathsf{E}x$]

The intuitive meaning of [N], [$\lambda x.y$], [$x\,'\,y$], and [$\text{if}(x,y,z)$] are as in Section 2.5. The intuitive meaning of [$\boxed{\mathsf{E}x}$] is that for all [x],

$$\begin{array}{lcll} [\,\mathsf{E}x & \equiv & \mathsf{N} &] \quad \text{if } [\,x\,'\,y \equiv \mathsf{N}\,] \text{ for some } [\,y\,] \\ [\,\mathsf{E}x & \equiv & \bot &] \quad \text{otherwise} \end{array}$$

In other words, [$\mathsf{E}x$] expresses a primitive notion of existence.

Let $\lambda\mathsf{N}{+}\mathsf{E}$ denote $\lambda\mathsf{N}$ extended with [$\mathsf{E}x$] but not with [εx]. In $\lambda\mathsf{N}{+}\mathsf{E}$, [$\mathsf{E}x$] is a computable construct (but not a very useful one). In $\lambda\mathsf{N}{+}\mathsf{E}$, the trisection procedure has to be modified thus: When [$\mathcal{R}(\mathsf{E}x)$] is to be computed, compute [$\mathcal{R}(x\,'\,y)$] for all terms [y] in parallel. There are countably many terms, so one has to arrange that all the countably many computations receive a fair share of the cpu-time spent on computing [$\mathcal{R}(\mathsf{E}x)$].

In $\lambda\mathsf{N}{+}\mathsf{E}$, [$\mathsf{E}x \equiv \mathsf{N}$] if and only if [$x\,'\,y \equiv \mathsf{N}$] for some term [y]. In MT, however, [$\mathsf{E}x \equiv \mathsf{N}$] if and only if [$x\,'\,y \equiv \mathsf{N}$] for some element [y] of whatever model of MT is used.

The definition

$$\Big[\,\boxed{\mathsf{E}x{:}\,y} \doteq \mathsf{E}\lambda x.y\,\Big]^*$$

allows to use [E] as a quantifier: [$\mathsf{E}x{:}\,\mathcal{A}$] is true if there exists an [x] for which [$\mathcal{A}$] is true.

8.4 Axioms about [$\mathsf{E}x$]

In MT, the axioms about [$\mathsf{E}x$] are needed to prove some fundamental theorems about classicality, but otherwise they are probably of limited interest to the working mathematician. The axioms are:

$$\begin{array}{lll} [\,\text{Map rule} & & \\ \;\text{Existence:} & \mathsf{ET} \equiv \mathsf{T} & ; \\ \;\text{NonExistence:} & \mathsf{E}\bot \equiv \bot & ; \\ \;\text{ImpliedExistence:} & \mathsf{E}(x \circ y) \to \mathsf{E}x & ; \\ \;\text{TruthExistence:} & \mathsf{E}x \equiv \mathsf{E}(? \circ x) &]^* \end{array}$$

where

$$\begin{array}{lcll} [\,\boxed{x \circ y} & \doteq & \lambda z.x\,'(y\,'\,z) &]^* \\ [\,\boxed{?} & \doteq & \lambda x.\text{if}(x,\mathsf{T},\bot) &]^* \end{array}$$

[Existence] and [ImpliedExistence] allow to prove existence constructively and [NonExistence] allows to prove non-existence. [TruthExistence] states that [$\mathsf{E}x$] merely depends on whether or not [$x\,'\,y$] equals [T].

8.5 The intuitive meaning of [εx]

Suppose [L] is a minimal, transfinite universe, i.e. a set of maps that satisfies a property which is stated in Section 8.18. We shall refer to elements of [L] as *classical* maps. The intended meaning of [εx] is that it is a choice construct with the following properties:

- If [$x\,'\,y \equiv \bot$] for some classical [y] then [$\varepsilon x \equiv \bot$].
- If [$x\,'\,y \not\equiv \bot$] for all classical [y] then [$\varepsilon x \in L$].
- If [$x\,'\,y \not\equiv \bot$] for all classical [y] and [$x\,'\,y \equiv N$] for some classical [y] then [$x\,'\,\varepsilon x \equiv N$].

8.6 Classicality

Section 8.18 defines the notion of a transfinite universe and thereby defines what it means to be classical. To get an intuitive understanding of the concept, however, consider the following example. Let

[$f \equiv \lambda x.x\,'\,1\,'\,2\,'\,3 \wedge x\,'\,4\,'\,5$].

The value of [$f\,'\,x$] can be [T] or [F] or [$\bot$]. For that reason we shall say that the *non-classical outer range* of [f] is [$\{\mathsf{T},\mathsf{F},\bot\}$]. Furthermore, we shall refer to [$\{\mathsf{T},\mathsf{F},\bot\}$] and any superset of [$\{\mathsf{T},\mathsf{F},\bot\}$] as a *non-classical outer hull* of [f]. Hence, [S] is a non-classical outer hull of a function [g] if [$g\,'\,x \in S$] for all maps [x]. Furthermore, the outer range of a function [g] is the minimal non-classical outer hull of [g].

The value of [$f\,'\,x$] depends on the value of [$x\,'\,1\,'\,2\,'\,3$] and [$x\,'\,4\,'\,5$]. On the contrary, [$f\,'\,x$] is independent of the value of e.g. [$x\,'\,6\,'\,7$]. For that reason we shall say that [f] depends on [$\langle 1,2,3\rangle$] and [$\langle 4,5\rangle$] and is independent of [$\langle 6,7\rangle$].

[f] is independent of all tuples outside [$\{1,2,3,4,5\}^{<\omega}$] (for the sake of clarity we ignore some messy details related to monotonicity here). For that reason we refer to [$\{1,2,3,4,5\}$] as a *non-classical inner hull* of [f]. Any superset of a non-classical inner hull is also an inner hull. If a function [g] has a minimal non-classical inner hull then we refer to that hull as the *non-classical inner range* of [g].

The value of [$(\lambda u.u)\,'\,x$] depends on the value of [$x\,'\,m_1\,'\ldots'\,m_n$] for all maps [$m_1,\ldots,m_n$]. For that reason the only inner hull of [$\lambda u.u$] is the set [M] of all maps. Since [M] is minimal among all inner hulls of [$\lambda u.u$], [$\lambda u.u$] has an inner range. That inner range is of course [M].

classical map
non-classical outer range
non-classical outer hull
non-classical inner hull
non-classical inner range

The key property of classicality is that a map [g] is classical it it has an inner hull of limited size. There are some technicalities, however.

We shall now introduce classical versions of the above concepts. To give illustrative examples it is necessary to reveal that [$\perp$] is non-classical:

$$[\,\perp \notin L\,].$$

Furthermore, [$\mathsf{T}, \mathsf{F}, 0, 1, 2, \ldots$] are classical:

$$[\,\mathsf{T}, \mathsf{F}, 0, 1, 2, \ldots \in L\,].$$

Finally, if [x] and [y] are classical then [$x\,'\,y$] is classical:

$$[\,\forall x, y{\in}L{:}\, x\,'\,y \in L\,].$$

If [x] is classical then [$x\,'\,1\,'\,2\,'\,3$] and [$x\,'\,4\,'\,5$] are classical according to the properties stated above. Hence, if [x] is classical then [$x\,'\,1\,'\,2\,'\,3$] and [$x\,'\,4\,'\,5$] differ from [$\perp$] (since [$\perp$] is non-classical). So, if [x] is classical then [$f\,'\,x$] differs from [$\perp$] (where [$f \equiv \lambda x.x\,'\,1\,'\,2\,'\,3 \wedge x\,'\,4\,'\,5$] was defined in the beginning of the present section). This allows to conclude that if [x] is classical then [$f\,'\,x$] equals [T] or [F]. For that reason we shall refer to [$\{\mathsf{T}, \mathsf{F}\}$] as the *classical outer range* of [f]. We refer to any superset of [$\{\mathsf{T}, \mathsf{F}\}$] as a *classical outer hull* of [f].

Next step is to introduce the *classical inner hull*:

If [$p \subseteq M$] we shall say that [u] and [v] are *equal on* [p], written [$u \sim_p v$] if

$$[\,\forall x \in p^{<\omega}{:}\, u\,''\,x \approx v\,''\,x\,].$$

Extensionality says that two maps are equal if they are equal on [M]:

$$[\,x \equiv y \Leftrightarrow x \sim_M y\,].$$

We shall say that two maps are *classicaly equal*, written [$\boxed{x \sim y}$], if they are equal on [L]:

$$[\,x \sim y \Leftrightarrow x \sim_L y\,].$$

Finally, we shall say that [p] is a classical inner hull of [g] if

$$[\,\forall u, v{\in}L{:}\, (u \sim_p v \Rightarrow g\,'\,u \sim g\,'\,v)\,].$$

In other words: [p] is a classical inner hull of [g] if [$g\,'\,u$] is independent (up to [$x \sim y$]) of tuples outside [$p^{<\omega}$].

The definition of classicality is recursive:

classical outer range
classical outer hull
classical inner hull
equal on [p]
classicaly equal

(1) [T] is classical

(2) If [$g\,'\,x$] is classical for all classical [x] and if there exists a set [p] of classical maps such that [p] is of limited size and such that [p] is a classical inner range of [g], then [g] is classical.

The only problem in the definition above is the notion of "limited size". A set is of "limited size" if its cardinality is less than [σ], where [σ] is the inaccessible ordinal used for constructing the model of [Map]. However, axioms and definitions in [Map] cannot refer directly to [σ]. Fortunately, the following holds:

(3) The outer range of a classical [p] is of limited size.

Therefore, one can reformulate (2) thus:

(4) If [$g\,'\,x$] is classical for all classical [x] and if there exists a classical [p] such that the classical outer range of [p] is a classical inner hull of [g], then [g] is classical.

This is a reasonable definition of classicality except that with this definition one cannot prove the theorem that corresponds to the union set axiom of ZFC. To make the union set axiom provable we define the *second outer range* of [g] to be

$$[\ \{g\,'\,x\,'\,y | x, y \in L\}\ ].$$

The second outer range of a classical [p] is also of limited size, so we may reformulate (2) thus:

(5) If [$g\,'\,x$] is classical for all classical [x] and if there exists a classical [p] such that the second outer range of [p] is a classical inner hull of [g], then [g] is classical.

Property (1) and (5) above together with a minimality property constitutes the definition of classicality. A more formal definition is given in Section 8.18. Section 8.10 defines [ℓ] such that [$\ell\,'\,x \equiv \mathsf{T}$] if [x] is classical and [$\ell\,'\,x \equiv \bot$] otherwise.

8.7 Quantification

Define

$$\begin{array}{llll}
[& \boxed{\exists}\,x & \doteq & \approx\!(x\,'\,\varepsilon x) &] \\
[& \boxed{\exists x\colon y} & \doteq & \exists\lambda x.y &] \\
[& \boxed{\forall x\colon y} & \doteq & \neg\exists x\colon\neg y &]
\end{array}$$

The construct [$\exists x\colon y$] satisfies the following:

second outer range

- If $[\, y \equiv \bot \,]$ for some classical $[\, x \,]$ then $[\, \exists x \colon y \equiv \bot \,]$.
- If $[\, y \not\equiv \bot \,]$ for all classical $[\, x \,]$ and $[\, y \equiv \mathsf{T} \,]$ for some classical $[\, x \,]$ then $[\, \exists x \colon y \equiv \mathsf{T} \,]$.
- If $[\, y \not\equiv \bot \,]$ for all classical $[\, x \,]$ and $[\, y \not\equiv \mathsf{T} \,]$ for all classical $[\, x \,]$ then $[\, \exists x \colon y \equiv \mathsf{F} \,]$.

The construct $[\, \forall x \colon y \,]$ satisfies the following:

- If $[\, y \equiv \bot \,]$ for some classical $[\, x \,]$ then $[\, \forall x \colon y \equiv \bot \,]$.
- If $[\, y \equiv \mathsf{T} \,]$ for all classical $[\, x \,]$ then $[\, \forall x \colon y \equiv \mathsf{T} \,]$.
- If $[\, y \not\equiv \bot \,]$ for all classical $[\, x \,]$ and $[\, y \not\equiv \mathsf{T} \,]$ for some classical $[\, x \,]$ then $[\, \forall x \colon y \equiv \mathsf{F} \,]$.

8.8 Types and strictness

The term

$$\left[\, \boxed{!x} \doteq \mathrm{if}(x, \mathsf{T}, \mathsf{T}) \,\right]^*$$

is true if $[\, x \,]$ represents a Boolean value where $[\, \mathsf{T} \,]$ represents truth and all functions represent falsehood. In other words, $[\, !x \,]$ is true unless $[\, x \,]$ equals $[\, \bot \,]$. Terms like $[\, !x \,]$ will be referred to as *type predicates*. The statement

$$[\, !x \rightarrow !y \rightarrow !(x \wedge y) \,]$$

states that if $[\, x \,]$ and $[\, y \,]$ are Boolean then $[\, x \wedge y \,]$ is Boolean. Such statements will be referred to as *type rules*.

A reverse type rule like

$$[\, !(x \wedge y) \rightarrow !x \,]$$

states that if $[\, !(x \wedge y) \,]$ is Boolean then $[\, x \,]$ is Boolean. Such statements will be referred to as *strictness rules*. The statement above is equivalent to

$$[\, \bot \wedge y \equiv \bot \,]$$

which is what the literature normally regards as a strictness rule. In the present context, however, strictness rules are consistently presented as reverse type rules.

$\left[\, \boxed{\ell}\text{'}\, x \,\right]$ is another type predicate. Section 8.10 defines $[\, \ell \,]$ such that

$$\begin{array}{llll} [\, \ell\text{'}x & \equiv & \mathsf{N} \,] & \text{if } [\, x \,] \text{ is classical} \\ [\, \ell\text{'}x & \equiv & \bot \,] & \text{if } [\, x \,] \text{ is non-classical} \end{array}$$

type predicate
type rule
strictness rule

In addition to the type predicates [$!x$] and [$\ell\text{'}\,x$], the following predicates will be needed:

$$\begin{array}{lll} [\,\ell_1 a & \doteq & \forall x\colon \ell\text{'}(a\text{'}\,x) \quad]^* \\ [\,\ell_2 a & \doteq & \forall x\colon \forall y\colon \ell\text{'}(a\text{'}\,x\text{'}\,y) \quad]^* \\ [\,!_1 a & \doteq & \forall x\colon !(a\text{'}\,x) \quad]^* \\ [\,!_2 a & \doteq & \forall x\colon \forall y\colon !(a\text{'}\,x\text{'}\,y) \quad]^* \end{array}$$

8.9 Axioms about [εx]

[ε] satisfies the following type and strictness rules:

$$\begin{array}{l} [\,!_1 a \to \ell\text{'}\,\varepsilon x\colon a\text{'}\,x \quad] \\ [\,\ell\text{'}\,\varepsilon x\colon a\text{'}\,x \to !_1 a \quad] \end{array}$$

The [StrictTypeChoice] axiom combines these two rules into one:

[Map rule
 StrictTypeChoice: $\ell\text{'}\,\varepsilon x\colon a\text{'}\,x \equiv !_1 a$]*

[$\forall$] satisfies similar type and strictness rules:

$$\begin{array}{l} [\,!_1 a \to !\forall x\colon a\text{'}\,x \quad] \\ [\,!\forall x\colon a\text{'}\,x \to !_1 a \quad] \end{array}$$

The [StrictTypeAll] axiom combines these two rules into one:

[Map rule
 StrictTypeAll: $!\forall x\colon a\text{'}\,x \equiv !_1 a$]*

The [ElimAll] axiom says that if [$a\text{'}\,x$] is true for all classical [x] and if [b] is classical then [$a\text{'}\,b$] is true:

[Map rule
 ElimAll: $\forall x\colon a\text{'}\,x \to \ell\text{'}\,b \to a\text{'}\,b$]*

In a proof, [ElimAll] allows to *eliminate* a universal quantifier [$\forall x\colon a$] by replacing it with [$\ell\text{'}\,b \to a\text{'}\,b$].

The axioms above follow from the definition of [ε] stated in Section 8.5. The [Ackermann] axiom puts an additional requirement on [ε], namely that it satisfies *Ackermanns axiom* ([4], p.244). The [Ackermann] axiom reads:

[Map rule
 AckermannChoice: $\varepsilon x\colon a\text{'}\,x \equiv \varepsilon x\colon \ell\text{'}\,x \wedge a\text{'}\,x$]*

For any classical [x], [$\ell\text{'}\,x \wedge \mathcal{A}$] equals [T] if [$\mathcal{A}$] equals [T], [$\ell\text{'}\,x \wedge \mathcal{A}$] equals [$\perp$] if [$\mathcal{A}$] equals [$\perp$], and [$\ell\text{'}\,x \wedge \mathcal{A}$] equals [F] otherwise. Hence, for

eliminate
Ackermanns axiom

classical [x], the value of [$\ell\,'\,x \wedge \mathcal{A}$] merely depends on the label of the root of [$\mathcal{A}$] (c.f. Section 2.12). In other words, when [x] is classical, the value of [$\ell\,'\,x \wedge \mathcal{A}$] merely depends on which truth value [$\mathcal{A}$] represents. When [x] is non-classical, [$\ell\,'\,x \wedge \mathcal{A}$] equals [$\bot$] regardless of [$\mathcal{A}$]. Hence, the [Ackermann] axiom says that [$\varepsilon x{:}\, a\,'\,x$] merely depends on the truth value of [$a\,'\,x$] and merely depends on that value for classical [x]. This is somewhat analogous to Ackermanns axiom.

8.10 The definition of [ℓ]

Define

$$\left[\boxed{x =_p y} \doteq x \begin{cases} y \begin{cases} \mathsf{T} \\ \mathsf{F} \end{cases} \\ y \begin{cases} \mathsf{F} \\ \forall u{:}\, \forall v{:}\, x\,'(p\,'\,u\,'\,v) =_p y\,'(p\,'\,u\,'\,v) \end{cases} \end{cases} \right]^* .$$

[$x =_p y$] corresponds to [$x \sim_p y$] in Section 8.6 and $\left[x =_{M,\{p'u'v|u,v\in L\}} y \right]$ in Section 8.18. Next define

$$\left[\boxed{\mathsf{K}} \doteq \lambda x.\lambda y.x \right]^* .$$

The set

$$[\{ \mathsf{K}\,'\,u\,'\,v | u, v \in L \}]$$

equals [L], so [$x =_{\mathsf{K}} y$] corresponds to [$x \sim y$] in Section 8.6 and [$x =_{M,L} y$] in Section 8.18. Finally define

$$\begin{array}{lll} [\,\boxed{\ell} & \doteq & \mathsf{Y}\,'\,\lambda f.\lambda x.\mathrm{if}(x, \mathsf{T}, (\forall y{:}\, f\,'(x\,'\,y)) \wedge \\ & & \mathsf{E}p{:}\, f\,'\,p \wedge \forall u{:}\, \forall v{:}\, (u =_p v \Rightarrow x\,'\,u =_L y\,'\,v)) \quad]^* \end{array}$$

[$\ell\,'\,x$] equals [T] when [x] is classical and equals [$\bot$] otherwise.

8.11 Sets of terms and rules

The remainder of Chapter 8 explains the notion of classicality in more detail within the framework of ZFC+SI. Presentation of a model of MT is outside the scope of the present paper. The rest of Chapter 8 can be skipped without loss of continuity.

In continuation of Section 2.4 define

$$\begin{array}{lcll}
[\, x_i & \doteq & \langle 0, i\rangle &] \\
[\, \mathsf{N} & \doteq & \langle 1\rangle &] \\
[\, \lambda x.y & \doteq & \langle 2, x, y\rangle &] \\
[\, x\,\text{'}\,y & \doteq & \langle 3, x, y\rangle &] \\
[\, \mathrm{if}(x,y,z) & \doteq & \langle 4, x, y, z\rangle &] \\
[\, \varepsilon x & \doteq & \langle 5, x\rangle &] \\
[\, \mathsf{E}x & \doteq & \langle 6, x\rangle &] \\
[\, x \equiv y & \doteq & \langle 7, x, y\rangle &] \\
[\, x \vdash y & \doteq & \langle 8, x, y\rangle &] \\
[\, x \Vdash y & \doteq & \langle 9, x, y\rangle &] \\
[\, x ; y & \doteq & \langle 10, x, y\rangle &] \\
[\, \langle x{\in}\mathcal{V}\rangle y & \doteq & \langle 11, x, y\rangle &] \\
[\, \langle x{\in}\mathcal{T}\rangle y & \doteq & \langle 12, x, y\rangle &]
\end{array}$$

From now on, $\left[\,\boxed{\mathcal{T}}\,\right]$ denotes the smallest set that satisfies

$$\begin{array}{lcll}
[\, i \in \omega & \Rightarrow & x_i \in \mathcal{T} &] \\
[\, 0 = 0 & \Rightarrow & \mathsf{N} \in \mathcal{T} &] \\
[\, x \in \mathcal{V} \wedge y \in \mathcal{T} & \Rightarrow & \lambda x.y \in \mathcal{T} &] \\
[\, x \in \mathcal{T} \wedge y \in \mathcal{T} & \Rightarrow & x\,\text{'}\,y \in \mathcal{T} &] \\
[\, x \in \mathcal{T} \wedge y \in \mathcal{T} \wedge z \in \mathcal{T} & \Rightarrow & \mathrm{if}(x,y,z) \in \mathcal{T} &] \\
[\, x \in \mathcal{T} & \Rightarrow & \varepsilon x \in \mathcal{T} &] \\
[\, x \in \mathcal{T} & \Rightarrow & \mathsf{E}x \in \mathcal{T} &]
\end{array}$$

$[\, \mathcal{T}_{\mathrm{comp}} \,]$ denotes the set of computable terms. i.e. terms that do not contain the constructs $[\, \varepsilon x \,]$ and $[\, \mathsf{E}x \,]$. The set $\left[\,\boxed{\mathcal{R}}\,\right]$ of rules denotes the smallest set that satisfies:

$$\begin{array}{lcll}
[\, x \in \mathcal{T} \wedge y \in \mathcal{T} & \Rightarrow & (x \equiv y) \in \mathcal{R} &] \\
[\, x \in \mathcal{R} \wedge y \in \mathcal{R} & \Rightarrow & (x \vdash y) \in \mathcal{R} &] \\
[\, x \in \mathcal{T}_{\mathrm{comp}} \wedge y \in \mathcal{R} & \Rightarrow & (x \Vdash y) \in \mathcal{R} &] \\
[\, x \in \mathcal{R} \wedge y \in \mathcal{R} & \Rightarrow & (x ; y) \in \mathcal{R} &] \\
[\, x \in \mathcal{V} \wedge y \in \mathcal{R} & \Rightarrow & (\langle x{\in}\mathcal{V}\rangle y) \in \mathcal{R} &] \\
[\, x \in \mathcal{V} \wedge y \in \mathcal{R} & \Rightarrow & (\langle x{\in}\mathcal{T}\rangle y) \in \mathcal{R} &]
\end{array}$$

8.12 Functions

For all functions $[\, f \,]$ let $\left[\,\boxed{f^d}\,\right]$ and $\left[\,\boxed{f^r}\,\right]$ denote the *domain* and *range* of $[\, f \,]$, respectively. For all sets $[\, A \,]$ and $[\, B \,]$ let $\left[\,\boxed{A \rightarrow B}\,\right]$ denote the set of functions $[\, f \,]$ for which $[\, f^d \equiv A \,]$ and $[\, f^r \subseteq B \,]$. Let $[\, A \rightarrow B \,]$ be right associative such that $[\, A \rightarrow B \rightarrow C \,]$ means $[\, A \rightarrow (B \rightarrow C) \,]$. Let $[\, x(y) \,]$ be a binary operator which applies a function $[\, x \,]$ to an argument $[\, y \,]$.

domain
range

8.13 Nine-tuples

We shall represent models of MT by nine element tuples (*nine-tuples*)

$$[\ \langle \tilde{D}, \hat{D}, \tilde{\hat{\equiv}}, \tilde{N}, \hat{\lambda}, \tilde{A}, \tilde{I}, \tilde{\varepsilon}, \tilde{\mathsf{E}} \rangle\ ].$$

The intended meaning is as follows:

- [$\tilde{D}$] is a set. [$\tilde{D}$] denotes the domain of MT. Every closed term of MT has a *value* which belongs to [$\tilde{D}$].
- [$\hat{D} \subseteq (\mathcal{V} \to \tilde{D}) \to \tilde{D}$]. Every term of MT, open or closed, has an *interpretation* which belongs to [$\hat{D}$]. The interpretation [f] of a term [$\mathcal{A}$] has the property that [$f(e)$] is the value of [$\mathcal{A}$] when the variables [$x \in \mathcal{V}$] of MT have values [$e(x)$], respectively.
- [$\tilde{\hat{\equiv}} \subseteq \tilde{D} \times \tilde{D}$]. [$\langle x, y \rangle \in \tilde{\hat{\equiv}}$] is true when [x] and [y] denote the same map. In other words, [$\langle x, y \rangle \in \tilde{\hat{\equiv}}$] models [$x \equiv y$]. A model of MT is *normal* if [$\tilde{\hat{\equiv}}$] is the equality relation on [$\tilde{D}$].
- [$\tilde{N} \in \tilde{D}$]. [$\tilde{N}$] is the value of the term [N]. In other words, [$\tilde{N}$] models [N].
- [$\tilde{A} \in \tilde{D} \to \tilde{D} \to \tilde{D}$]. [$\tilde{A}(x)(y)$] denotes [x] applied to [y]. In other words, [$\tilde{A}(x)(y)$] models [$x\,'\,y$].
- [$\tilde{I} \in \tilde{D} \to \tilde{D} \to \tilde{D} \to \tilde{D}$]. [$\tilde{I}(x)(y)(z)$] models [$\mathrm{if}(x, y, z)$].
- [$\tilde{\varepsilon} \in \tilde{D} \to \tilde{D}$]. [$\tilde{\varepsilon}(x)$] models [εx].
- [$\tilde{\mathsf{E}} \in \tilde{D} \to \tilde{D}$]. [$\tilde{\mathsf{E}}(x)$] models [$\mathsf{E}x$].
- [$\hat{\lambda} \in \mathcal{V} \to \hat{D} \to \hat{D}$]. [$\hat{\lambda}(x)(y)$] models [$\lambda x.y$]. Note that [$\hat{\lambda}(x)(y)$] operates on interpretations rather than values.

nine-tuples
value
interpretation
normal

8.14 Operations on nine-tuples

For all nine-tuples $[\, M = \langle \tilde{D}, \hat{D}, \tilde{\equiv}, \tilde{N}, \hat{\lambda}, \tilde{A}, \tilde{I}, \tilde{\varepsilon}, \tilde{\mathsf{E}} \rangle \,]$ define

$$\begin{array}{llll}
[\, \boxed{\tilde{D}_M} & \doteq & \tilde{D} &] \\
[\, \boxed{\hat{D}_M} & \doteq & \hat{D} &] \\
[\, \boxed{x \mathrel{\tilde{\equiv}_M} y} & \doteq & \langle x, y \rangle \in \tilde{\equiv} &] \\
[\, \boxed{\tilde{N}_M} & \doteq & \tilde{N} &] \\
[\, \boxed{\hat{\lambda}_M x.y} & \doteq & \hat{\lambda}(x)(y) &] \\
[\, \boxed{\tilde{A}_M} & \doteq & \tilde{A} &] \\
[\, \boxed{\tilde{I}_M} & \doteq & \tilde{I} &] \\
[\, \boxed{\tilde{\varepsilon}_M} & \doteq & \tilde{\varepsilon} &] \\
[\, \boxed{\tilde{\mathsf{E}}_M} & \doteq & \tilde{\mathsf{E}} &]
\end{array}$$

Furthermore, for all nine-tuples $[\, M \,]$ define

$$\begin{array}{lll}
[\, \boxed{E_M} & \doteq & \mathcal{V} \rightarrow \tilde{D}_M \,] \\
[\, \boxed{\bar{D}_M} & \doteq & E_M \rightarrow \tilde{D}_M \,]
\end{array}$$

$[\, E_M \,]$ denotes the set of *environments*, i.e. the set of possible assignments of values to variables. For models $[\, M \,]$, $[\, \hat{D}_M \,]$ will satisfy $[\, \hat{D}_M \subseteq \bar{D}_M \,]$.

Let $[\, \hat{\equiv}_M \,]$ be the relation

$$\left[\boxed{x \mathrel{\hat{\equiv}_M} y} \Leftrightarrow \forall e \in E\colon x(e) \mathrel{\tilde{\equiv}} y(e) \right].$$

If $[\, x \,]$ and $[\, y \,]$ are interpretations of terms $[\, \mathcal{A} \,]$ and $[\, \mathcal{B} \,]$, then $[\, x \mathrel{\hat{\equiv}_M} y \,]$ is true if and only if $[\, \mathcal{A} \,]$ and $[\, \mathcal{B} \,]$ are equal in MT.

Let $[\, \hat{A}_M \in \hat{D}_M \rightarrow \hat{D}_M \rightarrow \bar{D}_M \,]$, $[\, \hat{N}_M \in \bar{D}_M \,]$, $[\, \hat{I}_M \in \hat{D}_M \rightarrow \hat{D}_M \rightarrow \hat{D}_M \rightarrow \bar{D}_M \,]$, $[\, \hat{\varepsilon}_M \in \hat{D}_M \rightarrow \bar{D}_M \,]$, $[\, \hat{\mathsf{E}}_M \in \hat{D}_M \rightarrow \bar{D}_M \,]$, and $[\, \mathcal{X}_M \in \mathcal{V} \rightarrow \bar{D}_M \,]$ satisfy the following for all $[\, x, y, z \in \hat{D} \,]$, $[\, e \in E \,]$, and $[\, v \in \mathcal{V} \,]$:

$$\begin{array}{lll}
[\, \boxed{\hat{N}_M(e)} & = & \tilde{N}_M \,] \\
[\, \boxed{\hat{A}_M(x)(y)(e)} & = & \tilde{A}_M(x(e))(y(e)) \,] \\
[\, \boxed{\hat{I}_M(x)(y)(z)(e)} & = & \tilde{I}_M(x(e))(y(e))(z(e)) \,] \\
[\, \boxed{\hat{\varepsilon}_M(x)(e)} & = & \hat{\varepsilon}_M(x(e)) \,] \\
[\, \boxed{\hat{\mathsf{E}}_M(x)(e)} & = & \hat{\mathsf{E}}_M(x(e)) \,] \\
[\, \boxed{\mathcal{X}_M(v)(e)} & = & e(v) \,]
\end{array}$$

---

environment

We shall represent models of MT by closed nine-tuple where a nine-tuple [M] is *closed* if the following holds:

$$\begin{array}{lll}
[\ \hat{D}_M & \subseteq & \bar{D}_M\ ] \\
[\ \hat{N}_M & \in & \hat{D}_M\ ] \\
[\ \hat{A}_M & \in & \hat{D}_M \to \hat{D}_M \to \hat{D}_M\ ] \\
[\ \hat{I}_M & \in & \hat{D}_M \to \hat{D}_M \to \hat{D}_M \to \hat{D}_M\ ] \\
[\ \hat{\varepsilon}_M & \in & \hat{D}_M \to \hat{D}_M\ ] \\
[\ \hat{\mathsf{E}}_M & \in & \hat{D}_M \to \hat{D}_M\ ] \\
[\ \mathcal{X}_M & \in & \mathcal{V} \to \hat{D}_M\ ]
\end{array}$$

8.15 Models

Assume there is a one-to-one relation between concrete variables of ZFC and concrete variables of MT. For all concrete variables [x] of MT let [$\bar{x}$] denote the corresponding concrete variable of ZFC.

In Chapter 8, concrete variables of ZFC are used as meta-variables of MT. For all variables [x] and rules [$\mathcal{A}$] of MT let [$\mathcal{A}_x$] denote the result of replacing all occurrences of [x] by [$\bar{x}$] in [$\mathcal{A}$] except that all [x] in side-conditions are replaced by [$\lceil\bar{x}\rceil$]. For all [x] in [$\mathcal{T}$], [$\lceil x\rceil$] denotes an element of [$\bar{\mathcal{T}}$] whose value equals the Gödel-number of [x] for some, suitable Gödel-numbering (recall that [$\bar{\mathcal{T}}$] is the set of elements of [$\mathcal{T}$] which has no free variables, where "variables" means "concrete variables of MT").

For all rules [$\mathcal{A}$] and all closed nine-tuples [M] we define the interpretation $\left[\ \boxed{\mathcal{I}_M(\mathcal{A})}\ \right]$ by structural recursion as follows:

$$\begin{array}{llcll}
[\ \forall x{\in}\mathcal{V}: & \mathcal{I}_M(x) & = & \mathcal{X}_M(x) &] \\
[& \mathcal{I}_M(\mathsf{N}) & = & \hat{N}_M &] \\
[\ \forall x{\in}\mathcal{V}\forall\mathcal{A}{\in}\mathcal{T}: & \mathcal{I}_M(\lambda x.\mathcal{A}) & = & \hat{\lambda}_M(x)(\mathcal{I}_M(\mathcal{A})) &] \\
[\ \forall\mathcal{A},\mathcal{B}{\in}\mathcal{T}: & \mathcal{I}_M(\mathcal{A}\,{}^{\backprime}\mathcal{B}) & = & \hat{A}_M(\mathcal{I}_M(\mathcal{A}))(\mathcal{I}_M(\mathcal{B})) &] \\
[\ \forall\mathcal{A},\mathcal{B},\mathcal{C}{\in}\mathcal{T}: & \mathcal{I}_M(\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C})) & = & \hat{I}_M(\mathcal{I}_M(\mathcal{A}))(\mathcal{I}_M(\mathcal{B}))(\mathcal{I}_M(\mathcal{C})) &] \\
[\ \forall\mathcal{A}{\in}\mathcal{T}: & \mathcal{I}_M(\varepsilon\mathcal{A}) & = & \hat{\varepsilon}_M(\mathcal{I}_M(\mathcal{A})) &] \\
[\ \forall\mathcal{A}{\in}\mathcal{T}: & \mathcal{I}_M(\mathsf{E}\mathcal{A}) & = & \hat{\mathsf{E}}_M(\mathcal{I}_M(\mathcal{A})) &] \\
[\ \forall\mathcal{A},\mathcal{B}{\in}\mathcal{T}: & \mathcal{I}_M(\mathcal{A}\equiv\mathcal{B}) & \Leftrightarrow & \mathcal{I}_M(\mathcal{A})\equiv_M\mathcal{I}_M(\mathcal{B}) &] \\
[\ \forall\mathcal{A},\mathcal{B}{\in}\mathcal{R}: & \mathcal{I}_M(\mathcal{A}\vdash\mathcal{B}) & \Leftrightarrow & (\mathcal{I}_M(\mathcal{A})\Rightarrow\mathcal{I}_M(\mathcal{B})) &] \\
[\ \forall\mathcal{A}{\in}\mathcal{T}\forall\mathcal{B}{\in}\mathcal{R}: & \mathcal{I}_M(\mathcal{A}\Vdash\mathcal{B}) & \Leftrightarrow & (\mathcal{I}_M(\mathcal{A})\hat{=}_M\hat{N}_M)\Rightarrow\mathcal{I}_M(\mathcal{B})) &] \\
[\ \forall\mathcal{A},\mathcal{B}{\in}\mathcal{R}: & \mathcal{I}_M(\mathcal{A};\mathcal{B}) & \Leftrightarrow & \mathcal{I}_M(\mathcal{A})\wedge\mathcal{I}_M(\mathcal{B}) &] \\
[\ \forall x{\in}\mathcal{V}\forall\mathcal{A}{\in}\mathcal{R}: & \mathcal{I}_M(\langle x{\in}\mathcal{V}\rangle\mathcal{A}) & \Leftrightarrow & \forall\bar{x}{\in}\mathcal{V}{:}\,\mathcal{I}_M(\mathcal{A}_x) &] \\
[\ \forall x{\in}\mathcal{T}\forall\mathcal{A}{\in}\mathcal{R}: & \mathcal{I}_M(\langle x{\in}\mathcal{T}\rangle\mathcal{A}) & \Leftrightarrow & \forall\bar{x}{\in}\mathcal{T}{:}\,\mathcal{I}_M(\mathcal{A}_x) &]
\end{array}$$

Recall that [Map] denotes the list of all rules of MT separated with semi-colons. A closed nine-tuple [M] is a *model* of MT if

[$\mathcal{I}_M(\mathrm{Map})$]

closed nine-tuple
model

is true.

8.16 Cardinality and power sets

A set [X] is said to by [Y]-*small*, written $\left[\boxed{X \prec Y} \right]$, if the cardinality of [X] is less than the cardinality of [Y]. As an example, the set [ω] of natural numbers is [$\mathbf{R}$]-small where [$\mathbf{R}$] denotes the set of real numbers. As another example, a set is *finite* if and only if it is [ω]-small.

For all sets [Z], $\left[\boxed{\mathcal{P}(Z)} \right]$ denotes the set of all subsets of [Z]; $\left[\boxed{\mathcal{P}^Y(Z)} \right]$ denotes the set of [Y]-small subsets of [Z].

Let [$\boxed{\sigma}$] be an inaccessible ordinal.

8.17 Strong QND

Let [e_M] denote the *empty environment* for which $\left[e_M(x) \equiv \tilde{N}_M \right]$ for all [$x \in \mathcal{V}$]. Let $\left[\tilde{\bot}_M \right]$ denote [$\mathcal{I}_M(\bot)(e_M)$]. Let $\left[\tilde{F} \right]$ denote [$\mathcal{I}_M(\lambda y.\lambda z.y\,'\,z)(e_M)$]. Define

$$\begin{array}{llll} [\ \tilde{D}_M^N & \equiv & \{x{\in}\tilde{D}_M | x \mathrel{\hat{\equiv}_M} \tilde{N}_M\} &] \\ [\ \tilde{D}_M^F & \equiv & \{x{\in}\tilde{D}_M | x \mathrel{\hat{\equiv}_M} \tilde{A}_M(\tilde{F}_M, x)\} &] \\ [\ \tilde{D}_M^\bot & \equiv & \{x{\in}\tilde{D}_M | x \mathrel{\hat{\equiv}_M} \tilde{\bot}_M\} &] \end{array}$$

A nine-tuple [M] will be said to satisfy *Strong QND* (*SQND*) if

$$\left[\tilde{D}_M^N \cup \tilde{D}_M^F \cup \tilde{D}_M^\bot = \tilde{D}_M \right].$$

A nine-tuple [M] is *classical* if it satisfies SQND.

8.18 Transfinite universes

For all sets [G] let $\left[\boxed{G^{<\omega}} \right]$ denote the set of tuples of elements of [G].

For all [$x \in D_M$] and for all $\left[y \in D_M^{<\omega} \right]$ define $\left[\boxed{A_M(x,y)} \right]$ such that

$$\left[A_M(x, \langle y_1, y_2, \ldots, y_n \rangle) = \tilde{A}_M(\cdots \tilde{A}_M(\tilde{A}_M(x, y_1), y_2) \cdots, y_n) \right].$$

For all $\left[x, y \in \tilde{D}_M \right]$ define

$$\left[x \approx_M y \Leftrightarrow (x \mathrel{\hat{\equiv}_M} \tilde{N}_N \Leftrightarrow y \mathrel{\hat{\equiv}_M} \tilde{N}_N) \land (x \mathrel{\hat{\equiv}_M} \tilde{\bot}_N \Leftrightarrow y \mathrel{\hat{\equiv}_M} \tilde{\bot}_N) \right].$$

---

[x]-small
finite
empty environment
Strong QND
SQND
classical

For all $[\, G \subseteq \tilde{D}_M \,]$ and $[\, x, y \in \tilde{D}_M \,]$ define

$$[\, x =_{M,G} y \Leftrightarrow \forall z{\in}G^{<\omega}\colon \tilde{A}_M(x,z) \approx_M \tilde{A}_M(y,z) \,].$$

A set $[\, L \subseteq \tilde{D}_M \,]$ is a *transfinite universe* if

$$\begin{array}{lll} [\, x \in L & \Leftrightarrow & x \equiv_M N_M \vee (\forall y{\in}L\colon \tilde{A}_M(x,y) \in L) \wedge \\ & & \exists Y{\in}\mathcal{P}^\sigma(L) \forall u, v{\in}L\colon (u{=}_{M,Y} v \Rightarrow \tilde{A}_M(x,u){=}_{M,L} \tilde{A}_M(x,v)) \end{array}$$

A set $[\, L \subseteq D_M \,]$ is a *minimal, transfinite universe* if $[\, L \,]$ is a transfinite universe and, furthermore, $[\, L \subseteq \tilde{L} \,]$ whenever

$$\begin{array}{lll} [\, x \in \tilde{L} & \Leftrightarrow & x \equiv_M N_M \vee (\forall y{\in}L\colon \tilde{A}_M(x,y) \in \tilde{L}) \wedge \\ & & \exists Y{\in}\mathcal{P}^\sigma(\tilde{L}) \forall u, v{\in}L\colon (u{=}_{M,Y} v \Rightarrow \tilde{A}_M(x,u){=}_{M,L} \tilde{A}_M(x,v)) \end{array}$$

A nine-tuple $[\, M \,]$ is *transfinite* if $[\, M \,]$ contains a minimal, transfinite universe.

transfinite universe
minimal, transfinite universe
transfinite

Chapter 9

Presentation of MT

9.1 Introduction

Chapter 9 presents MT as an axiomatic theory. The reader is assumed to have an intuitive understanding of MT from Chapter 8, so inference rules and axiom schemes are stated with little or no explanation. Rather, this chapter focuses on proof techniques in MT. MT is an extension of $\lambda\mathsf{N}$, and all proof techniques presented in Chapter 4 carry over to MT.

9.2 Overloading

Section 2.2 stated that in Chapter 2, [$\mathcal{A} \wedge \mathcal{B}$] stood for conjunction in ZFC. In this chapter, all terms and formulas (i.e. well-formed formulas) are terms and formulas, respectively, of MT unless otherwise noted. As an example, in this chapter, [$\mathcal{A} \wedge \mathcal{B}$] denotes conjunction in MT. [$\mathcal{A} \wedge \mathcal{B}$] was defined in Section 3.6.

9.3 The syntax of MT

The syntax of variables [$\mathcal{V}$], terms [$\mathcal{T}$], and formulas (i.e. well-formed formulas) [$\mathcal{F}$] of $\lambda\mathsf{N}$ reads:

$$\begin{array}{lcll}
[\,\mathcal{V} & ::= & x \mid y \mid z \mid \cdots &] \\
[\,\mathcal{T} & ::= & \mathcal{V} \mid \mathsf{N} \mid \lambda\mathcal{V}.\mathcal{T} \mid \mathcal{T}\,\text{'}\,\mathcal{T} \mid \mathrm{if}(\mathcal{T},\mathcal{T},\mathcal{T}) \mid \varepsilon\mathcal{T} \mid \mathsf{E}\mathcal{T} &] \\
[\,\mathcal{F} & ::= & \mathcal{T} \equiv \mathcal{T} &]
\end{array}$$

9.4 Pure existence

Recall the following definitions from Chapter 8:

$$
\begin{array}{llll}
[\,\mathsf{E}x\colon y & \doteq & \mathsf{E}\lambda x.y &]^* \\
[\,x \circ y & \doteq & \lambda z.x\,\text{'}(y\,\text{'}\,z) &]^* \\
[\,? & \doteq & \lambda x.\mathrm{if}(x,\mathsf{T},\bot) &]^*
\end{array}
$$

As mentioned in Section 8.4, the axioms about [$\mathsf{E}x$] read

[Map rule

Existence:	$\mathsf{ET} \equiv \mathsf{T}$	;
NonExistence:	$\mathsf{E}\bot \equiv \bot$	;
ImpliedExistence:	$\mathsf{E}(x \circ y) \to \mathsf{E}x$	;
TruthExistence:	$\mathsf{E}x \equiv \mathsf{E}(? \circ x)$	]*

The lemmas below express some intuitions about pure existence:

[Map lemma

$\mathrm{DomainPure}_{139}$:	$\langle x{\in}\mathcal{V}; \mathcal{A}{\in}\mathcal{T}\rangle$	$\mathsf{E}x\colon \mathcal{A} \equiv \mathsf{E}x\colon ?\mathcal{A}$	;
$\mathrm{RangePure}_{140}$:	$\langle x{\in}\mathcal{V}; \mathcal{A}{\in}\mathcal{T}\rangle$	$\mathsf{E}x\colon \mathcal{A} \equiv ?\mathsf{E}x\colon \mathcal{A}$	;
$\mathrm{ImplyPure}(\mathcal{C})_{140}$:	$\langle x{\in}\mathcal{V}; \mathcal{A},\mathcal{B},\dot{\mathcal{B}}{\in}\mathcal{T}\rangle$	$\dot{\mathcal{B}} \simeq \langle \mathcal{B} \mid x{:=}\mathcal{C}\rangle \Vdash$ $\mathsf{E}x\colon \mathcal{A} \vdash \mathcal{A} \to \dot{\mathcal{B}} \vdash \mathsf{E}x\colon \mathcal{B}$	;
$\mathrm{IntroPure}(\mathcal{C})_{140}$:	$\langle x{\in}\mathcal{V}; \mathcal{B},\dot{\mathcal{B}}{\in}\mathcal{T}\rangle$	$\dot{\mathcal{B}} \simeq \langle \mathcal{B} \mid x{:=}\mathcal{C}\rangle \Vdash \dot{\mathcal{B}} \to \mathsf{E}x\colon \mathcal{B}$	;
$\mathrm{ElimPure}_{140}$:	$\langle x{\in}\mathcal{V}; \mathcal{A},\mathcal{B}{\in}\mathcal{T}\rangle$	$\mathrm{notfree}(x;\mathcal{B}) \Vdash$ $\mathsf{E}x\colon \mathcal{A} \vdash \mathcal{A} \to \mathcal{B} \vdash \mathcal{B}$	]*

[DomainPure] says that [$\mathsf{E}x\colon\mathcal{A}$] merely tests whether or not [$\mathcal{A}$] equals [T]. In other words, [$\mathsf{E}x\colon\mathcal{A}$] ignores any other aspect of [$\mathcal{A}$]. Rules of this kind will be referred to as *domain rules*.

[RangePure] says that [$\mathsf{E}x\colon\mathcal{A}$] equals [T] or [$\bot$]. In other words, [$\mathsf{E}x\colon\mathcal{A}$] cannot generate any other values than these two. Rules of this kind will be referred to as *range rules*.

[ImplyPure] expresses Rule C (c.f. [9]); it allows to prove [$\mathsf{E}y\colon\mathcal{B}$] from [$\mathsf{E}x\colon\mathcal{A}$] if it is possible to construct a [y] that satisfies [$\mathcal{B}$] from [x] assuming that [x] satisfies [$\mathcal{A}$].

[IntroPure] presents a way to prove [$\mathsf{E}x\colon\mathcal{A}$]. Rules of this kind will be referred to as *introduction rules*.

[ElimPure] presents a way to prove a statement that does not contain [E] from one that does. Rules of this kind will be referred to as *elimination rules*.

domain rule
range rule
introduction rule
elimination rule

9.5 Axioms about $[\, \varepsilon x \,]$

Recall the following definitions from Chapter 8:

$$\begin{array}{lcll}
[\, \varepsilon x\colon y & \doteq & \varepsilon \lambda x.y &]^* \\
[\, \exists x\colon y & \doteq & \approx((\lambda x.y)\,'\,\varepsilon x\colon y) &]^* \\
[\, \forall x\colon y & \doteq & \neg\exists x\colon \neg y &]^* \\
[\, x =_p y & \doteq & x\begin{cases} y\begin{cases} \mathsf{T} \\ \mathsf{F} \end{cases} \\ y\begin{cases} \mathsf{F} \\ \forall u\colon \forall v\colon x\,'(p\,'\,u\,'\,v) =_p y\,'(p\,'\,u\,'\,v) \end{cases} \end{cases} &]^* \\
[\, \mathsf{K} & \doteq & \lambda x.\lambda y.x &]^* \\
[\, x \in_l p & \doteq & \forall u \forall v\colon (u =_p v \Rightarrow x\,'\,u =_{\mathsf{K}} x\,'\,v) &]^* \\
[\, \bar{\ell} & \doteq & \lambda f.\lambda x.\mathrm{if}(x, \mathsf{T}, (\forall y\colon f\,'(x\,'\,y)) \wedge \mathsf{E}p\colon f\,'\,p \wedge x \in_l p) &]^* \\
[\, \ell & \doteq & \mathsf{Y}\,'\,\bar{\ell} &]^* \\
[\, \ell_1 a & \doteq & \forall x\colon \ell\,'(a\,'\,x) &]^* \\
[\, \ell_2 a & \doteq & \forall x\colon \forall y\colon \ell\,'(a\,'\,x\,'\,y) &]^* \\
[\, !a & \doteq & \mathrm{if}(a, \mathsf{T}, \mathsf{T}) &]^* \\
[\, !_1 a & \doteq & \forall x\colon !\,'(a\,'\,x) &]^* \\
[\, !_2 a & \doteq & \forall x\colon \forall y\colon !\,'(a\,'\,x\,'\,y) &]^*
\end{array}$$

As mentioned in Section 8.9, the axioms about $[\, \varepsilon x \,]$ read

$$\begin{array}{lll}
[\,\text{Map rule} & & \\
\quad \text{ElimAll:} & \forall x\colon a\,'\,x \rightarrow \ell\,'\,b \rightarrow a\,'\,b & ; \\
\quad \text{AckermannChoice:} & \varepsilon x\colon a\,'\,x \equiv \varepsilon x\colon \ell\,'\,x \wedge a\,'\,x & ; \\
\quad \text{StrictTypeChoice:} & \ell\,'\,\varepsilon x\colon a\,'\,x \equiv !_1 a & ; \\
\quad \text{StrictTypeAll:} & !\forall x\colon a\,'\,x \equiv !_1 a &]^*
\end{array}$$

9.6 Domain and range of quantifiers

The domain rules below state that $[\, \varepsilon x\colon \mathcal{A} \,]$, $[\, \forall x\colon \mathcal{A} \,]$ and $[\, \exists x\colon \mathcal{A} \,]$ merely depend on the truth value of $[\, \mathcal{A} \,]$. The range rules below state that the value of $[\, \ell\,'\,x \,]$ is either $[\, \mathsf{T} \,]$ or $[\, \bot \,]$ whereas the value of $[\, \forall x\colon \mathcal{A} \,]$ or $[\, \exists x\colon \mathcal{A} \,]$ is either $[\, \mathsf{T} \,]$, $[\, \mathsf{F} \,]$ or $[\, \bot \,]$. The rules are handy e.g. when proving $[\, \neg\forall x\colon \neg\mathcal{A} \equiv \exists x\colon \mathcal{A} \,]$.

$$\begin{array}{lll}
[\,\text{Map lemma} & & \\
\quad \text{DomainChoice}_{141}\text{:} & \varepsilon x\colon a\,'\,x \equiv \varepsilon x\colon \approx a\,'\,x & ; \\
\quad \text{DomainExists}_{141}\text{:} & \exists x\colon a\,'\,x \equiv \exists x\colon \approx a\,'\,x & ; \\
\quad \text{DomainAll}_{141}\text{:} & \forall x\colon a\,'\,x \equiv \forall x\colon \approx a\,'\,x & ; \\
\quad \text{RangeEll}_{141}\text{:} & \ell\,'\,x \equiv ?\ell\,'\,x & ; \\
\quad \text{RangeExists}_{141}\text{:} & \exists x\colon a\,'\,x \equiv \approx\exists x\colon a\,'\,x & ; \\
\quad \text{RangeAll}_{141}\text{:} & \forall x\colon a\,'\,x \equiv \approx\forall x\colon a\,'\,x &]^*
\end{array}$$

9.7 Instantiation

Note that the variables in the lemmas of Section 9.6 are concrete variables. As an example, $[\ a\ ]$ and $[\ x\ ]$ in [DomainChoice] are concrete variables. One may use e.g.

$$[\ \text{Instantiation} \rhd \text{DomainChoice}\ ]$$

to conclude

$$[\ \varepsilon x\colon (\lambda u.u{+}2{=}5)\,\text{'}\,x \equiv \varepsilon x\colon {\approx}(\lambda u.u{+}2{=}5)\,\text{'}\,x\ ].$$

To make lemmas with concrete variables more useful, [Instantiation'] is introduced such that e.g.

$$[\ \text{Instantiation'} \rhd \text{DomainChoice}\ ]$$

allows to conclude e.g.

$$[\ \varepsilon x\colon x{+}2{=}5 \equiv \varepsilon x\colon {\approx}(x{+}2{=}5)\ ].$$

$[\ \text{Instantiation'} \rhd \mathcal{P}\ ]$ enhances $[\ \text{Instantiation} \rhd \mathcal{P}\ ]$ in that it allows reduction of redexes of form $[\ (\lambda x_1 \ldots \lambda x_n.a)\,\text{'}\,b_1\,\text{'}\cdots\text{'}\,b_n\ ]$ that occur in the premise $[\ \mathcal{P}\ ]$. Furthermore, [Instantiation'] is defined such that it allows forward and backward replacement in a context:

$$[\ \text{Instantiation'} \rhd \text{DomainChoice} \gg 2 + \varepsilon x\colon x{+}2{=}5 \equiv 2 + \varepsilon x\colon {\approx}(x{+}2{=}5)\ ]$$

$$[\ \text{Instantiation'} \rhd \text{DomainChoice} \gg 2 + \varepsilon x\colon {\approx}(x{+}2{=}5) \equiv 2 + \varepsilon x\colon x{+}2{=}5\ ]$$

Let $\left[\ \mathcal{A} \overset{1}{\rightarrow} \mathcal{B}\ \right]$ be true if $[\ \mathcal{B}\ ]$ arises from $[\ \mathcal{A}\ ]$ by renaming of bound variables and reduction of redexes of form $[\ (\lambda x_1 \ldots \lambda x_n.a)\,\text{'}\,b_1\,\text{'}\cdots\text{'}\,b_n\ ]$ that occur in $[\ \mathcal{A}\ ]$. The transitive closure of $\left[\ \mathcal{A} \overset{1}{\rightarrow} \mathcal{B}\ \right]$ is identical to usual beta-reduction [2]. Let $\left[\ \text{Instance}'(\mathcal{A},\mathcal{B},\dot{\mathcal{A}},\dot{\mathcal{B}})\ \right]$ be true if there exists variables $[\ z_1,\ldots z_n\ ]$ and terms $[\ \mathcal{C}_1,\ldots,\mathcal{C}_n,\mathcal{D}\ ]$ such that

$$\left[\ \mathcal{D}\,\text{'}\langle \mathcal{A} \mid z_1{:=}\mathcal{C}_1 \mid \cdots \mid z_n{:=}\mathcal{C}_n\rangle\,\text{'}\langle \mathcal{B} \mid z_1{:=}\mathcal{C}_1 \mid \cdots \mid z_n{:=}\mathcal{C}_n\rangle \overset{1}{\rightarrow} \dot{\mathcal{A}}\ \right] \text{ and}$$

$$\left[\ \mathcal{D}\,\text{'}\langle \mathcal{B} \mid z_1{:=}\mathcal{C}_1 \mid \cdots \mid z_n{:=}\mathcal{C}_n\rangle\,\text{'}\langle \mathcal{A} \mid z_1{:=}\mathcal{C}_1 \mid \cdots \mid z_n{:=}\mathcal{C}_n\rangle \overset{1}{\rightarrow} \dot{\mathcal{B}}\ \right].$$

[Instantiation'] reads:

$[\ \text{Map lemma Instantiation'}\colon \text{Instance}'(\mathcal{A},\mathcal{B},\dot{\mathcal{A}},\dot{\mathcal{B}}) \Vdash \mathcal{A} \equiv \mathcal{B} \vdash \dot{\mathcal{A}} \equiv \dot{\mathcal{B}}\ ]^*$

The definition

$$\left[\ \boxed{x^*} \doteq \text{Instantiation'} \rhd x\ \right]^*$$

gives a convenient notation for instantiating lemmas with concrete variables. As an example:

$$[\ \text{DomainChoice}^* \gg 2 + \varepsilon x\colon {\approx}(x + 2 = 5) \equiv 2 + \varepsilon x\colon x + 2 = 5\ ].$$

9.8 Negation of quantifiers

The definition of [$\forall x\colon a$] states that [$\forall x\colon a$] equals [$\neg\exists x\colon\neg a$]. The following three rules list the remaining combinations of quantification and negation.

[Map lemma

AllNot$_{142}$:	$\forall x\colon\neg a\,'\,x \equiv \neg\exists x\colon a\,'\,x$	;
NotAll$_{142}$:	$\neg\forall x\colon a\,'\,x \equiv \exists x\colon\neg a\,'\,x$	;
NotAllNot$_{142}$:	$\neg\forall x\colon\neg a\,'\,x \equiv \exists x\colon a\,'\,x$	]*

9.9 Type and strictness of quantifiers

[TypeChoice] below says that [$\varepsilon x\colon a\,'\,x$] is classical if [$a\,'\,x$] is Boolean for all classical [x]. [StrictChoice] says the opposite. Type and strictness rules for quantifiers are used remarkably often in proofs. In general, type checking chores take up a lot of proof lines. Type checking cannot be mechanized completely, but a mechanical proof system for [Map] should definitely include facilities for type checking.

[Map lemma

StrictTypeExists$_{142}$:	$!\exists x\colon a\,'\,x \equiv !_1 a$	;
TypeChoice$_{142}$:	$!_1 a \rightarrow \ell\,'\,\varepsilon x\colon a\,'\,x$	;
TypeExists$_{142}$:	$!_1 a \rightarrow !\exists x\colon a\,'\,x$	;
TypeAll$_{142}$:	$!_1 a \rightarrow !\forall x\colon a\,'\,x$	;
StrictChoice$_{142}$:	$\ell\,'\,\varepsilon x\colon a\,'\,x \rightarrow !_1 a$	;
StrictExists$_{142}$:	$!\exists x\colon a\,'\,x \rightarrow !_1 a$	;
StrictAll$_{143}$:	$!\forall x\colon a\,'\,x \rightarrow !_1 a$	;
StrictEll$_{143}$:	$!\ell\,'\,x \rightarrow \ell\,'\,x$	]*

The following conventions save space when formulating type rules:

[$\ell x\colon y$	$\doteq$	$\ell\,'\,x \rightarrow y$	]*
[$\ell_1 x\colon y$	$\doteq$	$\ell_1 x \rightarrow y$	]*
[$\ell_2 x\colon y$	$\doteq$	$\ell_2 x \rightarrow y$	]*
[$!x\colon y$	$\doteq$	$!x \rightarrow y$	]*
[$!_1 x\colon y$	$\doteq$	$!_1 x \rightarrow y$	]*
[$!_2 x\colon y$	$\doteq$	$!_2 x \rightarrow y$	]*

As an example, [TypeChoice] may be written

[$!_1 a\colon \ell\,'\,\varepsilon x\colon a\,'\,x$].

Furthermore, notation like

[$!_1 a, b, c\colon d$]

will be used as shorthand for

[$!_1 a\colon !_1 b\colon !_1 c\colon d$].

9.10 Elimination of quantifiers

[ElimExists] and [ElimExistsEll] below say that if [$a\,'\,x$] is true for some classical [x] then [$\varepsilon x\colon a\,'\,x$] is such an [x].

[ElimAll2], [ElimAll3], and [ElimAll4] correspond to two, three, and four applications of [ElimAll], respectively; they save a remarkable amount of proof lines.

[Map lemma
ElimExists$_{143}$: $\exists x\colon a\,'\,x \to a\,'\,\varepsilon x\colon a\,'\,x$;
ElimExistsEll$_{143}$: $\exists x\colon a\,'\,x \to \ell\,'\,\varepsilon x\colon a\,'\,x$;
ElimAll2$_{143}$: $\forall u \forall v\colon a\,'\,u\,'\,v \to \ell\,'\,b \to \ell\,'\,c \to a\,'\,b\,'\,c$;
ElimAll3$_{143}$: $\forall u \forall v \forall w\colon a\,'\,u\,'\,v\,'\,w \to$
$\ell\,'\,b \to \ell\,'\,c \to \ell\,'\,d \to a\,'\,b\,'\,c\,'\,d$;
ElimAll4$_{143}$: $\forall u \forall v \forall w \forall x\colon a\,'\,u\,'\,v\,'\,w\,'\,x \to$
$\ell\,'\,b \to \ell\,'\,c \to \ell\,'\,d \to \ell\,'\,e \to a\,'\,b\,'\,c\,'\,d\,'\,e$]*

9.11 Ackermann rules

[Ackermann] below is a formulation of Ackermanns original axiom ([4], p.244).

[AckermannExists] says something about the domain of [$\exists x\colon a$], so it would have been reasonable to call it [DomainExists]. That name, however, was used for another lemma in Section 9.6.

[AckermannAll] is analogous to [AckermannExists].

[Map lemma
AckermannExists$_{144}$: $\exists x\colon a\,'\,x \equiv \exists x\colon \ell\,'\,x \wedge a\,'\,x$;
AckermannAll$_{144}$: $\forall x\colon a\,'\,x \equiv \forall x\colon \ell\,'\,x \wedge a\,'\,x$;
Ackermann$_{144}$: $\forall x\colon (a\,'\,x \Leftrightarrow b\,'\,x) \to \varepsilon x\colon a\,'\,x \equiv \varepsilon x\colon b\,'\,x$]*

9.12 Introduction of quantifiers

[IntroExists] below allows to prove [$\exists x\colon a\,'\,x$] by giving a concrete example [b] of an object that satisfies [$a\,'\,x$].

The rule for introducing universal quantifiers ought to be named “IntroAll”, but the name [Gen] (for *generalization*) will be used instead. The rules [Gen2] and [Gen3] correspond to two and three applications of Gen, respectively, and save many proof lines.

[Map lemma
IntroExists$_{145}$: $!_1 a \to \ell\,'\,b \to a\,'\,b \to \exists x\colon a\,'\,x$;
Gen$_{145}$: $\langle x \in \mathcal{V}; \mathcal{A} \in \mathcal{T} \rangle \ell\,'\,x \to \mathcal{A} \vdash \forall x\colon \mathcal{A}$;
Gen2$_{145}$: $\langle x, y \in \mathcal{V}; \mathcal{A} \in \mathcal{T} \rangle \ell\,'\,x \to \ell\,'\,y \to \mathcal{A} \vdash \forall x\colon \forall y\colon \mathcal{A}$;
Gen3$_{145}$: $\langle x, y, z \in \mathcal{V}; \mathcal{A} \in \mathcal{T} \rangle \ell\,'\,x \to \ell\,'\,y \to \ell\,'\,z \to \mathcal{A} \vdash \forall x\colon \forall y\colon \forall z\colon \mathcal{A}$]*

generalization

9.13 Transfinite Induction

The definition of [ℓ] is recursive and, hence, gives rise to an induction principle:

[Map lemma
Transfinite$_{147}$: $\langle x, y{\in}\mathcal{V}; \mathcal{A}, \mathcal{A}_\mathsf{T}, \mathcal{A}_{xy}, \mathcal{A}_y{\in}\mathcal{T}\rangle$notfree$(y; \mathcal{A}) \wedge$
$\mathcal{A}_\mathsf{T} \simeq \langle \mathcal{A} \mid x{:=}\mathsf{T}\rangle \wedge$
$\mathcal{A}_{xy} \simeq \langle \mathcal{A} \mid x{:=}x\,\text{'}\,y\rangle \wedge$
$\mathcal{A}_y \simeq \langle \mathcal{A} \mid x{:=}y\rangle \Vdash$
$\mathcal{A}_\mathsf{T} \vdash$
$\neg x \to \forall y\colon \mathcal{A}_{xy} \to \mathsf{E}y\colon \mathcal{A}_y \wedge x \in_l y \to \mathcal{A} \vdash$
$\ell\,\text{'}\,x \to \mathcal{A}$]*

The proof of [Transfinite] does not use any of the axioms introduced here in Chapter 9. Rather, the proof relies on the minimality of fixed points.

[Transfinite] has two induction hypotheses, [$\forall y\colon \mathcal{A}_{xy}$] and [$\mathsf{E}y\colon \mathcal{A}_y{\wedge}x \in_l y$]. The former, which we shall refer to as the *outer* induction hypothesis, corresponds to the induction hypothesis of transfinite induction in ZFC. The latter, which we shall refer to as the *inner* induction hypothesis makes [Transfinite] more powerful than transfinite induction in ZFC. The outer hypothesis corresponds to well-foundedness in ZFC. The inner hypothesis has to do with limitation of size and avoidance of Burali-Fortis paradox.

The inner induction hypothesis is merely used in particularly intricate induction proofs, so it is convenient to formulate the following simplified lemma:

[Map lemma
Transfinite'$_{149}$: $\langle x, y{\in}\mathcal{V}; \mathcal{A}, \mathcal{A}_\mathsf{T}, \mathcal{A}_{xy}, \mathcal{A}_y{\in}\mathcal{T}\rangle$notfree$(y; \mathcal{A}) \wedge$
$\mathcal{A}_\mathsf{T} \simeq \langle \mathcal{A} \mid x{:=}\mathsf{T}\rangle \wedge \mathcal{A}_{xy} \simeq \langle \mathcal{A} \mid x{:=}x\,\text{'}\,y\rangle \Vdash$
$\mathcal{A}_\mathsf{T} \vdash \neg x \to \ell\,\text{'}\,x \to \forall y\colon \mathcal{A}_{xy} \to \mathcal{A} \vdash \forall x\colon \mathcal{A}$]*

[Transfinite'] is a special case of the following more general induction principle, which will be used when modeling the union set operator of ZFC:

[Map lemma
Transfinite"$_{148}$: $\langle x, y{\in}\mathcal{V}; \mathcal{A}, \mathcal{A}_\mathsf{T}, \mathcal{A}_{xy}, \mathcal{A}_y{\in}\mathcal{T}\rangle$notfree$(y; \mathcal{A}) \wedge$
$\mathcal{A}_\mathsf{T} \simeq \langle \mathcal{A} \mid x{:=}\mathsf{T}\rangle \wedge$
$\mathcal{A}_{xy} \simeq \langle \mathcal{A} \mid x{:=}x\,\text{'}\,y\rangle \wedge$
$\mathcal{A}_y \simeq \langle \mathcal{A} \mid x{:=}y\rangle \Vdash$
$\mathcal{A}_\mathsf{T} \vdash$
$\neg x \to \ell\,\text{'}\,x \to \forall y\colon \mathcal{A}_{xy} \to \mathsf{E}y\colon \ell\,\text{'}\,y \wedge \mathcal{A}_y \wedge x \in_l y \to \mathcal{A} \vdash$
$\forall x\colon \mathcal{A}$]*

9.14 Type and subtype rules

Another bunch of rules for type checking chores read:

outer induction hypothesis
inner induction hypothesis

[Map lemma

TypeT_{149}:	$\ell\,\text{'}\,\mathsf{T}$	;
TypeK_{149}:	$\ell_2\mathsf{K}$	;
$\mathrm{TypeIfBLL}_{149}$:	$!x{:}\,\ell y, z{:}\,\ell\,\text{'}\,\mathrm{if}(x,y,z)$	;
$\mathrm{SubtypeLB}_{149}$:	$\ell\,\text{'}\,x \to !x$	;
$\mathrm{SubtypeLL1}_{150}$:	$\ell\,\text{'}\,a \to \ell_1 a$	;
$\mathrm{TypeApply}_{150}$:	$\ell\,\text{'}\,a \to \ell\,\text{'}\,b \to \ell\,\text{'}(a\,\text{'}\,b)$	;
$\mathrm{SubtypeL1L2}_{150}$:	$\ell_1 a \to \ell_2 a$	;
$\mathrm{SubtypeL2B2}_{150}$:	$\ell_2 a \to !_2 a$	;
$\mathrm{SubtypeB1B}_{150}$:	$!_1 a \to !a$	;
$\mathrm{SubtypeB2B1}_{150}$:	$!_2 a \to !_1 a$	;
$\mathrm{SubtypeLL2}_{151}$:	$\ell\,\text{'}\,a \to \ell_2 a$	;
$\mathrm{SubtypeL1B}_{151}$:	$\ell_1 a \to !a$	]*

Chapter 10

Classicality

This chapter investigates classicality. Or, rather, it establishes lemmas that allow to prove [$\ell\,'\,a$] for a wide range of terms [a]. The chapter starts with a systematic treatment of the auxiliary concepts that appear in the definition of [ℓ].

10.1 Rules about [$x =_p y$]

[Map lemma		
TypeEquiv$_{152}$:	$\ell_2 p\colon \ell x, y\colon !(x =_p y)$	;
ElimEquiv$_{152}$:	$\ell u, v\colon a =_p b \to a\,'(p\,'\,u\,'\,v) =_p b\,'(p\,'\,u\,'\,v)$	;
IntroEquiv$_{153}$:	$\langle u, v {\in} \mathcal{V}; a, b, p {\in} \mathcal{T}\rangle \mathrm{notfree}(u, v; a, b, p) \Vdash$	
	$\neg a \vdash \neg b \vdash \ell u, v\colon a\,'(p\,'\,u\,'\,v) =_p b\,'(p\,'\,u\,'\,v) \vdash a =_p b$	;
RefEquiv$_{153}$:	$\ell_2 p\colon \ell a\colon a =_p a$	;
ComEquiv$_{153}$:	$\ell_2 p\colon \ell a, b\colon a =_p b \to b =_p a$	;
TransEquiv$_{156}$:	$\ell_2 p\colon \ell a, b, c\colon a =_p b \to b =_p c \to a =_p c$	]*

10.2 Lemmas about [$x \sim_p y$]

[$x \sim_p y$] is a strict version of [$x =_p y$]; lemmas about [$x \sim_p y$] tend to have fewer premises than lemmas about [$x =_p y$] (c.f. [ComEquivP] and [TransEquivP] below). The definitions and lemmas read:

$$\Big[\ \boxed{x \sim_p y} \doteq \ell\,'\,p \wedge \ell\,'\,x \wedge \ell\,'\,y \wedge x =_p y\ \Big]^*.$$

[Map lemma

TypeEquivP$_{157}$:	$\ell p, x, y \colon !x \sim_p y$	;
StrictEquivPX$_{157}$:	$!x \sim_p y \to \ell\text{'}x$	;
StrictEquivPY$_{157}$:	$!x \sim_p y \to \ell\text{'}y$	;
StrictEquivPP$_{157}$:	$!x \sim_p y \to \ell\text{'}p$	;
RefEquivP$_{157}$:	$\ell p, a \colon a \sim_p a$	;
ComEquivP$_{157}$:	$a \sim_p b \to b \sim_p a$	;
TransEquivP$_{158}$:	$a \sim_p b \to b \sim_p c \to a \sim_p c$	]*

10.3 Lemmas about $[\, x \sim y \,]$

$[\, x \sim y \,]$ is a strict version of $[\, x =_{\mathsf{K}} y \,]$. $[\, x \sim y \,]$ corresponds to equality of classical mathematics. Informally, if $[\, L \,]$ is the class of classical maps then the quotient $[\, L/\sim \,]$ resembles the universe of classical mathematics. See Chapter 11 for a model of ZFC in MT. See Section 10.6 for more on the quotient $[\, L/\sim \,]$.

$$\left[\boxed{x \sim y} \doteq \ell\text{'}x \wedge \ell\text{'}y \wedge x =_{\mathsf{K}} y \right]^*.$$

[Map lemma

TypeEquivK$_{158}$:	$\ell x, y \colon !x \sim y$	;
StrictEquivKX$_{158}$:	$!x \sim y \to \ell\text{'}x$	;
StrictEquivKY$_{158}$:	$!x \sim y \to \ell\text{'}y$	;
ElimEquivK$_{158}$:	$\ell u \colon a \sim b \to a\text{'}u \sim b\text{'}u$	;
RefEquivK$_{159}$:	$\ell a \colon a \sim a$	;
ComEquivK$_{159}$:	$a \sim b \to b \sim a$	;
TransEquivK$_{159}$:	$a \sim b \to b \sim c \to a \sim c$	]*

10.4 Lemmas about $[\, x \in_\ell p \,]$

$[\, x \in_\ell p \,]$ is a strict version of $[\, x \in_l p \,]$; it allows to state certain lemmas with fewer hypotheses than the corresponding lemmas about $[\, x \in_l p \,]$.

$[\, x \in_\ell p \,]$ and $[\, x \in_l p \,]$ have little to do with membership. The relations are neither reflexive nor irreflexive. Neither are they commutative nor anti-commutative; and they are not transitive. The character $[\, \in \,]$ was used in the name of $[\, x \in_\ell p \,]$ for the somewhat far fetched reason that membership does not necessarily have any of those five properties either. Furthermore, some rules stated later display a superficial analogy between membership and $[\, x \in_\ell p \,]$. The most notable one is [SubInEll] in Section 10.8 which says that if $[\, x \in_\ell p \,]$ and if $[\, p \,]$ is a subset of $[\, q \,]$ (in some sense) then $[\, x \in_\ell q \,]$. The definitions are lemmas read:

$$\left[\boxed{x \in_\ell p} \doteq \ell_1 x \wedge \ell\text{'}p \wedge x \in_l p \right]^*.$$

[Map lemma

TypeLim$_{160}$:	$\ell_1 x{:}\, \ell p{:}\, !x \in_l p$	;
TypeInEll$_{160}$:	$\ell_1 x{:}\, \ell p{:}\, !x \in_\ell p$	;
StrictInEllX$_{160}$:	$!x \in_\ell p \to \ell_1 x$	;
StrictInEllP$_{160}$:	$!x \in_\ell p \to \ell\,\text{'}\,p$	;
ElimInEll$_{160}$:	$x \in_\ell p \to u \sim_p v \to x\,\text{'}\,u \sim x\,\text{'}\,v$	;
IntroInEll$_{161}$:	$\langle u, v{\in}\mathcal{V}; x, p{\in}\mathcal{T}\rangle \text{notfree}(u, v; x, p) \Vdash$	
	$\ell\,\text{'}\,p \vdash u \sim_p v \to x\,\text{'}\,u \sim x\,\text{'}\,v \vdash x \in_\ell p$	;
IntroInEllT$_{161}$:	$\ell\,\text{'}\,p \to \mathsf{T} \in_\ell p$	]*

10.5 Lemmas about [ℓ]

We shall refer to [$\mathcal{I}x$] as the *inner range* of [x] (as opposed to the outer range, c.f. Section 10.7). All classical [x] have an inner range which is classical. The definitions are lemmas read:

[$\mathcal{I}x \doteq \varepsilon p{:}\, x \in_l p$]*.

[Map lemma

ElimEll$_{162}$:	$\ell\,\text{'}\,x \to x \in_\ell \mathcal{I}x$	;
IntroEll$_{163}$:	$x \in_\ell p \to \ell\,\text{'}\,x$	]*

10.6 Classical Extensionality

If [$f\,\text{'}\,x$] is classical for all classical [x], then [f] has the extensionality property

[$x \sim y \to f\,\text{'}\,x \sim f\,\text{'}\,y$].

We shall refer to this property as *classical extensionality* (c.f. Section 10.3 which stated that [$L/\sim$] is a classical universe).

[Ext] expresses classical extensionality. Classical extensionality depends on the monotonicity of [f] and the definition of classicality. Lemma [EllOrderA] indicates how monotonicity comes into play.

[Map lemma

IntroEquivK$_{163}$:	$\neg a \to \neg b \to \ell\,\text{'}\,a \to \forall x{:}\, a\,\text{'}\,x \sim b\,\text{'}\,x \to a \sim b$	;
EllOrderA$_{163}$:	$x \sim y \to \ell\,\text{'}(x \downarrow y)$	;
Ext$_{164}$:	$\ell_1 f{:}\, x \sim y \to f\,\text{'}\,x \sim f\,\text{'}\,y$	;
ExtBool$_{165}$:	$!_1 f{:}\, x \sim y \to f\,\text{'}\,x \Leftrightarrow f\,\text{'}\,y$	;
ExtBool'$_{165}$:	$!_1 f{:}\, x \sim y \to f\,\text{'}\,x \to f\,\text{'}\,y$	;
ExtBool"$_{165}$:	$!_1 f{:}\, x \sim y \to f\,\text{'}\,y \to f\,\text{'}\,x$	]*

There is a twist in the proof of [EllOrderA]. The proof requires transfinite induction, but the thing to prove by transfinite induction is

[$\forall x \forall y{:}\, (x \sim y \mathrel{\dot{\Rightarrow}} x \sim x \downarrow y$].

inner range
classical extensionality

Once the statement above is proved, [$\ell\text{'}(x \downarrow y)$] follows from [$x \sim x \downarrow y$] by strictness of [$\sim$].

10.7 Primitive membership

To investigate the nature of classicality, we need to introduce a primitive notion of sets and membership. Chapter 11 will define a more advanced notion [$x \in y$] of membership that models the membership relation of ZFC. For the present, we shall introduce a whole sequence [$x \in_1 y, x \in_2 y, x \in_3 y, \ldots$] of primitive membership relations:

$$\begin{array}{llll} [\, x \in_1 y & \doteq \exists s\colon & x \sim y\text{'}s &]^* \\ [\, \boxed{x \in_2 y} & \doteq \exists s, t\colon & x \sim y\text{'}s\text{'}t &]^* \\ [\, x \in_3 y & \doteq \exists s, t, u\colon & x \sim y\text{'}s\text{'}t\text{'}u &]^* \end{array}$$

In this paper we shall merely use [$x \in_2 y$], however.

We shall say that [x] belongs to the *first range* of [y] if [$x \in_1 y$]. The *second range* and so on are defined analogously. We shall sometimes refer to the second range as the *outer range*. When there is no risk of misunderstanding, we shall refer to the first range simply as the *range*.

The intuition is: The first and second range of any classical map [x] is a non-empty <u>set</u> of classical maps. The first range of [$\lambda x.x$] and the second range of [K] equals the <u>class</u> of all classical maps. The intuition can be made precise in the framework of Chapter 8 in that the ranges of classical maps are sets with *essential cardinality* lower than the inaccessible ordinal [σ] whereas the ranges of maps that are not classical can be of essential cardinality equal to [σ]. See [3] for definitions of essential cardinality.

If [$x \in_2 y$] then [$x \sim y\text{'}s_2(x,y)\text{'}t_2(x,y)$] where [$s_2(x,y)$] and [$t_2(x,y)$] are defined thus:

$$\begin{array}{lll} [\, \boxed{s_2(x,y)} & \doteq \varepsilon s\colon \exists t\colon x \sim y\text{'}s\text{'}t &]^* \\ [\, \boxed{t_2(x,y)} & \doteq \varepsilon t\colon x \sim y\text{'}s_2(x,y)\text{'}t &]^* \end{array}$$

The elementary lemmas about primitive membership read:

---

first range
second range
outer range
range
essential cardinality

[Map lemma

TypeInTwo$_{166}$:	$\ell x{:}\,\ell_2 y{:}\,!x \in_2 y$	;
StrictInTwoX$_{166}$:	$!x \in_2 y \to \ell\,\text{'}\,x$	;
StrictInTwoY$_{166}$:	$!x \in_2 y \to \ell_2 y$	;
ElimInTwo$_{166}$:	$x \in_2 y \to x \sim y\,\text{'}\,s_2(x,y)\,\text{'}\,t_2(x,y)$	;
ElimInTwoS$_{167}$:	$x \in_2 y \to \ell\,\text{'}\,s_2(x,y)$	;
ElimInTwoT$_{167}$:	$x \in_2 y \to \ell\,\text{'}\,t_2(x,y)$	;
IntroInTwo$_{167}$:	$\ell_2 y{:}\,\ell s,t{:}\,x \sim y\,\text{'}\,s\,\text{'}\,t \to x \in_2 y$	;
IntroInTwo'$_{167}$:	$\ell_2 y{:}\,\ell s,t{:}\,y\,\text{'}\,s\,\text{'}\,t \in_2 y$	;
IntroInTwoK$_{167}$:	$\ell x{:}\,x \in_2 \mathsf{K}$	;
IntroInTwoT$_{168}$:	$\mathsf{T} \in_2 \mathsf{T}$	;
ElimEquiv'$_{168}$:	$\ell x,y{:}\,x =_p y \to z \in_2 p \to x\,\text{'}\,z =_p y\,\text{'}\,z$	;
IntroEquiv'$_{168}$:	$\langle z{\in}\mathcal{V}; x,y,p{\in}\mathcal{T}\rangle \text{notfree}(z;x,y,p) \Vdash$ $\ell_2 p \vdash \neg x \vdash \neg y \vdash z \in_2 p \to x\,\text{'}\,z =_p y\,\text{'}\,z \vdash x =_p y$	;
ElimEquivP$_{168}$:	$x \sim_p y \to z \in_2 p \to x\,\text{'}\,z \sim_p y\,\text{'}\,z$	]*

10.8 Lemmas about [$x \subseteq_2 y$]

The subset relation [$x \subseteq_2 y$] is defined the obvious way:

$$[\, x \subseteq_2 y \quad \doteq \quad \forall z{:}\,(z \in_2 x \Rightarrow z \in_2 y) \quad]^*$$

[Map lemma

TypeSub$_{169}$:	$\ell_2 x,y{:}\,!x \subseteq_2 y$	;
StrictSubX$_{169}$:	$!x \subseteq_2 y \to \ell_2 x$	;
StrictSubY$_{169}$:	$!x \subseteq_2 y \to \ell_2 y$	;
ElimSub$_{169}$:	$p \subseteq_2 q \to x \in_2 p \to x \in_2 q$	;
IntroSub$_{170}$:	$\ell_2 p \vdash x \in_2 p \to x \in_2 q \vdash p \subseteq_2 q$	;
TransSub$_{170}$:	$p \subseteq_2 q \to q \subseteq_2 r \to p \subseteq_2 r$	;
IntroSubK$_{170}$:	$\ell_2 p \to p \subseteq_2 \mathsf{K}$	;
IntroKSub$_{170}$:	$\ell x,y{:}\,\mathsf{K}\,\text{'}(x\,\text{'}\,y) \subseteq_2 x$	;
SubEquiv$_{172}$:	$\ell x,y{:}\,p \subseteq_2 q \to x =_q y \to x =_p y$	;
SubEquivP$_{173}$:	$\ell p{:}\,p \subseteq_2 q \to x \sim_q y \to x \sim_p y$	;
SubEquivK$_{173}$:	$\ell p{:}\,x \sim y \to x \sim_p y$	;
SubInEll$_{173}$:	$\ell q{:}\,p \subseteq_2 q \to x \in_\ell p \to x \in_\ell q$	]*

10.9 Extensionality lemmas

One way to prove the classicality of a term [a] is to prove [$a \sim \underline{a}$] for some term [$\underline{a}$] and then conclude [$\ell\,\text{'}\,a$] by strictness of [$\sim$]. This seemingly round-about approach is convenient when proving the classicality of terms that contain lambdas (c.f. [ExtLambda] below). For an example of use, see the proofs in Section 10.10.

[Map lemma

ExtT$_{174}$:	$\mathsf{T} \sim \mathsf{T}$	;
ExtKApplyP$_{174}$:	$a \in_\ell p \to a \sim \underline{a} \to x \sim_p \underline{x} \to a\,\text{'}\,x \sim \underline{a}\,\text{'}\,\underline{x}$	;
ExtPApplyK$_{174}$:	$a \in_2 p \to x \sim_p \underline{x} \to a \sim \underline{a} \to x\,\text{'}\,a \sim_p \underline{x}\,\text{'}\,\underline{a}$	;
ExtKApplyK$_{174}$:	$a \sim \underline{a} \to b \sim \underline{b} \to a\,\text{'}\,b \sim \underline{a}\,\text{'}\,\underline{b}$	;
ExtIfBKK$_{175}$:	$x \Leftrightarrow \underline{x} \to b \sim \underline{b} \to c \sim \underline{c} \to \text{if}(x,b,c) \sim \text{if}(\underline{x},\underline{b},\underline{c})$	;
ExtIfPKK$_{175}$:	$x \sim_p \underline{x} \to b \sim \underline{b} \to c \sim \underline{c} \to \text{if}(x,b,c) \sim \text{if}(\underline{x},\underline{b},\underline{c})$	;
ExtIfKKK$_{175}$:	$a \sim \underline{a} \to b \sim \underline{b} \to c \sim \underline{c} \to \text{if}(a,b,c) \sim \text{if}(\underline{a},\underline{b},\underline{c})$	;
ExtLambda$_{175}$:	$\langle x, \underline{x} {\in} \mathcal{V}; a, \underline{a}, p {\in} \mathcal{T} \rangle \text{notfree}(x; \underline{a}) \wedge \text{notfree}(\underline{x}; a) \Vdash$ $\ell\,\text{'}\,p \vdash x \sim_p \underline{x} \to a \sim \underline{a} \vdash \lambda x.a \sim \lambda \underline{x}.\underline{a}$	]*

10.10 Elementary type lemmas

Some elementary type lemmas are stated in the following.

The proofs of [ExtKPairK], [TypePair], and [TypeCompose] are stated here instead of in the appendix because they illustrate how to use extensionality rules for proving classicality. Extensionality rules replace the monstrous metatheorem of "totality" in [5] where "total" is an older name for "classical".

[Map lemma

ExtKPairK$_{114}$:	$a \sim \underline{a} \to b \sim \underline{b} \to a :: b \sim \underline{a} :: \underline{b}$	;
TypePair$_{114}$:	$\ell a, b{:}\,\ell\,\text{'}(a :: b)$	;
ExtF$_{176}$:	$\mathsf{F} \sim \mathsf{F}$	;
TypeF$_{176}$:	$\ell\,\text{'}\,\mathsf{F}$	;
TypeKa$_{177}$:	$\ell a{:}\,\ell\,\text{'}(\mathsf{K}\,\text{'}\,a)$	;
TypeCompose$_{114}$:	$\ell_1 f{:}\, \ell a{:}\, \ell\,\text{'}(f \circ a)$	]*

[The Map proof of ExtKPairK reads

L1:	Hypothesis $\gg$	$a \sim \underline{a}$	;
L2:	Hypothesis $\gg$	$b \sim \underline{b}$	;
L3:	TypeT $\gg$	$\ell\,\text{'}\,\mathsf{T}$	;
L4:	Block $\gg$	Begin	;
L5:	Hypothesis $\gg$	$x \sim_{\mathsf{T}} \underline{x}$	;
L6:	ExtIfPKK $\unrhd$ L5 $\unrhd$ L1 $\unrhd$ L2 $\gg$	$\text{if}(x,a,b) \sim \text{if}(\underline{x},\underline{a},\underline{b})$	;
L7:	Block $\gg$	End	;
L8:	ExtLambda $\rhd$ L3 $\rhd$ L6 $\gg$	$\lambda x.\text{if}(x,a,b) \sim \lambda \underline{x}.\text{if}(\underline{x},\underline{a},\underline{b})$	;
L9:	Rename $\rhd$ L8 $\gg$	$a :: b \sim \underline{a} :: \underline{b}$	]*

[The Map proof of TypePair reads

L1:	Hypothesis $\gg$	$\ell\,\text{'}\,a$	;
L2:	Hypothesis $\gg$	$\ell\,\text{'}\,b$	;
L3:	RefEquivK $\unrhd$ L1 $\gg$	$a \sim a$	;
L4:	RefEquivK $\unrhd$ L2 $\gg$	$b \sim b$	;
L5:	ExtKPairK $\unrhd$ L3 $\unrhd$ L4 $\gg$	$a :: b \sim a :: b$	;
L6:	StrictEquivKX $\unrhd$ L5 $\gg$	$\ell\,\text{'}(a :: b)$	]*

[The Map proof of TypeCompose reads

L01:	Hypothesis $\gg$	$\ell_1 f$	;
L02:	Hypothesis $\gg$	$\ell ' a$	;
L03:	ElimEll $\trianglerighteq$ L2 $\gg$	$a \in_\ell \mathcal{I}a$	;
L04:	StrictInEllP $\trianglerighteq$ L3 $\gg$	$\ell ' \mathcal{I}a$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	$\quad x \sim_{\mathcal{I}a} \underline{x}$	;
L07:	ElimInEll $\trianglerighteq$ L3 $\trianglerighteq$ L6 $\gg$	$\quad a ' x \sim a ' \underline{x}$	;
L08:	Ext $\trianglerighteq$ L1 $\trianglerighteq$ L7 $\gg$	$\quad f '(a ' x) \sim f '(a ' \underline{x})$	;
L09:	Block $\gg$	End	;
L10:	ExtLambda $\triangleright$ L4 $\triangleright$ L8 $\gg$	$\lambda x.f '(a ' x) \sim \lambda \underline{x}.f '(a ' \underline{x})$	;
L11:	StrictEquivKX $\trianglerighteq$ L10 $\gg$	$\ell ' \lambda x.f '(a ' x)$	;
L12:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L11 $\gg$	$\ell '(f \circ a)$	]*

10.11 Transitive maps

In ZFC, a set [b] is *transitive* if [$x \in y$] and [$y \in b$] imply [$x \in b$]. The *transitive closure* of a set [a] is the minimal transitive set [b] for which [$a \subseteq b$].

We shall need a similar concept in [Map], but the transitive maps [p] we shall need are "more closed" than those of ZFC in that they are closed in two respects: they are closed under outer ranges in the sense that if [$x \in_2 p$] then [$x ' y \in_2 p$] and they are closed under inner ranges in the sense that if [$x \in_2 p$] then [$x \in_\ell p$]. Closure under outer ranges corresponds to closure in ZFC. The predicate [Tr(x)] is true if [p] is transitive:

$$\left[\ \boxed{\mathrm{Tr}(p)} \doteq \forall x\colon (x \in_2 p \Rightarrow x \in_\ell p \wedge \forall y\colon x ' y \in_2 p\ \right]^*$$

We shall say that [p] is a *transitive hull* of [x] if [p] is transitive and [$x \in_2 p$]. As shown below, all classical [x] have a transitive hull.

If a transitive hull [p] of [x] is minimal w.r.t. [$\subseteq_2$] then [p] is said to be a *transitive closure* of [x]. There probably exist classical maps [x] that have no transitive closure, but all maps definitely have a classical hull.

The main use for transitive hulls is the following: Suppose one has to prove e.g.

$$[\ \forall a, b, c\colon \ell ' \lambda x.a '(x ' b) '(x ' c)\ ].$$

A reasonable approach is to let [p] be a transitive hull of [$a :: b :: c$], to prove

$$[\ x \sim_p \underline{x} \to a '(x ' b) '(x ' c) \sim a '(\underline{x} ' b) '(\underline{x} ' c)\ ],$$

and to conclude [$\forall a, b, c\colon \ell ' \lambda x.a '(x ' b) '(x ' c)$]. It takes about thirty pages to prove the existence of transitive hulls once and for all, and after that it is easy to prove the classicality of a wide range of maps.

transitive
transitive closure
transitive hull
transitive closure

[Map lemma

TypeTr$_{196}$:	$\ell p\colon !\mathrm{Tr}(p)$	;
StrictTr$_{196}$:	$!\mathrm{Tr}(p) \to \ell\,\text{'}\,p$	;
ElimTrInEll$_{196}$:	$\mathrm{Tr}(p) \to x \in_2 p \to x \in_\ell p$	;
ElimTrInTwo$_{196}$:	$\ell y\colon \mathrm{Tr}(p) \to x \in_2 p \to x\,\text{'}\,y \in_2 p$	;
ElimTrHead$_{197}$:	$\mathrm{Tr}(p) \to x :: y \in_2 p \to x \in_2 p$	;
ElimTrTail$_{197}$:	$\mathrm{Tr}(p) \to x :: y \in_2 p \to y \in_2 p$	;
ElimTrSub$_{197}$:	$\mathrm{Tr}(p) \to x \in_2 p \to x \subseteq_2 p$	;
ExistsHull$_{207}$:	$\ell x\colon \exists a\colon x \in_2 a \wedge \mathrm{Tr}(a)$	]*

10.12 Lemmas about [x^{-1}]

In this section we state the theorems of compactness, discrimination and inverse. All three are stated and proved in order to prove the power set axiom of ZFC in Chapter 11, but the theorems are stated here because they are of general interest.

If variables tacitly range over classical maps, then [Compactness] says that if [$x \sim_p y$] for all [p] then [$x \sim y$]. [Discriminate] says that for all [f] one can find a [p] such that [$f\,\text{'}\,x \sim_p f\,\text{'}\,y$] if and only if [$f\,\text{'}\,x \sim f\,\text{'}\,y$] (i.e. one can discriminate between elements of the range of [f] using [$\sim_p$]). The theorem of inverses [IntroInv] says that for all functions [x] one can find an inverse [y] of [x] such that [$x \circ y \circ x \sim x$]. [x^{-1}] denotes one such inverse. The definition of [x^{-1}] reads:

[$x^{-1} \quad \doteq \quad \varepsilon y\colon x \circ y \circ x \sim x$]*

[Map lemma

Compactness$_{212}$:	$\ell x, y\colon \neg x \sim y \to \exists p\colon \neg x \sim_p y$	;
Discriminate$_{214}$:	$\ell x\colon \exists p\colon \forall u\colon \forall v\colon (x\,\text{'}\,u \sim_p x\,\text{'}\,v \Rightarrow x\,\text{'}\,u \sim x\,\text{'}\,v)$	;
TypeInv$_{217}$:	$\ell x\colon \ell\,\text{'}\,x^{-1}$	;
IntroInv$_{218}$:	$\ell x\colon \neg x \to x \circ x^{-1} \circ x \sim x$	]*

Chapter 11

Development of ZFC in MT

11.1 Introduction

Chapter 11 develops ZFC set theory within MT. This serves several purposes:

- It demonstrates that the power (i.e. consistency power) of MT is at least as great as that of ZFC.
- It gives an example of a mathematical treatise expressed in MT for which the topic is non-trivial.
- It allows to import to MT every mathematical theorem that has ever been expressed and proved in ZFC.
- It pushes MT to its limits, which produces a wealth of useful meta-theorems as a side effect.

The theorems presented in Chapter 11 are proved in Chapter A. Readers who want to apply MT to some topic may prove the theorems in Chapter 11 as an exercise for getting acquainted with MT and may use Chapter A as a checklist.

11.2 Overloading

Section 2.2 stated that in Chapter 2, $[\, \mathcal{A} \wedge \mathcal{B} \,]$ stood for conjunction in ZFC. In this chapter, all terms and formulas (i.e. well-formed formulas) are terms and formulas, respectively, of MT unless otherwise noted. As an example, in this chapter, $[\, \mathcal{A} \wedge \mathcal{B} \,]$ denotes conjunction in MT. $[\, \mathcal{A} \wedge \mathcal{B} \,]$ was defined in Section 3.6.

11.3 Presentation of ZFC

The following axiomatization of ZFC is inspired by [8] and [9]. The syntax of terms [T], and formulas (i.e. well-formed formulas) [F] of ZFC reads:

$$\begin{array}{lll} [\,T & ::= & \mathcal{V}; \emptyset; \{T,T\}; \bigcup T; \mathcal{P}T; \omega; \mathsf{C}T \quad]^* \\ [\,F & ::= & T \in T; T = T; \neg F; F \Rightarrow F; \exists \mathcal{V}\colon F \quad]^* \end{array}$$

ZFC has the same variables as MT. [$\emptyset$] denotes the empty set. [$\{x,y\}$] denotes the pair set that contains [x] and [y]. [$\bigcup x$] denotes the union of [x]. [$\mathcal{P}x$] denotes the power set of [x]. [ω] denotes the smallest infinite von Neumann ordinal. [$\mathsf{C}x$] denotes an element of [x] (provided [x] is non-empty). Hence, [$\mathsf{C}x$] is a global choice construct ([8]); [C] is included to simplify the formulation of the axiom of choice.

In the axioms and inference rules, the following definitions are in effect:

$$\begin{array}{llll} [\,x \dot\vee y & \doteq & \neg x \Rightarrow y &]^* \\ [\,x \dot\wedge y & \doteq & \neg(\neg x \dot\vee \neg y) &] \\ [\,x \dot\Leftrightarrow y & \doteq & (x \Rightarrow y) \dot\wedge (y \Rightarrow x) &] \\ [\,\forall x\colon \mathcal{A} & \doteq & \neg\exists x\colon \neg\mathcal{A} &] \\ [\,x \notin y & \doteq & \neg(x \in y) &] \\ [\,x \neq y & \doteq & \neg(x = y) &] \\ [\,x \cup y & \doteq & \bigcup\{x,y\} &] \\ [\,\{x\} & \doteq & \{x,x\} &] \\ [\,x^+ & \doteq & x \cup \{x\} &] \end{array}$$

The axioms and inference rules of ZFC are:

[ZFC rule

Z-MP:	$\langle a, b{\in}F\rangle$	$a \vdash a{\Rightarrow}b \vdash b$	;
Z-Gen:	$\langle x{\in}\mathcal{V}; a{\in}F\rangle$	$a \vdash \forall x\colon a$	;
Z-A1:	$\langle a, b{\in}F\rangle$	$a{\Rightarrow}(b{\Rightarrow}a)$	;
Z-A2:	$\langle a, b, c{\in}F\rangle$	$(a{\Rightarrow}(b{\Rightarrow}c)){\Rightarrow}((a{\Rightarrow}b){\Rightarrow}(a{\Rightarrow}c))$	;
Z-A3:	$\langle a, b{\in}F\rangle$	$(\neg b{\Rightarrow}\neg a){\Rightarrow}((\neg b{\Rightarrow}a){\Rightarrow}b)$	;
Z-A4:	$\langle x{\in}\mathcal{V}; t{\in}T; a, b{\in}F\rangle$	$b \simeq \langle a \mid x{:=}t\rangle \Vdash \forall x\colon a{\Rightarrow}b$	;
Z-A5:	$\langle x{\in}\mathcal{V}; a, b{\in}F\rangle$	notfree$(x; a) \Vdash \forall x\colon (a{\Rightarrow}b){\Rightarrow}(a{\Rightarrow}\forall x\colon b)$	;
Z-S:	$\langle x, y, z{\in}\mathcal{V}; a{\in}F\rangle$	notfree$(y, z; a) \Vdash \forall y\exists z\forall x\colon (x{\in}z{\dot\Leftrightarrow}x{\in}y{\dot\wedge}a)$	;
Z-R:	$\langle x, y, z, u{\in}\mathcal{V}; a{\in}F\rangle$	notfree$(y, z; a) \Vdash$ $\forall z\exists y\forall x\colon (x{\in}z{\dot\wedge}\exists u\colon a{\Rightarrow}\exists u\colon u{\in}y{\dot\wedge}a)$	;
Z-Q:		$\forall x\forall y\colon (x{=}y \dot\Leftrightarrow \forall z\colon (z{\in}x \dot\Leftrightarrow z{\in}y))$	;
Z-E:		$\forall x\forall y\forall z\colon (x{=}y{\Rightarrow}(x{\in}z{\Rightarrow}y{\in}z))$	;
Z-A:		$\forall x\forall y\colon (x{=}y{\Rightarrow}\mathsf{C}x{=}\mathsf{C}y)$	;
Z-N:		$\forall x\colon x \notin \emptyset$	;
Z-P:		$\forall x\forall y\forall z\colon (x{\in}\{y,z\} \dot\Leftrightarrow x{=}y \dot\vee x{=}z)$	;
Z-U:		$\forall x\forall y\colon (x{\in}\bigcup y \dot\Leftrightarrow \exists z\colon x{\in}z{\dot\wedge}z{\in}y)$	;
Z-W:		$\forall x\forall y\colon (x{\in}\mathcal{P}y \dot\Leftrightarrow \forall z\colon (z{\in}x{\Rightarrow}z{\in}y))$	;
Z-I:		$\emptyset{\in}\omega{\dot\wedge}\forall x\colon (x{\in}\omega{\Rightarrow}x^+{\in}\omega)$	;
Z-C:		$\forall x\colon (x{\neq}\emptyset{\Rightarrow}\mathsf{C}x{\in}x)$	;
Z-D:		$\forall x\colon (x{\neq}\emptyset{\Rightarrow}\exists z\colon z{\in}x{\dot\wedge}\neg\exists u\colon u{\in}x{\dot\wedge}u{\in}z)$	]*

In [Z-R], [a] may contain [x] and [u] free. [a] is *univocal* if for all [z] there exists at most one [u] for which [a] is true. [Z-R] expresses the axiom of replacement [9] when [a] is univocal. When [a] is not univocal, there is some overlap between [Z-R] and the axiom of choice. [Z-R] is stronger than the axiom of replacement, but [Z-R] plus the axiom of choice is equivalent to the axiom of replacement plus the axiom of choice.

[Z-D] states that every non-empty set [x] contains a set [z] that is disjoint from [x]. The axiom of restriction implies the axiom of foundation [9] which says that there is no infinite sequence [$x_1, x_2, \ldots$] of sets such that [$x_1 \ni x_2 \ni x_3 \ni \cdots$].

11.4 Modelling ZFC in MT

Section 11.4 introduces a model of ZFC within MT. Within the model, every classical map represents exactly one set and every set is represented by at least one classical map. Let (informally) [L] denote the class of all classical maps. If [$\mathrm{Set}(x)$] denotes the set represented by the classical map [x], then [$\mathrm{Set}(x)$] will satisfy

$$\begin{array}{llll}[\ \mathrm{Set}(\mathsf{N}) & = & \emptyset &] \\ [\ \mathrm{Set}(\lambda x.s) & = & \{\mathrm{Set}(s) \mid x \in L\} &]\end{array}$$

Hence, the set equality relation [$x = y$] must have the following properties:

$$\begin{array}{llll}[\ \mathrm{Set}(\mathsf{N}) & = & \mathrm{Set}(\mathsf{N}) &] \\ [\ \mathrm{Set}(\lambda x.s) & \neq & \mathrm{Set}(\mathsf{N}) &] \\ [\ \mathrm{Set}(\mathsf{N}) & \neq & \mathrm{Set}(\lambda y.t) &]\end{array}$$

Furthermore, the equality relation must satisfy:

$$\begin{array}{lll}[& \mathrm{Set}(\lambda x.s) = \mathrm{Set}(\lambda y.t) & \dot\Leftrightarrow \\ & (\forall u{\in}\mathrm{Set}(\lambda x.s)\colon u \in \mathrm{Set}(\lambda y.t)) \mathbin{\dot\wedge} (\forall v{\in}\mathrm{Set}(\lambda y.t)\colon v \in \mathrm{Set}(\lambda x.s)) & \dot\Leftrightarrow \\ & (\forall x{\in}L\colon \mathrm{Set}(s) \in \mathrm{Set}(\lambda y.t)) \mathbin{\dot\wedge} (\forall y{\in}L\colon \mathrm{Set}(t) \in \mathrm{Set}(\lambda x.s)) & \dot\Leftrightarrow \\ & (\forall x{\in}L\exists y{\in}L\colon \mathrm{Set}(s) = \mathrm{Set}(t)) \mathbin{\dot\wedge} (\forall y{\in}L\exists x{\in}L\colon \mathrm{Set}(t) = \mathrm{Set}(s)) &]\end{array}$$

This allows to define the set equality relation in MT:

$$\left[\ s = t \doteq s \begin{cases} t\begin{cases}\mathsf{T} \\ \mathsf{F}\end{cases} \\ t\begin{cases}\mathsf{F} \\ (\forall x \exists y\colon s\,\text{'}\,x = t\,\text{'}\,y) \mathbin{\dot\wedge} (\forall y \exists x\colon s\,\text{'}\,x = t\,\text{'}\,y)\end{cases}\end{cases}\ \right]^*$$

The constructs [$\neg\mathcal{A}$], [$\mathcal{A} \Rightarrow \mathcal{B}$], and [$\exists x\colon \mathcal{A}$] of ZFC are modeled by the constructs of MT with the same names that were defined in Section 3.6 and

univocal

Section 8.7. Variables of ZFC are modeled by variables of MT. The remaining constructs are modeled thus:

$$
\begin{array}{llll}
[\, s \in t & \doteq & \mathrm{if}(t, \mathsf{F}, \exists x{:}\, s = t\,\text{'}\,x) &]^{*} \\
[\, \emptyset & \doteq & \mathsf{N} &]^{*} \\
[\, \{x,y\} & \doteq & x :: y &]^{*} \\
[\, \mathsf{S}'(y,f) & \doteq & \exists x{:}\, x{\in}y \mathrel{\dot\wedge} f\,\text{'}\,x &]^{*} \\
[\, \mathsf{S}''(y,f) & \doteq & \lambda x.\mathrm{if}(f\,\text{'}(y\,\text{'}\,x), y\,\text{'}\,x, \varepsilon x{:}\, x \in y \mathrel{\dot\wedge} f\,\text{'}\,x) &]^{*} \\
[\, \mathsf{S}(y,f) & \doteq & \mathrm{if}(\mathsf{S}'(y,f), \mathsf{S}''(y,f), \emptyset) &]^{*} \\
[\, \{x{\in}y|\mathcal{A}\} & \doteq & \mathsf{S}(y, \lambda x.\mathcal{A}) &]^{*} \\
[\, u_4(a) & \doteq & \lambda x.a\,\text{'}(x\,\text{'}\,\mathsf{T})\,\text{'}(x\,\text{'}\,\mathsf{F}) &]^{*} \\
[\, \bigcup x & \doteq & \{u \in u_4(x) | \exists v{:}\, u \in v \mathrel{\dot\wedge} v \in x\} &]^{*} \\
[\, W(x) & \doteq & \lambda y.\mathrm{if}(y, \emptyset, \lambda z.x\,\text{'}(y\,\text{'}(x\,\text{'}\,z))) &]^{*} \\
[\, \mathcal{P}x & \doteq & \{u \in W(x) | \forall v{:}\, (v \in u \Rightarrow v \in x)\} &]^{*} \\
[\, \omega & \doteq & \lambda x.\mathrm{if}(x, \emptyset, (\omega\,\text{'}(x\,\text{'}\,\mathsf{N}))^{+}) &]^{*} \\
[\, \mathsf{C}x & \doteq & \varepsilon y{:}\, y \in x &]^{*}
\end{array}
$$

11.5 Verification of the model

To verify that the constructs above model ZFC we need two auxiliary constructs [ZfcExt] and [ZfcExt$'$]. [ZfcExt($\lambda u.\mathcal{A}$)] is true if the truth value of [$\mathcal{A}$] merely depends on which set [u] represents. Hence, if [ZfcExt(f)] is true then one can interpret [f] as a predicate on sets. Likewise, if [ZfcExt$'$(f)] is true then one can interpret [f] as a function from sets to sets. The definitions read:

$$
\begin{array}{lll}
[\, \mathrm{ZfcExt}'(f) & \doteq & \forall u{:}\forall v{:}\, (u = v \Rightarrow \ell\,\text{'}(f\,\text{'}\,u) \mathrel{\dot\wedge} f\,\text{'}\,u = f\,\text{'}\,v) \\
[\, \mathrm{ZfcExt}(f) & \doteq & \forall u{:}\forall v{:}\, (u = v \Rightarrow (f\,\text{'}\,u \mathrel{\dot\Leftrightarrow} f\,\text{'}\,v))
\end{array}
$$

To prove that the model presented in Section 11.4 models ZFC, we shall prove the following five claims:

(1) If [t] is a term of ZFC whose free variables occur among [$x_1, \ldots, x_n$] then [$\ell\,\text{'}\,x_1 \twoheadrightarrow \cdots \twoheadrightarrow \ell\,\text{'}\,x_n \twoheadrightarrow \ell\,\text{'}\,t$].

(2) If [a] is a formula of ZFC whose free variables occur among [$x_1, \ldots, x_n$] then [$\ell\,\text{'}\,x_1 \twoheadrightarrow \cdots \twoheadrightarrow \ell\,\text{'}\,x_n \twoheadrightarrow\, !a$].

(3) If [t] is a term of ZFC whose free variables occur among [$x_1, \ldots, x_n, y$] then [$\ell\,\text{'}\,x_1 \twoheadrightarrow \cdots \twoheadrightarrow \ell\,\text{'}\,x_n \twoheadrightarrow \mathrm{ZfcExt}'(\lambda y.t)$].

(4) If [a] is a formula of ZFC whose free variables occur among [$x_1, \ldots, x_n, y$] then [$\ell\,\text{'}\,x_1 \twoheadrightarrow \cdots \twoheadrightarrow \ell\,\text{'}\,x_n \twoheadrightarrow \mathrm{ZfcExt}(\lambda y.a)$].

(5) If [a] is a theorem of ZFC whose free variables occur among [$x_1, \ldots, x_n$] then [$\ell\,\text{'}\,x_1 \twoheadrightarrow \cdots \twoheadrightarrow \ell\,\text{'}\,x_n \twoheadrightarrow a$].

Claim (5) above is the one that says that the model models ZFC. The observation [$\emptyset \in \emptyset \equiv \mathsf{F}$] ensures that the model is non-trivial in that there exist formulas $[\,\mathcal{A}\,]$ which the model says are false. Claim (1) to (4) are auxiliary claims for proving side conditions needed when proving Claim (5).

Claim (1) follows by structural induction from the following lemmas

[Map lemma

z-TypeEmpty$_{244}$:	$\ell\,'\,\emptyset$	;
z-TypePair$_{244}$:	$\ell x, y\colon \ell\,'\{x, y\}$	;
z-TypeUnion$_{252}$:	$\ell x\colon \ell\,'\bigcup x$	;
z-TypePower$_{257}$:	$\ell x\colon \ell\,'\,\mathcal{P}x$	;
z-TypeInfty$_{266}$:	$\ell\,'\,\omega$	;
z-TypeChoice$_{268}$:	$\ell x\colon \ell\,'\,\mathsf{C}x$	]*

Claim (2) follows by structural induction from the following lemmas:

[Map lemma

z-TypeIn$_{229}$:	$\ell x, y\colon !x \in y$	;
z-TypeEqual$_{221}$:	$\ell x, y\colon !x = y$	;
z-TypeNot$_{218}$:	$!a\colon !\neg a$	;
z-TypeImply$_{218}$:	$!a, b\colon !(a \Rightarrow b)$	;
z-TypeExists$_{218}$:	$\langle x{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \ell\,'\,x \to\, !a \vdash\, !\forall x\colon a$	]*

Claim (3) follows by structural induction from the following lemmas

[Map lemma

z-ExtEmpty$_{244}$:	$\mathrm{ZfcExt}'(\lambda y.\emptyset)$	;
z-ExtPair$_{247}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(b) \to \mathrm{ZfcExt}'(\lambda y.\{a\,'\,y, b\,'\,y\})$	;
z-ExtUnion$_{254}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(\lambda y.\bigcup a\,'\,y)$	;
z-ExtPower$_{264}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(\lambda y.\mathcal{P}a\,'\,y)$	;
z-ExtInfty$_{267}$:	$\mathrm{ZfcExt}'(\lambda y.\omega)$	;
z-ExtChoice$_{268}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(\lambda y.\mathsf{C}a\,'\,y)$	;
z-ExtVarX$_{238}$:	$\ell\,'\,x \to \mathrm{ZfcExt}'(\lambda y.x)$	;
z-ExtVarY$_{238}$:	$\mathrm{ZfcExt}'(\lambda y.y)$	]*

Claim (4) follows by structural induction from the following lemmas:

[Map lemma

z-ExtIn$_{238}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(b) \to \mathrm{ZfcExt}(\lambda y.a\,'\,y \in b\,'\,y)$	;
z-ExtEqual$_{239}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(b) \to \mathrm{ZfcExt}(\lambda y.a\,'\,y = b\,'\,y)$	;
z-ExtNot$_{240}$:	$\mathrm{ZfcExt}(a) \to \mathrm{ZfcExt}(\lambda y.\neg a\,'\,y)$	;
z-ExtImply$_{240}$:	$\mathrm{ZfcExt}(a) \to \mathrm{ZfcExt}(b) \to \mathrm{ZfcExt}(\lambda y.a\,'\,y \Rightarrow b\,'\,y)$	;
z-ExtExists$_{242}$:	$\langle x{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \ell\,'\,x \to \mathrm{ZfcExt}(a) \vdash \mathrm{ZfcExt}(\lambda y.\exists x\colon a\,'\,y)$	]*

Claim (5) follows by induction in the length of the ZFC-proof from the following lemmas:

[Map lemma
z-mp$_{219}$: $a \to a \Rightarrow b \to b$;
z-gen$_{219}$: $\langle x{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \ell\text{'}x \to a \vdash \forall x\colon a$;
z-a1$_{219}$: $!a, b\colon a \Rightarrow (b \Rightarrow a)$;
z-a2$_{219}$: $!a, b, c\colon (a \Rightarrow (b \Rightarrow c)) \Rightarrow ((a \Rightarrow b) \Rightarrow (a \Rightarrow c))$;
z-a3$_{219}$: $!a, b\colon (\neg b \Rightarrow \neg a) \Rightarrow ((\neg b \Rightarrow a) \Rightarrow b)$;
z-a4$_{219}$: $\langle x{\in}\mathcal{V}; a, b, t{\in}\mathcal{T}\rangle b \simeq \langle a \mid x{:=}t\rangle \Vdash$
$\ell\text{'}x \to !a \vdash !b \vdash \ell\text{'}t \vdash \forall x\colon a \Rightarrow b$;
z-a5$_{219}$: $\langle x{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \mathrm{notfree}(x; \mathcal{A}) \Vdash$
$!a \vdash \ell\text{'}x \to !b \vdash \forall x\colon (a \Rightarrow b) \Rightarrow (a \Rightarrow \forall x\colon b)$;
z-q$_{233}$: $\forall x \forall y\colon (x{=}y \mathrel{\dot{\Leftrightarrow}} \forall z\colon (z{\in}x \mathrel{\dot{\Leftrightarrow}} z{\in}y))$;
z-e$_{234}$: $\forall x \forall y \forall z\colon (x{=}y{\Rightarrow}(x{\in}z{\Rightarrow}y{\in}z))$;
z-a$_{268}$: $\forall x \forall y\colon (x{=}y{\Rightarrow}\mathsf{C}x{=}\mathsf{C}y)$;
z-s$_{251}$: $\langle x, y, z{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \mathrm{notfree}(y, z; a) \Vdash \mathrm{ZfcExt}(\lambda x.a) \to$
$\forall y \exists z \forall x\colon (x \in z \mathrel{\dot{\Leftrightarrow}} x \in y \mathrel{\dot{\wedge}} a)$;
z-p$_{246}$: $\forall x \forall y \forall z\colon (x{\in}\{y, z\} \mathrel{\dot{\Leftrightarrow}} x{=}y \mathrel{\dot{\vee}} x{=}z)$;
z-n$_{244}$: $\forall x\colon x \notin \emptyset$;
z-u$_{254}$: $\forall x \forall y\colon (x{\in}\bigcup y \mathrel{\dot{\Leftrightarrow}} \exists z\colon x{\in}z{\dot{\wedge}}z{\in}y)$;
z-w$_{264}$: $\forall x \forall y\colon (x{\in}\mathcal{P}y \mathrel{\dot{\Leftrightarrow}} \forall z\colon (z{\in}x{\Rightarrow}z{\in}y))$;
z-i$_{267}$: $\emptyset{\in}\omega{\dot{\wedge}}\forall x\colon (x{\in}\omega{\Rightarrow}x^{+}{\in}\omega)$;
z-r$_{271}$: $\langle x, y, z, u{\in}\mathcal{V}; \mathcal{A}{\in}F\rangle \mathrm{notfree}(y, z; \mathcal{A}) \Vdash \ell u\colon \mathrm{ZfcExt}(\lambda x.\mathcal{A}) \vdash$
$\forall z \exists y \forall x\colon (x \in z \mathrel{\dot{\wedge}} \exists u\colon \mathcal{A} \Rightarrow \exists u\colon u \in y \mathrel{\dot{\wedge}} \mathcal{A})$;
z-c$_{269}$: $\forall x\colon (x \neq \emptyset \Rightarrow \mathsf{C}x \in x)$;
z-d$_{278}$: $\forall x\colon (x \neq \emptyset \Rightarrow \exists z\colon z \in x \mathrel{\dot{\wedge}} \neg \exists u\colon u \in x \mathrel{\dot{\wedge}} u \in z)$]*

The formal lemmas presented above are proved in Chapter A. The five claims about ZFC follow from the formal lemmas. In conclusion, ZFC has been modeled in MT.

Appendix A

Formal proofs

A.1 Fundamental proofs

[The Map proof of Replace' reads

L1:	Premise $\gg$	$\mathcal{A} \equiv \dot{\mathcal{A}}$	;
L2:	Premise $\gg$	$\mathcal{B} \equiv \mathcal{C}$	;
L3:	Block $\gg$	Begin	;
L4:	Algebra $\gg$	$\dot{\mathcal{B}}$	;
L5:	Commutativity $\rhd$ (Replace $\rhd$ L1) $\gg$	$\mathcal{B}$	;
L6:	Repetition $\rhd$ L2 $\gg$	$\mathcal{C}$	;
L7:	Replace $\rhd$ L1 $\gg$	$\dot{\mathcal{C}}$	;
L8:	Block $\gg$	End	]*

The proof above shows that a proof may end inside an algebraic block: After elimination of shorthand notation, Line [L7] is the last line of the proof, so the proof does prove [Replace'].

[The Map proof of L5.5.1 reads

L1:	Premise $\gg$	$\mathcal{A} \equiv \dot{\mathcal{A}}$	;
L2:	SubLambda $\rhd$ L1 $\gg$	$\lambda x.\mathcal{A} \equiv \lambda x.\dot{\mathcal{A}}$	]*

[The Map proof of L5.5.2 reads

L1:	Premise $\gg$	$\mathcal{A} \equiv \dot{\mathcal{A}}$	;
L2:	Premise $\gg$	$\mathcal{B} \equiv \dot{\mathcal{B}}$	;
L3:	Block $\gg$	Begin	;
L4:	Algebra $\gg$	$\mathcal{A} \mathbin{\text{'}} \mathcal{B}$	;
L5:	Commutativity $\rhd$ ApplyLambda $\gg$	$(\lambda x.x \mathbin{\text{'}} \mathcal{B}) \mathbin{\text{'}} \mathcal{A}$	;
L6:	SubApply $\gg$	$(\lambda x.x \mathbin{\text{'}} \mathcal{B}) \mathbin{\text{'}} \dot{\mathcal{A}}$	;
L7:	ApplyLambda $\gg$	$\dot{\mathcal{A}} \mathbin{\text{'}} \mathcal{B}$	;
L8:	SubApply $\gg$	$\dot{\mathcal{A}} \mathbin{\text{'}} \dot{\mathcal{B}}$	;
L7:	Block $\gg$	End	]*

[The Map proof of L5.5.3 reads

L1:	Premise $\gg$	$\mathcal{A} \equiv \dot{\mathcal{A}}$	;
L2:	Premise $\gg$	$\mathcal{B} \equiv \dot{\mathcal{B}}$	;
L3:	Premise $\gg$	$\mathcal{C} \equiv \dot{\mathcal{C}}$	;
L4:	Block $\gg$	Begin	;
L5:	Algebra $\gg$	$\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C})$	;
L6:	Commutativity $\rhd$ ApplyLambda $\gg$	$(\lambda x.\mathrm{if}(x,\mathcal{B},\mathcal{C}))\mathbin{\text{'}}\mathcal{A}$	;
L7:	SubApply $\rhd$ L1 $\gg$	$(\lambda x.\mathrm{if}(x,\mathcal{B},\mathcal{C}))\mathbin{\text{'}}\dot{\mathcal{A}}$	;
L8:	ApplyLambda $\gg$	$\mathrm{if}(\dot{\mathcal{A}},\mathcal{B},\mathcal{C})$	;
L9:	Commutativity $\rhd$ ApplyLambda $\gg$	$(\lambda x.\mathrm{if}(\dot{\mathcal{A}},x,\mathcal{C}))\mathbin{\text{'}}\mathcal{B}$	;
L10:	SubApply $\rhd$ L2 $\gg$	$(\lambda x.\mathrm{if}(\dot{\mathcal{A}},x,\mathcal{C}))\mathbin{\text{'}}\dot{\mathcal{B}}$	;
L11:	ApplyLambda $\gg$	$\mathrm{if}(\dot{\mathcal{A}},\dot{\mathcal{B}},\mathcal{C})$	;
L12:	Commutativity $\rhd$ ApplyLambda $\gg$	$(\lambda x.\mathrm{if}(\dot{\mathcal{A}},\dot{\mathcal{B}},x))\mathbin{\text{'}}\mathcal{C}$	;
L13:	SubApply $\rhd$ L3 $\gg$	$(\lambda x.\mathrm{if}(\dot{\mathcal{A}},\dot{\mathcal{B}},x))\mathbin{\text{'}}\dot{\mathcal{C}}$	;
L14:	ApplyLambda $\gg$	$\mathrm{if}(\dot{\mathcal{A}},\dot{\mathcal{B}},\dot{\mathcal{C}})$	;
L15:	Block $\gg$	End	]*

A.2 Lemmas that use QND

[The Map proof of Q(x) reads

L1:	Premise $\gg$	$(\lambda x.\mathcal{A})\mathbin{\text{'}}\mathsf{N} \equiv (\lambda x.\mathcal{B})\mathbin{\text{'}}\mathsf{N}$	;
L2:	Premise $\gg$	$(\lambda x.\mathcal{A})\mathbin{\text{'}}\Lambda x \equiv (\lambda x.\mathcal{B})\mathbin{\text{'}}\Lambda x$	;
L3:	Premise $\gg$	$(\lambda x.\mathcal{A})\mathbin{\text{'}}\perp \equiv (\lambda x.\mathcal{B})\mathbin{\text{'}}\perp$	;
L4:	QND $\rhd$ L1 $\rhd$ L2 $\rhd$ L3 $\gg$	$(\lambda x.\mathcal{A})\mathbin{\text{'}}x \equiv (\lambda x.\mathcal{B})\mathbin{\text{'}}x$	;
L5:	Transitivity $\rhd$ L4 $\rhd$ ApplyLambda $\gg$	$(\lambda x.\mathcal{A})\mathbin{\text{'}}x \equiv \mathcal{B}$	;
L6:	Trans $\rhd$ ApplyLambda $\rhd$ L5 $\gg$	$\mathcal{A} \equiv \mathcal{B}$	]*

[The Map proof of Ded:Hyp reads

L1:	Logic $\gg$	$\mathcal{H} \tilde{\wedge} \mathcal{H} \equiv \mathcal{H} \tilde{\wedge} \mathsf{T}$	;
L1:	Repetition $\rhd$ L1 $\gg$	$\mathcal{H} \rightarrow \mathcal{H}$	]*

[The Map proof of Ded:Import reads

L1:	Premise $\gg$	$\mathcal{A} \equiv \mathcal{B}$	;
L2:	Replace $\rhd$ L1 $\gg$	$\mathcal{H} \tilde{\wedge} \mathcal{A} \equiv \mathcal{H} \tilde{\wedge} \mathcal{B}$	;
L3:	Repetition $\rhd$ L2 $\gg$	$\mathcal{H} \rightarrow \mathcal{A} \equiv \mathcal{B}$	]*

[The Map proof of Ded:Trans reads

L1:	Premise $\gg$	$\mathcal{H} \rightarrow \mathcal{A} \equiv \mathcal{B}$	;
L2:	Premise $\gg$	$\mathcal{H} \rightarrow \mathcal{A} \equiv \mathcal{C}$	;
L3:	Repetition $\rhd$ L1 $\gg$	$\mathcal{H} \tilde{\wedge} \mathcal{A} \equiv \mathcal{H} \tilde{\wedge} \mathcal{B}$	;
L4:	Repetition $\rhd$ L2 $\gg$	$\mathcal{H} \tilde{\wedge} \mathcal{A} \equiv \mathcal{H} \tilde{\wedge} \mathcal{C}$	;
L5:	Trans $\rhd$ L3 $\rhd$ L4 $\gg$	$\mathcal{H} \tilde{\wedge} \mathcal{B} \equiv \mathcal{H} \tilde{\wedge} \mathcal{C}$	;
L6:	Repetition $\rhd$ L5 $\gg$	$\mathcal{H} \rightarrow \mathcal{B} \equiv \mathcal{C}$	]*

```
[ The Map proof of Ded:SubLambda reads
   L1:  Premise ≫               $\mathcal{H} \to \mathcal{A} \equiv \mathcal{B}$                                   ;
   L2:  Repetition ▷ L1 ≫       $\mathcal{H} \tilde{\wedge} \mathcal{A} \equiv \mathcal{H} \tilde{\wedge} \mathcal{B}$                         ;
   L3:  SubLambda ▷ L2 ≫        $\lambda x.\mathcal{H} \tilde{\wedge} \mathcal{A} \equiv \lambda x.\mathcal{H} \tilde{\wedge} \mathcal{B}$           ;
   L4:  Logic ≫                 $y \tilde{\wedge} \lambda x.\mathcal{A} \equiv y \tilde{\wedge} \lambda x.y \tilde{\wedge} \mathcal{A}$         ;
   L5:  Logic ≫                 $y \tilde{\wedge} \lambda x.y \tilde{\wedge} \mathcal{B} \equiv y \tilde{\wedge} \lambda x.\mathcal{B}$         ;
   L6:  Block ≫                 Begin                                     ;
   L7:  Algebra ≫                 $\mathcal{H} \tilde{\wedge} \lambda x.\mathcal{A}$                           ;
   L8:  Instantiation ▷ L4 ≫      $\mathcal{H} \tilde{\wedge} \lambda x.\mathcal{H} \tilde{\wedge} \mathcal{A}$               ;
   L9:  Replace ▷ L3 ≫            $\mathcal{H} \tilde{\wedge} \lambda x.\mathcal{H} \tilde{\wedge} \mathcal{B}$               ;
  L10:  Instantiation ▷ L5 ≫      $\mathcal{H} \tilde{\wedge} \lambda x.\mathcal{B}$                           ;
  L11:  Block ≫                 End                                       ;
  L12:  Repetition ▷ L10 ≫      $\mathcal{H} \tilde{\wedge} \lambda x.\mathcal{A} \equiv \mathcal{H} \tilde{\wedge} \lambda x.\mathcal{B}$     ;
  L13:  Repetition ▷ L12 ≫      $\mathcal{H} \to \lambda x.\mathcal{A} \equiv \lambda x.\mathcal{B}$                     ]*
```

```
[ The Map proof of Ded:SubApply reads
   L1:  Premise ≫             $\mathcal{H} \to \mathcal{A} \equiv \mathcal{B}$                                     ;
   L2:  Repetition ▷ L1 ≫     $\mathcal{H} \tilde{\wedge} \mathcal{A} \equiv \mathcal{H} \tilde{\wedge} \mathcal{B}$                           ;
   L3:  Block ≫               Begin                                       ;
   L4:  Algebra ≫               $\mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}\,\mathcal{A}$                               ;
   L5:  Logic ≫                 $\mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}(\mathcal{H} \tilde{\wedge} \mathcal{A})$               ;
   L6:  Replace ▷ L2 ≫          $\mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}(\mathcal{H} \tilde{\wedge} \mathcal{B})$               ;
   L7:  Logic ≫                 $\mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}\,\mathcal{B}$                               ;
   L8:  Block ≫               End                                         ;
   L9:  Repetition ▷ L7 ≫     $\mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}\,\mathcal{A} \equiv \mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}\,\mathcal{B}$       ;
  L10:  Repetition ▷ L9 ≫     $\mathcal{H} \to \mathcal{C}\,\text{'}\,\mathcal{A} \equiv \mathcal{C}\,\text{'}\,\mathcal{B}$                 ]*
```

[Map lemma T4$_{125}$: $\mathcal{A} \equiv \mathcal{B} \vdash \mathcal{B} \equiv \mathcal{C} \vdash \mathcal{C} \equiv \mathcal{D} \vdash \mathcal{A} \equiv \mathcal{D}$]*

```
[ The Map proof of T4 reads
  L1:  Block ≫      Begin   ;
  L2:  Algebra ≫      $\mathcal{A}$     ;
  L3:  Premise ≫      $\mathcal{B}$     ;
  L4:  Premise ≫      $\mathcal{C}$     ;
  L5:  Premise ≫      $\mathcal{D}$     ;
  L6:  Block ≫      End     ]*
```

A.3 Local definitions

[The Map proof of Ded:QND reads

L1:	Premise $\gg$	$\mathcal{H} \mathbin{\tilde{\wedge}} \mathcal{A}\,\text{'}\,\mathsf{N} \equiv \mathcal{H} \mathbin{\tilde{\wedge}} \mathcal{B}\,\text{'}\,\mathsf{N}$	;
L2:	Premise $\gg$	$\mathcal{H} \mathbin{\tilde{\wedge}} \mathcal{A}\,\text{'}\,\Lambda\mathcal{C} \equiv \mathcal{H} \mathbin{\tilde{\wedge}} \mathcal{B}\,\text{'}\,\Lambda\mathcal{C}$	;
L3:	Premise $\gg$	$\mathcal{H} \mathbin{\tilde{\wedge}} \mathcal{A}\,\text{'}\,\perp \equiv \mathcal{H} \mathbin{\tilde{\wedge}} \mathcal{B}\,\text{'}\,\mathsf{N}$	;
L4:	Define $\gg$	$f \equiv \lambda x.\lambda y.\lambda z.x \mathbin{\tilde{\wedge}} y\,\text{'}\,z$	;
L5:	T4 $\rhd \bar{\mathcal{R}} \rhd$ L1 $\rhd \bar{\mathcal{R}} \gg$	$f\,\text{'}\,\mathcal{H}\,\text{'}\,\mathcal{A}\,\text{'}\,\mathsf{N} \equiv f\,\text{'}\,\mathcal{H}\,\text{'}\,\mathcal{B}\,\text{'}\,\mathsf{N}$	;
L6:	T4 $\rhd \bar{\mathcal{R}} \rhd$ L2 $\rhd \bar{\mathcal{R}} \gg$	$f\,\text{'}\,\mathcal{H}\,\text{'}\,\mathcal{A}\,\text{'}\,\Lambda\mathcal{C} \equiv f\,\text{'}\,\mathcal{H}\,\text{'}\,\mathcal{B}\,\text{'}\,\Lambda\mathcal{C}$	;
L7:	T4 $\rhd \bar{\mathcal{R}} \rhd$ L3 $\rhd \bar{\mathcal{R}} \gg$	$f\,\text{'}\,\mathcal{H}\,\text{'}\,\mathcal{A}\,\text{'}\,\perp \equiv f\,\text{'}\,\mathcal{H}\,\text{'}\,\mathcal{B}\,\text{'}\,\perp$	;
L8:	QND $\rhd$ L5 $\rhd$ L6 $\rhd$ L7 $\gg$	$f\,\text{'}\,\mathcal{H}\,\text{'}\,\mathcal{A}\,\text{'}\mathcal{C} \equiv f\,\text{'}\,\mathcal{H}\,\text{'}\,\mathcal{B}\,\text{'}\mathcal{C}$	;
L9:	T4 $\rhd \bar{\mathcal{R}} \rhd$ L8 $\rhd \bar{\mathcal{R}} \gg$	$\mathcal{H} \mathbin{\tilde{\wedge}} \mathcal{A}\,\text{'}\mathcal{C} \equiv \mathcal{H} \mathbin{\tilde{\wedge}} \mathcal{B}\,\text{'}\mathcal{C}$	]*

Line [L4] declares that [f] is shorthand for [$\lambda x.\lambda y.\lambda z.x \mathbin{\tilde{\wedge}} y\,\text{'}\,z$] throughout the proof. To eliminate this shorthand notation, replace all occurrences of [f] by [$\lambda x.\lambda y.\lambda z.x \mathbin{\tilde{\wedge}} y\,\text{'}\,z$] in the proof, and then replace $\left[\boxed{\text{Define}}\right]$ in Line 1 by Reflexivity.

Line [L4] is a local definition which has no effect outside the proof. The left hand side of [$\equiv$] in a local definition must be a variable. All free occurrences of the variable inside the scope of the local definition must be replaced by the right hand side of the local definition before checking the proof. Bound occurrences of the variable are not affected. The scope of a local definition starts at the location of the local definition. The scope ends at the "Block End" of the smallest enclosing block. If the local definition does not occur inside a block then the scope ends at the end of the proof. If more than one local definition of the same variable is in effect in a given proof line, then the latest of the local definitions counts.

[The Map proof of Q'(x) reads

L1:	Premise $\gg$	$\mathcal{A}' \equiv \mathcal{B}'$	;
L2:	Premise $\gg$	$\mathcal{A}'' \equiv \mathcal{B}''$	;
L3:	Premise $\gg$	$\mathcal{A}''' \equiv \mathcal{B}'''$	;
L4:	T4 $\rhd \bar{\mathcal{R}} \rhd$ L1 $\rhd \bar{\mathcal{R}} \gg$	$(\lambda x.\mathcal{A})\,\text{'}\,\mathsf{N} \equiv (\lambda x.\mathcal{B})\,\text{'}\,\mathsf{N}$	;
L5:	T4 $\rhd \bar{\mathcal{R}} \rhd$ L2 $\rhd \bar{\mathcal{R}} \gg$	$(\lambda x.\mathcal{A})\,\text{'}\,\Lambda x \equiv (\lambda x.\mathcal{B})\,\text{'}\,\Lambda x$	;
L6:	T4 $\rhd \bar{\mathcal{R}} \rhd$ L3 $\rhd \bar{\mathcal{R}} \gg$	$(\lambda x.\mathcal{A})\,\text{'}\,\perp \equiv (\lambda x.\mathcal{B})\,\text{'}\,\perp$	;
L7:	Q(x) $\rhd$ L4 $\rhd$ L5 $\rhd$ L6 $\gg$	$\mathcal{A} \equiv \mathcal{B}$	]*

[The Map proof of TND reads

L1:	Premise $\gg$	$!\mathcal{A}$	;
L2:	Premise $\gg$	$\mathcal{A} \to \mathcal{B} \equiv \mathcal{C}$	;
L3:	Premise $\gg$	$\neg\mathcal{A} \to \mathcal{B} \equiv \mathcal{C}$	;
L4:	Repetition $\triangleright$ L2 $\gg$	$\mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{B} \equiv \mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{C}$	;
L5:	Repetition $\triangleright$ L3 $\gg$	$\neg\mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{B} \equiv \neg\mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{C}$	;
L6:	Block $\gg$	Begin	;
L7:	Algebra $\gg$	$\quad\mathcal{B}$	;
L8:	Logic $\trianglerighteq$ L1 $\gg$	$\quad\mathrm{if}(\mathcal{A}, \mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{B}, \neg\mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{B})$	;
L9:	Replace $\triangleright$ L4 $\gg$	$\quad\mathrm{if}(\mathcal{A}, \mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{C}, \neg\mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{B})$	;
L10:	Replace $\triangleright$ L5 $\gg$	$\quad\mathrm{if}(\mathcal{A}, \mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{C}, \neg\mathcal{A} \mathbin{\tilde{\wedge}} \mathcal{C})$	;
L11:	Logic $\trianglerighteq$ L1 $\gg$	$\quad\mathcal{C}$	;
L12:	Block $\gg$	End	]*

[The Map proof of Indirect reads

L1:	Premise $\gg$	$!\mathcal{A}$	;
L2:	Premise $\gg$	$\neg\mathcal{A} \to \mathsf{F}$	;
L3:	Repetition $\triangleright$ L2 $\gg$	$\neg\mathcal{A} \mathbin{\tilde{\wedge}} \mathsf{F} \equiv \neg\mathcal{A} \mathbin{\tilde{\wedge}} \mathsf{T}$	;
L4:	Block $\gg$	Begin	;
L5:	Algebra $\gg$	$\quad\approx\mathcal{A}$	;
L6:	Logic $\gg$	$\quad\neg(\neg\mathcal{A} \mathbin{\tilde{\wedge}} \mathsf{T})$	;
L7:	Replace $\triangleright$ L3 $\gg$	$\quad\neg(\neg\mathcal{A} \mathbin{\tilde{\wedge}} \mathsf{F})$	;
L8:	Logic $\gg$	$\quad!\mathcal{A}$	;
L9:	Repetition $\triangleright$ L1 $\gg$	$\quad\mathsf{T}$	;
L10:	Block $\gg$	End	;
L11:	Repetition $\triangleright$ L9 $\gg$	$\approx\mathcal{A}$	;
L12:	Logic $\trianglerighteq$ L11 $\gg$	$\mathcal{A}$	]*

Note that Line [L9] and Line [L11] are both shorthand for [$\approx\mathcal{A} \equiv \mathsf{T}$] which allows Repetition to be used in Line [L11].

[The Map proof of Indirect' reads

L1:	Premise $\gg$	$!\mathcal{A} \to\ !\mathcal{B}$	;
L2:	Premise $\gg$	$\neg\mathcal{B} \to \neg\mathcal{A}$	;
L3:	Block $\gg$	Begin	;
L4:	Hypothesis $\gg$	$\quad\mathcal{A}$	;
L5:	Reduction $\gg$	$\quad\neg\mathsf{T} \equiv \mathsf{F}$	;
L6:	Replace' $\triangleright$ L4 $\triangleright$ L5 $\gg$	$\quad\neg\mathcal{A} \equiv \mathsf{F}$	;
L7:	Replace' $\triangleright$ L6 $\triangleright$ L5 $\gg$	$\quad\neg\mathcal{B} \to \mathsf{F}$	;
L8:	Reduction $\gg$	$\quad!\mathsf{T} \equiv \mathsf{T}$	;
L9:	Replace' $\triangleright$ L4 $\triangleright$ L8 $\gg$	$\quad!\mathcal{A} \equiv \mathsf{T}$	;
L10:	L1 $\trianglerighteq$ L9 $\gg$	$\quad!\mathcal{B}$	;
L11:	Indirect $\triangleright$ L10 $\triangleright$ L7 $\gg$	$\quad\mathcal{B}$	;
L12:	Block $\gg$	End	]*

[The Map proof of CDeduction reads

L1:	Premise $\gg$	$!\mathcal{A}$	;
L2:	Premise $\gg$	$!\mathcal{B}$	;
L3:	Premise $\gg$	$\mathcal{A} \to \mathcal{B}$	;
L4:	Repetition $\rhd$ L3 $\gg$	$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{A} \tilde{\wedge} \mathsf{T}$	;
L5:	Block $\gg$	Begin	;
L6:	Algebra $\gg$	$\mathcal{A} \Rightarrow \mathcal{B}$	;
L7:	Logic $\unrhd$ L2 $\gg$	$\mathcal{A} \Rightarrow \mathcal{A} \tilde{\wedge} \mathcal{B}$	;
L8:	Replace $\rhd$ L4 $\gg$	$\mathcal{A} \Rightarrow \mathcal{A} \tilde{\wedge} \mathsf{T}$	;
L9:	Logic $\gg$	$!\mathcal{A}$	;
L10:	Repetition $\rhd$ L1 $\gg$	T	;
L11:	Block $\gg$	End	;
L12:	Repetition $\rhd$ L10 $\gg$	$\mathcal{A} \Rightarrow \mathcal{B}$	]*

[The Map proof of Deduction” reads

L1:	Premise $\gg$	$!\mathcal{A}$	;
L2:	Premise $\gg$	$\mathcal{A} \to \mathcal{B}$	;
L3:	Repetition $\rhd$ L2 $\gg$	$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{A} \tilde{\wedge} \mathsf{T}$	;
L4:	Block $\gg$	Begin	;
L5:	Algebra $\gg$	$\mathcal{A} \tilde{\Rightarrow} \mathcal{B}$	;
L6:	Logic $\gg$	$\mathcal{A} \tilde{\Rightarrow} \mathcal{A} \tilde{\wedge} \mathcal{B}$	;
L7:	Replace $\rhd$ L3 $\gg$	$\mathcal{A} \tilde{\Rightarrow} \mathcal{A} \tilde{\wedge} \mathsf{T}$	;
L8:	Logic $\gg$	$!\mathcal{A}$	;
L9:	Repetition $\rhd$ L1 $\gg$	T	;
L10:	Block $\gg$	End	;
L11:	Repetition $\rhd$ L9 $\gg$	$\mathcal{A} \tilde{\Rightarrow} \mathcal{B}$	]*

[The Map proof of IntroIff reads

L1:	Premise $\gg$	$!\mathcal{A}$	;
L2:	Premise $\gg$	$!\mathcal{B}$	;
L3:	Premise $\gg$	$\mathcal{A} \to \mathcal{B}$	;
L4:	Premise $\gg$	$\mathcal{B} \to \mathcal{A}$	;
L5:	CDeduction $\rhd$ L1 $\rhd$ L2 $\rhd$ L3 $\gg$	$\mathcal{A} \Rightarrow \mathcal{B}$	;
L6:	CDeduction $\rhd$ L2 $\rhd$ L1 $\rhd$ L4 $\gg$	$\mathcal{B} \Rightarrow \mathcal{A}$	;
L7:	Logic $\unrhd$ L5 $\unrhd$ L6 $\gg$	$\mathcal{A} \Leftrightarrow \mathcal{B}$	]*

A.4 Greatest lower bounds

The proofs of [L7.3.1] and [L7.3.2] may serve as an introduction to the uses of [Extensionality].

The hard part of proving something by [Extensionality] is to define [$\mathcal{A}$], [$\mathcal{B}$], and [$\mathcal{C}$]. The proof of [L7.3.1] is straightforward, once [$\mathcal{A}$], [$\mathcal{B}$], and [$\mathcal{C}$] are defined:

[The Map proof of L7.3.1 reads

L1:	Define $\gg$	$\mathcal{A} \equiv \lambda x.x$	;
L2:	Define $\gg$	$\mathcal{B} \equiv \lambda x.x \downarrow x$	;
L3:	Define $\gg$	$\mathcal{C} \equiv \lambda u.\lambda v.u \text{'} v$	;
L4:	Block $\gg$	Begin	;
L5:	Algebra $\gg$	$\approx \mathcal{A} \text{'} u$	;
L6:	Reduction $\gg$	$\approx u$	;
L7:	Logic $\gg$	$\approx (u \downarrow u)$	;
L8:	Reduction $\gg$	$\approx \mathcal{B} \text{'} u$	;
L9:	Block $\gg$	End	;
L10:	Block $\gg$	Begin	;
L11:	Algebra $\gg$	$\mathcal{A} \text{'} u \text{'} v$	;
L12:	Reduction $\gg$	$u \text{'} v$	;
L13:	Reduction $\gg$	$\mathcal{A} \text{'} (\mathcal{C} \text{'} u \text{'} v)$	;
L14:	Block $\gg$	End	;
L15:	Block $\gg$	Begin	;
L16:	Algebra $\gg$	$\mathcal{B} \text{'} u \text{'} v$	;
L17:	Reduction $\gg$	$(u \downarrow u) \text{'} v$	;
L18:	Logic $\gg$	$u \text{'} v \downarrow u \text{'} v$	;
L19:	Reduction $\gg$	$\mathcal{B} \text{'} (\mathcal{C} \text{'} u \text{'} v)$	;
L20:	Block $\gg$	End	;
L21:	Extensionality $\rhd$ L8 $\rhd$ L13 $\rhd$ L19 $\gg$	$\mathcal{A} \text{'} u \equiv \mathcal{B} \text{'} u$	;
L22:	Instantiation $\rhd$ L21 $\gg$	$\mathcal{A} \text{'} x \equiv \mathcal{B} \text{'} x$	;
L23:	T4 $\rhd \bar{\mathcal{R}} \rhd$ L22 $\rhd \bar{\mathcal{R}} \gg$	$x \equiv x \downarrow x$	]*

The proof of [L7.3.2] uses the fact that

$$[\ (x \downarrow y) \text{'} z \equiv x \text{'} z \downarrow y \text{'} z\ ]$$

almost holds. The equation above fails e.g. if [$x \equiv \mathsf{N}$] and [$y \equiv \lambda u.\mathsf{N}$]. Now define

$$\left[\ \boxed{x \mathbin{@} y} \doteq \text{if}(x, y, y)\ \right].$$

[$x \mathbin{@} y$] equals [y] unless [x] equals [$\perp$] in which case [$x \mathbin{@} y$] equals [$\perp$]. The following enhancement of the equation above holds for all [x], [y], and [z]:

$$[\ (x \downarrow y) \text{'} z \equiv x \downarrow y \mathbin{@} x \text{'} z \downarrow y \text{'} z\ ]$$

Actually, the statement above is a tautology, so it is provable by [Logic].

The proof of [L7.3.2] also uses the following from Section 3.14:

$$\begin{array}{lcl} [\ n_1(\langle x, y \rangle) & \equiv & x\ ] \\ [\ n_2(\langle x, y \rangle) & \equiv & y\ ] \end{array}$$

And, finally, the proof of [L7.3.2] reads:

[The Map proof of L7.3.2 reads

L1:	Define $\gg$	$a \equiv n_1(u)$	;
L2:	Define $\gg$	$b \equiv n_2(u)$	;
L3:	Define $\gg$	$\mathcal{A} \equiv \lambda u.a \downarrow b$	;
L4:	Define $\gg$	$\mathcal{B} \equiv \lambda u.b \downarrow a$	;
L5:	Define $\gg$	$\mathcal{C}_1 \equiv a \downarrow b \mathbin{@} a\,'\,v$	;
L6:	Define $\gg$	$\mathcal{C}_2 \equiv a \downarrow b \mathbin{@} b\,'\,v$	;
L7:	Define $\gg$	$\mathcal{C} \equiv \lambda u.\lambda v.\langle \mathcal{C}_1, \mathcal{C}_2 \rangle$	;
L8:	Block $\gg$	Begin	;
L9:	Algebra $\gg$	$\approx \mathcal{A}\,'\,u$	;
L10:	Reduction $\gg$	$\approx (a \downarrow b)$	;
L11:	Logic $\gg$	$\approx (b \downarrow a)$	;
L12:	Reduction $\gg$	$\approx \mathcal{B}\,'\,u$	;
L13:	Block $\gg$	End	;
L14:	Block $\gg$	Begin	;
L15:	Algebra $\gg$	$\mathcal{A}\,'\,u\,'\,v$	;
L16:	Reduction $\gg$	$(a \downarrow b)\,'\,v$	;
L17:	Logic $\gg$	$\mathcal{C}_1 \downarrow \mathcal{C}_2$	;
L18:	Reduction $\gg$	$\mathcal{A}\,'(\mathcal{C}\,'\,u\,'\,v)$	;
L19:	Block $\gg$	End	;
L20:	Block $\gg$	Begin	;
L21:	Algebra $\gg$	$\mathcal{B}\,'\,u\,'\,v$	;
L22:	Reduction $\gg$	$(b \downarrow a)\,'\,v$	;
L23:	Logic $\gg$	$\mathcal{C}_2 \downarrow \mathcal{C}_1$	;
L24:	Reduction $\gg$	$\mathcal{B}\,'(\mathcal{C}\,'\,u\,'\,v)$	;
L25:	Block $\gg$	End	;
L26:	Extensionality $\rhd$ L12 $\rhd$ L18 $\rhd$ L24 $\gg$	$\mathcal{A}\,'\,u \equiv \mathcal{B}\,'\,u$	;
L27:	Instantiation $\rhd$ L26 $\gg$	$\mathcal{A}\,'\langle x, y \rangle \equiv \mathcal{B}\,'\langle x, y \rangle$	;
L28:	T4 $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L27 $\rhd$ $\bar{\mathcal{R}}$ $\gg$	$x \downarrow y \equiv y \downarrow x$	]*

The proof of [L7.3.3] contains no news compared to the proof of [L7.3.2]:

[The Map proof of L7.3.3 reads

L1:	Define ≫	$a \equiv n_1(u)$	;
L2:	Define ≫	$b \equiv n_2(u)$	;
L3:	Define ≫	$c \equiv n_3(u)$	;
L4:	Define ≫	$\mathcal{A} \equiv \lambda u.a \downarrow (b \downarrow c)$	;
L5:	Define ≫	$\mathcal{B} \equiv \lambda u.(a \downarrow b) \downarrow c$	;
L6:	Define ≫	$\mathcal{C}_1 \equiv a \downarrow b \downarrow c \mathbin{@} a\text{'}v$	;
L7:	Define ≫	$\mathcal{C}_2 \equiv a \downarrow b \downarrow c \mathbin{@} b\text{'}v$	;
L8:	Define ≫	$\mathcal{C}_3 \equiv a \downarrow b \downarrow c \mathbin{@} c\text{'}v$	;
L9:	Define ≫	$\mathcal{C} \equiv \lambda u.\lambda v.\langle \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 \rangle$	;
L10:	Block ≫	Begin	;
L11:	Algebra ≫	$\quad \approx \mathcal{A}\text{'}u$	;
L12:	Reduction ≫	$\quad \approx (a \downarrow (b \downarrow c))$	;
L13:	Logic ≫	$\quad \approx ((a \downarrow b) \downarrow c)$	;
L14:	Reduction ≫	$\quad \approx \mathcal{B}\text{'}u$	;
L15:	Block ≫	End	;
L16:	Block ≫	Begin	;
L17:	Algebra ≫	$\quad \mathcal{A}\text{'}u\text{'}v$	;
L18:	Reduction ≫	$\quad (a \downarrow (b \downarrow c))\text{'}v$	;
L19:	Logic ≫	$\quad \mathcal{C}_1 \downarrow (\mathcal{C}_2 \downarrow \mathcal{C}_3)$	;
L20:	Reduction ≫	$\quad \mathcal{A}\text{'}(\mathcal{C}\text{'}u\text{'}v)$	;
L21:	Block ≫	End	;
L22:	Block ≫	Begin	;
L23:	Algebra ≫	$\quad \mathcal{B}\text{'}u\text{'}v$	;
L24:	Reduction ≫	$\quad ((a \downarrow b) \downarrow c)\text{'}v$	;
L25:	Logic ≫	$\quad (\mathcal{C}_1 \downarrow \mathcal{C}_2) \downarrow \mathcal{C}_3$	;
L26:	Reduction ≫	$\quad \mathcal{B}\text{'}(\mathcal{C}\text{'}u\text{'}v)$	;
L27:	Block ≫	End	;
L28:	Extensionality ▷ L14 ▷ L20 ▷ L26 ≫	$\mathcal{A}\text{'}u \equiv \mathcal{B}\text{'}u$	;
L29:	Instantiation ▷ L28 ≫	$\mathcal{A}\text{'}\langle x, y, z \rangle \equiv \mathcal{B}\text{'}\langle x, y, z \rangle$	;
L30:	T4 ▷ $\bar{\mathcal{R}}$ ▷ L29 ▷ $\bar{\mathcal{R}}$ ≫	$x \downarrow (y \downarrow z) \equiv (x \downarrow y) \downarrow z$	]*

A.5 Extensionality and deduction

[Map lemma

NonmonoImply$_{131}$:	$\langle a, b, c \in \mathcal{T} \rangle a \rightarrow b \equiv c \vdash a \tilde{\Rightarrow} b \equiv a \tilde{\Rightarrow} c$	;
ImplyNonmono$_{132}$:	$\langle a, b, c \in \mathcal{T} \rangle a \tilde{\Rightarrow} b \equiv a \tilde{\Rightarrow} c \vdash a \rightarrow b \equiv c$	]*

[The Map proof of NonmonoImply reads

L1:	Premise ≫	$a \rightarrow b \equiv c$	;
L2:	Block ≫	Begin	;
L3:	Algebra ≫	$\quad a \tilde{\Rightarrow} b$	;
L4:	Logic ≫	$\quad a \tilde{\Rightarrow} (a \tilde{\wedge} b)$	;
L5:	Replace ▷ L1 ≫	$\quad a \tilde{\Rightarrow} (a \tilde{\wedge} c)$	;
L6:	Logic ≫	$\quad a \tilde{\Rightarrow} c$	;
L7:	Block ≫	End	]*

[The Map proof of ImplyNonmono reads

L1:	Premise $\gg$	$a \Rrightarrow b \equiv a \Rrightarrow c$	;
L2:	Block $\gg$	Begin	;
L3:	Algebra $\gg$	$\quad a \tilde{\wedge} b$	;
L4:	Logic $\gg$	$\quad a \tilde{\wedge} (a \Rrightarrow b)$	;
L5:	Replace $\triangleright$ L1 $\gg$	$\quad a \tilde{\wedge} (a \Rrightarrow c)$	;
L6:	Logic $\gg$	$\quad a \tilde{\wedge} c$	;
L7:	Block $\gg$	End	]*

[The Map proof of Ded:Extensionality reads

L1:	Premise $\gg$	$\mathcal{H} \rightarrow \approx(\mathcal{A}\,'\,u) \equiv \approx(\mathcal{B}\,'\,u)$	;
L2:	Premise $\gg$	$\mathcal{H} \rightarrow \mathcal{A}\,'\,u\,'\,v \equiv \mathcal{A}\,'(\mathcal{C}\,'\,u\,'\,v)$	;
L3:	Premise $\gg$	$\mathcal{H} \rightarrow \mathcal{B}\,'\,u\,'\,v \equiv \mathcal{B}\,'(\mathcal{C}\,'\,u\,'\,v)$	;
L4:	Define $\gg$	$\dot{\mathcal{A}} \equiv \lambda u.\mathcal{H} \Rrightarrow \mathcal{A}\,'\,u$	;
L5:	Define $\gg$	$\dot{\mathcal{B}} \equiv \lambda u.\mathcal{H} \Rrightarrow \mathcal{B}\,'\,u$	;
L6:	Block $\gg$	Begin	;
L7:	Algebra $\gg$	$\quad \approx\dot{\mathcal{A}}\,'\,u$	;
L8:	Reduction $\gg$	$\quad \approx(\mathcal{H} \Rrightarrow \mathcal{A}\,'\,u)$	;
L9:	Logic $\gg$	$\quad \mathcal{H} \Rrightarrow \approx\mathcal{A}\,'\,u$	;
L10:	NonmonoImply $\triangleright$ L1 $\gg$	$\quad \mathcal{H} \Rrightarrow \approx\mathcal{B}\,'\,u$	;
L11:	Logic $\gg$	$\quad \approx(\mathcal{H} \Rrightarrow \mathcal{A}\,'\,u)$	;
L12:	Reduction $\gg$	$\quad \approx\dot{\mathcal{B}}\,'\,u$	;
L13:	Block $\gg$	End	;
L14:	Block $\gg$	Begin	;
L15:	Algebra $\gg$	$\quad \dot{\mathcal{A}}\,'\,u\,'\,v$	;
L16:	Reduction $\gg$	$\quad (\mathcal{H} \Rrightarrow \mathcal{A}\,'\,u)\,'\,v$	;
L17:	Logic $\gg$	$\quad \mathcal{H} \Rrightarrow \mathcal{A}\,'\,u\,'\,v$	;
L18:	NonmonoImply $\triangleright$ L2 $\gg$	$\quad \mathcal{H} \Rrightarrow \mathcal{A}\,'(\mathcal{C}\,'\,u\,'\,v)$	;
L19:	Reduction $\gg$	$\quad \dot{\mathcal{A}}\,'(\mathcal{C}\,'\,u\,'\,v)$	;
L20:	Block $\gg$	End	;
L21:	Block $\gg$	Begin	;
L22:	Algebra $\gg$	$\quad \dot{\mathcal{B}}\,'\,u\,'\,v$	;
L23:	Reduction $\gg$	$\quad (\mathcal{H} \Rrightarrow \mathcal{B}\,'\,u)\,'\,v$	;
L24:	Logic $\gg$	$\quad \mathcal{H} \Rrightarrow \mathcal{B}\,'\,u\,'\,v$	;
L25:	NonmonoImply $\triangleright$ L3 $\gg$	$\quad \mathcal{H} \Rrightarrow \mathcal{B}\,'(\mathcal{C}\,'\,u\,'\,v)$	;
L26:	Reduction $\gg$	$\quad \dot{\mathcal{B}}\,'(\mathcal{C}\,'\,u\,'\,v)$	;
L27:	Block $\gg$	End	;
L28:	Extensionality$\triangleright$L12$\triangleright$L19$\triangleright$L26$\gg$	$\dot{\mathcal{A}}\,'\,u \equiv \dot{\mathcal{B}}\,'\,u$	;
L29:	T4 $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L28 $\triangleright$ $\bar{\mathcal{R}}$ $\gg$	$\mathcal{H} \Rrightarrow \mathcal{A}\,'\,u \equiv \mathcal{H} \Rrightarrow \mathcal{B}\,'\,u$	;
L30:	ImplyNonmono $\triangleright$ L29 $\gg$	$\mathcal{H} \rightarrow \mathcal{A}\,'\,u \equiv \mathcal{B}\,'\,u$	]*

A.6 Monotonicity of abstraction

[The Map proof of L7.5.1 reads

L1:	Premise $\gg$	$\mathcal{A} \preceq \mathcal{B}$	;
L2:	Block $\gg$	Begin	;
L3:	Algebra $\gg$	$\lambda x.\mathcal{A}$	;
L4:	Replace $\rhd$ L1 $\gg$	$\lambda x.\mathcal{A} \downarrow \mathcal{B}$	;
L5:	Reduction $\gg$	$(\lambda x.\mathcal{A}) \downarrow (\lambda x.\mathcal{B})$	;
L6:	Block $\gg$	End	]*

A.7 Some minimal and maximal elements

[The Map proof of L7.6.1 reads

L1:	Reduction $\gg$	$\perp \equiv \perp \downarrow \mathcal{A}$	]*

[The Map proof of L7.6.2 reads

L1:	Premise $\gg$	$\mathcal{A} \preceq \perp$	;
L2:	L7.6.1 $\gg$	$\perp \preceq \mathcal{A}$	;
L3:	L7.4.2 $\rhd$ L1 $\rhd$ L2 $\gg$	$\mathcal{A} \equiv \perp$	]*

[The Map proof of L7.6.3 reads

L1:	Premise $\gg$	$\top \preceq \mathcal{A}$	;
L2:	Commutativity $\rhd$ L1 $\gg$	$\top \downarrow \mathcal{A} \equiv \top$	;
L3:	Logic $\rhd$ L2 $\gg$	$\mathcal{A} \equiv \top$	]*

[The Map proof of L7.6.4 reads

L1:	Premise $\gg$	$\lambda x.\perp \preceq \mathcal{A}$	;
L2:	Block $\gg$	Begin	;
L3:	Algebra $\gg$	$\neg((\lambda x.\perp) \downarrow \mathcal{A})$	;
L4:	Replace $\rhd$ L1 $\gg$	$\neg\lambda x.\perp$	;
L5:	Reduction $\gg$	$\top$	;
L6:	Block $\gg$	End	;
L7:	Repetition $\rhd$ L5 $\gg$	$\neg((\lambda x.\perp) \downarrow \mathcal{A})$	;
L8:	Logic $\rhd$ L7 $\gg$	$\mathcal{A} \equiv \lambda x.\mathcal{A}\,'\,x$	]*

[The Map proof of L7.6.5 reads

L1:	Premise $\gg$	$\mathcal{A} \equiv \lambda x.\mathcal{A}\,'\,x$	;
L2:	Block $\gg$	Begin	;
L3:	Algebra $\gg$	$\lambda x.\perp$	;
L4:	Reduction $\gg$	$(\lambda x.\perp) \downarrow (\lambda x.\mathcal{A}\,'\,x)$	;
L5:	Replace $\rhd$ L1 $\gg$	$(\lambda x.\perp) \downarrow \mathcal{A}$	;
L6:	Block $\gg$	End	]*

[The Map proof of MP' reads

L1:	Premise $\gg$	$\mathcal{A}$	;
L2:	Premise $\gg$	$\mathcal{A} \preceq \mathcal{B}$	;
L3:	Replace' $\rhd$ L1 $\rhd$ L2 $\gg$	$\top \preceq \mathcal{B}$	;
L4:	L7.6.3 $\rhd$ L3 $\gg$	$\mathcal{B}$	]*

A.8 Greatest lower bounds again

[The Map proof of L7.7.1 reads

L1:	L7.3.1 $\gg$	$\mathcal{A} \equiv \mathcal{A} \downarrow \mathcal{A}$	;
L2:	Block $\gg$	Begin	;
L3:	Algebra $\gg$	$\mathcal{A} \downarrow \mathcal{B}$	;
L4:	Replace $\triangleright$ L1 $\gg$	$(\mathcal{A} \downarrow \mathcal{A}) \downarrow \mathcal{B}$	;
L5:	L7.3.3 $\gg$	$\mathcal{A} \downarrow (\mathcal{A} \downarrow \mathcal{B})$	;
L7:	L7.3.2 $\gg$	$(\mathcal{A} \downarrow \mathcal{B}) \downarrow \mathcal{A}$	]*

[The Map proof of L7.7.2 reads

L1:	L7.3.2 $\gg$	$\mathcal{B} \downarrow \mathcal{A} \equiv \mathcal{A} \downarrow \mathcal{B}$	;
L2:	L7.7.1 $\gg$	$\mathcal{B} \downarrow \mathcal{A} \preceq \mathcal{B}$	;
L3:	Replace' $\triangleright$ L1 $\triangleright$ L2 $\gg$	$\mathcal{A} \downarrow \mathcal{B} \preceq \mathcal{B}$	]*

[The Map proof of L7.7.3 reads

L1:	Premise $\gg$	$\mathcal{A} \preceq \mathcal{B}$	;
L2:	Premise $\gg$	$\mathcal{A} \preceq \mathcal{C}$	;
L3:	Block $\gg$	Begin	;
L4:	Algebra $\gg$	$\mathcal{A}$	;
L5:	Repetition $\triangleright$ L2 $\gg$	$\mathcal{A} \downarrow \mathcal{C}$	;
L6:	Replace $\triangleright$ L1 $\gg$	$(\mathcal{A} \downarrow \mathcal{B}) \downarrow \mathcal{C}$	;
L7:	L7.3.3 $\gg$	$\mathcal{A} \downarrow (\mathcal{B} \downarrow \mathcal{C})$	;
L8:	Block $\gg$	End	]*

A.9 Order and implication

[The Map proof of L7.8.1 reads

L1:	Premise $\gg$	$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{A} \tilde{\wedge} \mathsf{T}$	;
L2:	Block $\gg$	Begin	;
L3:	Algebra $\gg$	$?\mathcal{A}$	;
L4:	Logic $\gg$	$?(\mathcal{A} \tilde{\wedge} \mathsf{T})$	;
L5:	Commutativity $\triangleright$ L1 $\gg$	$?(\mathcal{A} \tilde{\wedge} \mathcal{B})$	;
L6:	Logic $\gg$	$?\mathcal{A} \downarrow ?\mathcal{B}$	;
L7:	Block $\gg$	End	]*

[The Map proof of L7.8.2 reads

L1:	Premise $\gg$	$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{A} \tilde{\wedge} \mathsf{T}$	;
L2:	Block $\gg$	Begin	;
L3:	Algebra $\gg$	$?\mathcal{A}$	;
L4:	Logic $\gg$	$?(\mathcal{A} \tilde{\wedge} \mathsf{T})$	;
L5:	Commutativity $\triangleright$ L1 $\gg$	$?(\mathcal{A} \tilde{\wedge} \mathcal{B})$	;
L6:	Logic $\gg$	$?\mathcal{A} \downarrow \mathcal{B}$	;
L7:	Block $\gg$	End	]*

[The Map proof of L7.8.3 reads

L1:	Premise $\gg$	$?\mathcal{A} \preceq ?\mathcal{B}$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad\mathcal{A}$	;
L4:	Logic $\rhd$ L3 $\gg$	$\quad ?\mathcal{A} \equiv \mathsf{T}$	;
L5:	Replace' $\rhd$ L4 $\rhd$ L1 $\gg$	$\quad \mathsf{T} \preceq ?\mathcal{B}$	;
L6:	L7.6.3 $\rhd$ L5 $\gg$	$\quad ?\mathcal{B} \equiv \mathsf{T}$	;
L7:	Logic $\rhd$ L6 $\gg$	$\quad \mathcal{B}$	;
L8:	Block $\gg$	End	]*

[The Map proof of L7.8.4 reads

L1:	Premise $\gg$	$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{A} \tilde{\wedge} \mathsf{T}$	;
L2:	Premise $\gg$	$\neg\mathcal{A} \tilde{\wedge} \neg\mathcal{B} \equiv \neg\mathcal{A} \tilde{\wedge} \mathsf{T}$	;
L3:	Block $\gg$	Begin	;
L4:	Algebra $\gg$	$\quad \approx\mathcal{A}$	;
L5:	Logic $\gg$	$\quad \mathrm{if}(\mathcal{A}, ?(\mathcal{A} \tilde{\wedge} \mathsf{T}), \neg?(\neg\mathcal{A} \tilde{\wedge} \mathsf{T}))$	;
L6:	Replace $\rhd$ L1 $\gg$	$\quad \mathrm{if}(\mathcal{A}, ?(\mathcal{A} \tilde{\wedge} \mathcal{B}), \neg?(\neg\mathcal{A} \tilde{\wedge} \mathsf{T}))$	;
L7:	Replace $\rhd$ L2 $\gg$	$\quad \mathrm{if}(\mathcal{A}, ?(\mathcal{A} \tilde{\wedge} \mathcal{B}), \neg?(\neg\mathcal{A} \tilde{\wedge} \neg\mathcal{B}))$	;
L8:	Logic $\gg$	$\quad \approx\mathcal{A} \downarrow \approx\mathcal{B}$	;
L9:	Block $\gg$	End	]*

[The Map proof of L7.8.5 reads

L1:	Premise $\gg$	$\approx\mathcal{A} \equiv \approx\mathcal{A} \downarrow \approx\mathcal{B}$	;
L2:	Replace $\gg$	$?\approx\mathcal{A} \equiv ?(\approx\mathcal{A} \downarrow \approx\mathcal{B})$	;
L3:	T4 $\rhd$ Logic $\rhd$ L2 $\rhd$ Logic $\gg$	$?\mathcal{A} \equiv ?\mathcal{A} \downarrow ?\mathcal{B}$	;
L4:	L7.8.3 $\rhd$ L3 $\gg$	$\mathcal{A} \to \mathcal{B}$	]*

[The Map proof of L7.8.6 reads

L1:	Premise $\gg$	$\approx\mathcal{A} \equiv \approx\mathcal{A} \downarrow \approx\mathcal{B}$	;
L2:	Replace $\gg$	$?\neg\approx\mathcal{A} \equiv ?\neg(\approx\mathcal{A} \downarrow \approx\mathcal{B})$	;
L3:	T4 $\rhd$ Logic $\rhd$ L2 $\rhd$ Logic $\gg$	$?\neg\mathcal{A} \equiv ?\neg\mathcal{A} \downarrow ?\neg\mathcal{B}$	;
L4:	L7.8.3 $\rhd$ L3 $\gg$	$\neg\mathcal{A} \to \neg\mathcal{B}$	]*

[The Map proof of L7.8.7 reads

L1:	Premise $\gg$	$\mathcal{A} \to \mathcal{B} \preceq \mathcal{C}$	;
L2:	Repetition $\rhd$ L1 $\gg$	$\mathcal{A} \to \mathcal{B} \equiv \mathcal{B} \downarrow \mathcal{C}$	;
L3:	Repetition $\rhd$ L2 $\gg$	$\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{A} \tilde{\wedge} (\mathcal{B} \downarrow \mathcal{C})$	;
L4:	Block $\gg$	Begin	;
L5:	Algebra $\gg$	$\quad \mathcal{A} \tilde{\wedge} \mathcal{B}$	;
L6:	Repetition $\rhd$ L3 $\gg$	$\quad \mathcal{A} \tilde{\wedge} (\mathcal{B} \downarrow \mathcal{C})$	;
L7:	Logic $\gg$	$\quad \mathcal{A} \tilde{\wedge} \mathcal{B} \downarrow \mathcal{A} \tilde{\wedge} \mathcal{C}$	;
L8:	Block $\gg$	End	]*

[The Map proof of L7.8.8 reads

L1: Premise $\gg$ $\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{A} \tilde{\wedge} \mathcal{B} \downarrow \mathcal{A} \tilde{\wedge} \mathcal{C}$;
L2: Block $\gg$ Begin ;
L3: Algebra $\gg$ $\mathcal{A} \tilde{\wedge} \mathcal{B}$;
L4: Repetition $\triangleright$ L1 $\gg$ $\mathcal{A} \tilde{\wedge} \mathcal{B} \downarrow \mathcal{A} \tilde{\wedge} \mathcal{C}$;
L5: Logic $\gg$ $\mathcal{A} \tilde{\wedge} (\mathcal{B} \downarrow \mathcal{C})$;
L6: Block $\gg$ End ;
L7: Repetition $\triangleright$ L5 $\gg$ $\mathcal{A} \tilde{\wedge} \mathcal{B} \equiv \mathcal{A} \tilde{\wedge} (\mathcal{B} \downarrow \mathcal{C})$;
L8: Repetition $\triangleright$ L7 $\gg$ $\mathcal{A} \to \mathcal{B} \equiv \mathcal{B} \downarrow \mathcal{C}$;
L9: Repetition $\triangleright$ L8 $\gg$ $\mathcal{A} \to \mathcal{B} \preceq \mathcal{C}$]*

A.10 Monotonicity

[Map lemma M4$_{136}$: $\mathcal{A} \equiv \mathcal{B} \vdash \mathcal{B} \preceq \mathcal{C} \vdash \mathcal{C} \equiv \mathcal{D} \vdash \mathcal{A} \preceq \mathcal{D}$]*

[The Map proof of M4 reads
L1: $\triangleright \gg$ $\mathcal{A} \equiv \mathcal{B}$;
L2: $\triangleright \gg$ $\mathcal{B} \preceq \mathcal{C}$;
L3: $\triangleright \gg$ $\mathcal{C} \equiv \mathcal{D}$;
L4: Replace' $\triangleright$ $\triangleright$L1 $\triangleright$ L2 $\gg$ $\mathcal{A} \preceq \mathcal{C}$;
L5: Replace' $\triangleright$ L3 $\triangleright$ L4 $\gg$ $\mathcal{A} \preceq \mathcal{D}$]*

[Map lemma
LA.10.1$_{136}$: $\mathcal{A} \preceq \dot{\mathcal{A}} \vdash \mathcal{A} \text{'} \mathcal{B} \preceq \dot{\mathcal{A}} \text{'} \mathcal{B}$;
LA.10.2$_{136}$: $\mathcal{B} \preceq \dot{\mathcal{B}} \vdash \mathcal{A} \text{'} \mathcal{B} \preceq \mathcal{A} \text{'} \dot{\mathcal{B}}$;
LA.10.3$_{136}$: $\mathcal{A} \preceq \dot{\mathcal{A}} \vdash \mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C}) \preceq \mathrm{if}(\dot{\mathcal{A}},\mathcal{B},\mathcal{C})$;
LA.10.4$_{136}$: $\mathcal{B} \preceq \dot{\mathcal{B}} \vdash \mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C}) \preceq \mathrm{if}(\mathcal{A},\dot{\mathcal{B}},\mathcal{C})$;
LA.10.5$_{137}$: $\mathcal{C} \preceq \dot{\mathcal{C}} \vdash \mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C}) \preceq \mathrm{if}(\mathcal{A},\mathcal{B},\dot{\mathcal{C}})$]*

[The Map proof of LA.10.1 reads
L1: Premise$\mathcal{A} \preceq \dot{\mathcal{A}}$;
L2: Monotonicity $\triangleright$ L1 $\gg$ $(\lambda x.x \text{'} \mathcal{B}) \text{'} \mathcal{A} \preceq (\lambda x.x \text{'} \mathcal{B}) \text{'} \dot{\mathcal{A}}$;
L3: M4 $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L2 $\triangleright$ $\bar{\mathcal{R}}$ $\gg$ $\mathcal{A} \text{'} \mathcal{B} \preceq \dot{\mathcal{A}} \text{'} \mathcal{B}$]*

[The Map proof of LA.10.2 reads
L1: Premise$\mathcal{B} \preceq \dot{\mathcal{B}}$;
L2: Monotonicity $\triangleright$ L1 $\gg$ $\mathcal{A} \text{'} \mathcal{B} \preceq \mathcal{A} \text{'} \dot{\mathcal{B}}$]*

[The Map proof of LA.10.3 reads
L1: Premise$\mathcal{A} \preceq \dot{\mathcal{A}}$;
L2: Monotonicity $\triangleright$ L1 $\gg$ $(\lambda x.\mathrm{if}(x,\mathcal{B},\mathcal{C})) \text{'} \mathcal{A} \preceq (\lambda x.\mathrm{if}(x,\mathcal{B},\mathcal{C})) \text{'} \dot{\mathcal{A}}$;
L3: M4 $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L2 $\triangleright$ $\bar{\mathcal{R}}$ $\gg$ $\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C}) \preceq \mathrm{if}(\dot{\mathcal{A}},\mathcal{B},\mathcal{C})$]*

[The Map proof of LA.10.4 reads
L1: Premise$\mathcal{B} \preceq \dot{\mathcal{B}}$;
L2: Monotonicity $\triangleright$ L1 $\gg$ $(\lambda x.\mathrm{if}(\mathcal{A},x,\mathcal{C})) \text{'} \mathcal{B} \preceq (\lambda x.\mathrm{if}(\mathcal{A},x,\mathcal{C})) \text{'} \dot{\mathcal{B}}$;
L3: M4 $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L2 $\triangleright$ $\bar{\mathcal{R}}$ $\gg$ $\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C}) \preceq \mathrm{if}(\mathcal{A},\dot{\mathcal{B}},\mathcal{C})$]*

[The Map proof of LA.10.5 reads
L1: Premise$\mathcal{C} \preceq \dot{\mathcal{C}}$;
L2: Monotonicity $\rhd$ L1 $\gg$ $(\lambda x.\mathrm{if}(\mathcal{A},\mathcal{B},x))\,\text{'}\,\mathcal{C} \preceq (\lambda x.\mathrm{if}(\mathcal{A},\mathcal{B},x))\,\text{'}\,\dot{\mathcal{C}}$;
L3: M4 $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L2 $\rhd$ $\bar{\mathcal{R}}$ $\gg$ $\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C}) \preceq \mathrm{if}(\mathcal{A},\mathcal{B},\dot{\mathcal{C}})$]*

[The Map proof of L7.9.1 reads
L1: L7.4.1 $\gg$ $\mathcal{B} \preceq \mathcal{B}$]*

[The Map proof of L7.9.2 reads
L1: Premise $\gg$ $\mathcal{A} \preceq \dot{\mathcal{A}}$;
L2: L7.5.1 $\rhd$ L1 $\gg$ $\lambda x.\mathcal{A} \equiv \lambda x.\dot{\mathcal{A}}$]*

[The Map proof of L7.9.3 reads
L1: Premise $\gg$ $\mathcal{A} \preceq \dot{\mathcal{A}}$;
L2: Premise $\gg$ $\mathcal{B} \preceq \dot{\mathcal{B}}$;
L3: LA.10.1 $\rhd$ L1 $\gg$ $\mathcal{A}\,\text{'}\,\mathcal{B} \preceq \dot{\mathcal{A}}\,\text{'}\,\mathcal{B}$;
L4: LA.10.2 $\rhd$ L2 $\gg$ $\dot{\mathcal{A}}\,\text{'}\,\mathcal{B} \preceq \dot{\mathcal{A}}\,\text{'}\,\dot{\mathcal{B}}$;
L5: L7.4.3 $\rhd$ L3 $\rhd$ L4 $\gg$ $\mathcal{A}\,\text{'}\,\mathcal{B} \preceq \dot{\mathcal{A}}\,\text{'}\,\dot{\mathcal{B}}$]*

[The Map proof of L7.9.4 reads
L1: Premise $\gg$ $\mathcal{A} \preceq \dot{\mathcal{A}}$;
L2: Premise $\gg$ $\mathcal{B} \preceq \dot{\mathcal{B}}$;
L3: Premise $\gg$ $\mathcal{C} \preceq \dot{\mathcal{C}}$;
L4: LA.10.3 $\rhd$ L1 $\gg$ $\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C}) \preceq \mathrm{if}(\dot{\mathcal{A}},\mathcal{B},\mathcal{C})$;
L5: LA.10.4 $\rhd$ L2 $\gg$ $\mathrm{if}(\dot{\mathcal{A}},\mathcal{B},\mathcal{C}) \preceq \mathrm{if}(\dot{\mathcal{A}},\dot{\mathcal{B}},\mathcal{C})$;
L6: LA.10.5 $\rhd$ L3 $\gg$ $\mathrm{if}(\dot{\mathcal{A}},\dot{\mathcal{B}},\mathcal{C}) \preceq \mathrm{if}(\dot{\mathcal{A}},\dot{\mathcal{B}},\dot{\mathcal{C}})$;
L7: L7.4.3 $\rhd$ L4 $\rhd$ L5 $\gg$ $\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C}) \preceq \mathrm{if}(\dot{\mathcal{A}},\dot{\mathcal{B}},\mathcal{C})$;
L8: L7.4.3 $\rhd$ L7 $\rhd$ L6 $\gg$ $\mathrm{if}(\mathcal{A},\mathcal{B},\mathcal{C}) \preceq \mathrm{if}(\dot{\mathcal{A}},\dot{\mathcal{B}},\dot{\mathcal{C}})$]*

A.11 Monotonicity and deduction

[The Map proof of Ded:Monotonicity reads
L1: Premise $\gg$ $\mathcal{H} \to \mathcal{A} \preceq \mathcal{B}$;
L2: L7.8.7 $\rhd$ L1 $\gg$ $\mathcal{H} \tilde{\wedge} \mathcal{A} \preceq \mathcal{H} \tilde{\wedge} \mathcal{B}$;
L3: Mono $\rhd$ L2 $\gg$ $\mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}(\mathcal{H} \tilde{\wedge} \mathcal{A}) \preceq \mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}(\mathcal{H} \tilde{\wedge} \mathcal{B})$;
L4: Logic $\gg$ $\mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}(\mathcal{H} \tilde{\wedge} \mathcal{A}) \equiv \mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}\,\mathcal{A}$;
L5: Replace' $\rhd$ L4 $\rhd$ L3 $\gg$ $\mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}\,\mathcal{A} \preceq \mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}(\mathcal{H} \tilde{\wedge} \mathcal{B})$;
L6: Logic $\gg$ $\mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}(\mathcal{H} \tilde{\wedge} \mathcal{B}) \equiv \mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}\,\mathcal{B}$;
L7: Replace' $\rhd$ L6 $\rhd$ L5 $\gg$ $\mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}\,\mathcal{A} \preceq \mathcal{H} \tilde{\wedge} \mathcal{C}\,\text{'}\,\mathcal{B}$;
L8: L7.8.8 $\rhd$ L7 $\gg$ $\mathcal{H} \to \mathcal{C}\,\text{'}\,\mathcal{A} \preceq \mathcal{C}\,\text{'}\,\mathcal{B}$]*

A.12 Minimality and deduction

[The Map proof of Ded:Minimality reads

L1:	Premise $\gg$	$\mathcal{H} \to \mathcal{A}\,'\,\mathcal{B} \preceq \mathcal{B}$	;
L2:	L7.8.7 $\rhd$ L1 $\gg$	$\mathcal{H} \mathbin{\tilde\wedge} \mathcal{A}\,'\,\mathcal{B} \preceq \mathcal{H} \mathbin{\tilde\wedge} \mathcal{B}$	;
L3:	Logic $\gg$	$\mathcal{H} \mathbin{\tilde\wedge} \mathcal{A}\,'\,\mathcal{B} \equiv (\lambda x.\mathcal{H} \mathbin{\tilde\wedge} \mathcal{A}\,'\,x)\,'(\mathcal{H} \mathbin{\tilde\wedge} \mathcal{B})$	;
L4:	Replace' $\rhd$ L3 $\rhd$ L2 $\gg$	$(\lambda x.\mathcal{H} \mathbin{\tilde\wedge} \mathcal{A}\,'\,x)\,'(\mathcal{H} \mathbin{\tilde\wedge} \mathcal{B}) \preceq \mathcal{H} \mathbin{\tilde\wedge} \mathcal{B}$	;
L5:	Minimality $\rhd$ L4 $\gg$	$\mathsf{Y}\,'\,\lambda x.\mathcal{H} \mathbin{\tilde\wedge} \mathcal{A}\,'\,x \preceq \mathcal{H} \mathbin{\tilde\wedge} \mathcal{B}$	;
L6:	Logic $\gg$	$\mathsf{Y}\,'\,\lambda x.\mathcal{H} \mathbin{\tilde\wedge} \mathcal{A}\,'\,x \equiv \mathcal{H} \mathbin{\tilde\wedge} \mathsf{Y}\,'\,\mathcal{A}$	;
L7:	Replace' $\rhd$ L6 $\rhd$ L5 $\gg$	$\mathcal{H} \mathbin{\tilde\wedge} \mathsf{Y}\,'\,\mathcal{A} \preceq \mathcal{H} \mathbin{\tilde\wedge} \mathcal{B}$	;
L8:	L7.8.8 $\rhd$ L7 $\gg$	$\mathcal{H} \to \mathsf{Y}\,'\,\mathcal{A} \preceq \mathcal{B}$	]*

A.13 Peano induction

[The Map proof of L7.13.1 reads

L1:	Premise $\gg$	$\mathcal{A}\,'\,0$	;
L2:	Premise $\gg$	$\neg x \to \mathcal{A}\,'(x^-) \to \mathcal{A}\,'\,x$	;
L3:	Reduction $\gg$	$\mathrm{if}(\mathsf{N},\mathsf{N},?\mathcal{A}\,'(\mathsf{N}^-)) \preceq ?\mathsf{N}$	;
L4:	Replace' $\rhd$ L1 $\rhd$ L3 $\gg$	$\mathrm{if}(\mathsf{N},\mathsf{N},?\mathcal{A}\,'(\mathsf{N}^-)) \preceq ?\mathcal{A}\,'\,\mathsf{N}$	;
L5:	Instantiation $\rhd$ L2 $\gg$	$\neg\Lambda x \to \mathcal{A}\,'((\Lambda x)^-) \to \mathcal{A}\,'\,\Lambda x$	;
L6:	L5 $\trianglerighteq$ $\bar{\mathcal{R}}$ $\gg$	$\mathcal{A}\,'((\Lambda x)^-) \to \mathcal{A}\,'\,\Lambda x$	;
L7:	L7.8.1 $\rhd$ L6 $\gg$	$?\mathcal{A}\,'((\Lambda x)^-) \preceq ?\mathcal{A}\,'\,\Lambda x$	;
L8:	Replace' $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L7 $\gg$	$\mathrm{if}(\Lambda x,\mathsf{N},?\mathcal{A}'((\Lambda x)^-)) \preceq ?\mathcal{A}'\Lambda x$	;
L9:	L7.6.1 $\gg$	$\bot \preceq ?\mathcal{A}\,'\,\bot$	;
L10:	Replace' $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L9 $\gg$	$\mathrm{if}(\bot,\mathsf{N},?\mathcal{A}\,'(\bot^-)) \preceq ?\mathcal{A}\,'\,\bot$	;
L11:	Q(x) $\rhd$		
	(T4 $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L4 $\rhd$ $\bar{\mathcal{R}}$) $\rhd$		
	(T4 $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L8 $\rhd$ $\bar{\mathcal{R}}$) $\rhd$		
	(T4 $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L10 $\rhd$ $\bar{\mathcal{R}}$) $\gg$	$\mathrm{if}(x,\mathsf{N},?\mathcal{A}\,'(x^-)) \preceq ?\mathcal{A}\,'\,x$	]*

[The Map proof of L7.13.2 reads

L1:	Premise $\gg$	$\mathcal{A} \preceq ?\mathcal{B}$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad\mathcal{A}$	;
L4:	Repetition $\rhd$ L3 $\gg$	$\quad\mathcal{A} \equiv \mathsf{T}$	;
L5:	Replace' $\rhd$ L4 $\rhd$ L1 $\gg$	$\quad\mathsf{T} \preceq ?\mathcal{B}$	;
L6:	L7.6.3 $\rhd$ L5 $\gg$	$\quad ?\mathcal{B} \equiv \mathsf{T}$	;
L7:	Logic $\rhd$ L6 $\gg$	$\quad\mathcal{B}$	;
L8:	Block $\gg$	End	]*

A.14 Lemmas about $[\,\mathsf{E}x\,]$

A.15 Lemmas about $[\,\varepsilon x\,]$

A.16 Existence

A.17 Universal quantification

A.18 Ackermanns axiom

A.19 Lemmas about $[\,\lim(w,x)\,]$

A.20 LemmasAbout $[\,x =_w y\,]$

A.21 Some classical maps

A.22 Primitive subsets

A.23 Envelopes

A.24 Pure existence

[Map lemma
PureA$_{139}$: $\mathsf{E}x\colon \mathsf{T} \equiv \mathsf{T}$;
PureB$_{139}$: $\langle \mathcal{A}{\in}\mathcal{T}\rangle \mathcal{A} \preceq \mathsf{T} \vdash \mathcal{A} \equiv {?}\mathcal{A}$]*

[The Map proof of PureA reads

L1:	Algebra $\gg$	$\mathsf{E}x\colon \mathsf{T}$	;
L2:	Reduction $\gg$	$\mathsf{E}({?} \circ \mathsf{T})$	;
L3:	Rev $\triangleright$ TruthExistence $\gg$	$\mathsf{E}\mathsf{T}$	;
L4:	Existence $\gg$	T	]*

[The Map proof of PureB reads

L1:	Premise $\gg$	$\mathcal{A} \preceq \mathsf{T}$	;
L2:	Repetition $\triangleright$ L1 $\gg$	$\mathcal{A} \equiv \mathcal{A} \downarrow \mathsf{T}$	;
L3:	Logic $\gg$	$\mathcal{A} \downarrow \mathsf{T} \equiv {?}\mathcal{A}$	;
L4:	Transitivity $\triangleright$ L2 $\triangleright$ L3 $\gg$	$\mathcal{A} \equiv {?}\mathcal{A}$	]*

[The Map proof of DomainPure reads

L1:	Algebra $\gg$	$\mathsf{E}x\colon \mathcal{A}$	;
L2:	Reflexivity $\gg$	$\mathsf{E}\lambda x.\mathcal{A}$	;
L3:	TruthExistence $\gg$	$\mathsf{E}((\lambda x.{?}x) \circ (\lambda x.\mathcal{A}))$	;
L4:	Reduction $\gg$	$\mathsf{E}\lambda x.{?}\mathcal{A}$	;
L5:	Reflexivity $\gg$	$\mathsf{E}x\colon {?}\mathcal{A}$	]*

[The Map proof of RangePure reads

L1:	Logic $\gg$	$?\mathcal{A} \preceq \mathsf{T}$	;
L2:	Mono $\rhd$ L1 $\gg$	$\mathsf{E}x\colon ?\mathcal{A} \preceq \mathsf{E}x\colon \mathsf{T}$	;
L3:	DomainPure $\gg$	$\mathsf{E}x\colon \mathcal{A} \equiv \mathsf{E}x\colon ?\mathcal{A}$	;
L4:	Replace' $\rhd$ L3 $\rhd$ L2 $\gg$	$\mathsf{E}x\colon \mathcal{A} \preceq \mathsf{E}x\colon \mathsf{T}$	;
L5:	Replace' $\rhd$ PureA $\rhd$ L4 $\gg$	$\mathsf{E}x\colon \mathcal{A} \preceq \mathsf{T}$	;
L6:	PureB $\gg$	$\mathsf{E}x\colon \mathcal{A} \equiv ?\mathsf{E}x\colon \mathcal{A}$	]*

[The Map proof of ImplyPure($\mathcal{C}$) reads

L1:	Premise $\gg$	$\mathsf{E}x\colon \mathcal{A}$	;
L2:	Premise $\gg$	$\mathcal{A} \to \dot{\mathcal{B}}$	;
L3:	Reduction $\gg$	$\lambda x.\dot{\mathcal{B}} \equiv (\lambda x.\mathcal{B}) \circ (\lambda x.\mathcal{C})$	;
L4:	Trans $\rhd$ DomainPure $\rhd$ L1 $\gg$	$\mathsf{E}x\colon ?\mathcal{A}$	;
L5:	L7.8.2 $\rhd$ L2 $\gg$	$?\mathcal{A} \preceq \dot{\mathcal{B}}$	;
L6:	Mono $\rhd$ L5 $\gg$	$\mathsf{E}x\colon ?\mathcal{A} \preceq \mathsf{E}x\colon \dot{\mathcal{B}}$	;
L7:	MP' $\rhd$ L4 $\rhd$ L6 $\gg$	$\mathsf{E}x\colon \dot{\mathcal{B}}$	;
L8:	Replace' $\rhd$ L3 $\rhd$ L7 $\gg$	$\mathsf{E}((\lambda x.\mathcal{B}) \circ (\lambda x.\mathcal{C}))$	;
L9:	ImpliedExistence $\rhd$ L8 $\gg$	$\mathsf{E}x\colon \mathcal{B}$	]*

[The Map proof of IntroPure($\mathcal{C}$) reads

L1:	PureA $\gg$	$\mathsf{E}x\colon \mathsf{T}$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\dot{\mathcal{B}}$	;
L4:	Logic $\unrhd$ L3 $\gg$	$\mathsf{T} \to \dot{\mathcal{B}}$	;
L5:	ImplyPure($\mathcal{C}$) $\rhd$ L1 $\rhd$ L4 $\gg$	$\mathsf{E}x\colon \mathcal{B}$	;
L6:	Block $\gg$	End	]*

[The Map proof of ElimPure reads

L01:	Premise $\gg$	$\mathsf{E}x\colon \mathcal{A}$	;
L02:	Premise $\gg$	$\mathcal{A} \to \mathcal{B}$	;
L03:	ImplyPure(x) $\rhd$ L1 $\rhd$ L2 $\gg$	$\mathsf{E}x\colon \mathcal{B}$	;
L04:	Logic $\gg$	$\mathsf{E}x\colon \mathsf{T} \to \mathsf{T}$	;
L05:	NonExistence $\gg$	$\mathsf{E}\bot \equiv \bot$	;
L06:	TruthExistence $\gg$	$\mathsf{E}\bot \equiv \mathsf{E}(? \circ \bot)$	;
L07:	Trans $\rhd$ L6 $\rhd$ L5 $\gg$	$\mathsf{E}(? \circ \bot) \equiv \bot$	;
L08:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L7 $\gg$	$\mathsf{E}x\colon \bot \equiv \bot$	;
L09:	Reduction $\gg$	$\bot \to \bot$	;
L10:	Replace' $\rhd$ L8 $\rhd$ L9 $\gg$	$\mathsf{E}x\colon \bot \to \bot$	;
L11:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L7 $\gg$	$\mathsf{E}x\colon ?\Lambda y \equiv \bot$	;
L12:	DomainPure $\gg$	$\mathsf{E}x\colon \Lambda y \equiv \mathsf{E}x\colon ?\Lambda y$	;
L13:	Transitivity $\rhd$ L12 $\rhd$ L11 $\gg$	$\mathsf{E}x\colon \Lambda y \equiv \bot$	;
L14:	Reduction $\gg$	$\bot \to \Lambda y$	;
L15:	Replace' $\rhd$ L13 $\rhd$ L14 $\gg$	$\mathsf{E}x\colon \Lambda y \to \Lambda y$	;
L16:	Q'(y) $\rhd$ L4 $\rhd$ L15 $\rhd$ L10 $\gg$	$\mathsf{E}x\colon y \to y$	;
L17:	Instantiation $\rhd$ L16 $\gg$	$\mathsf{E}x\colon \mathcal{B} \to \mathcal{B}$	;
L18:	L17 $\unrhd$ L3 $\gg$	$\mathcal{B}$	]*

A.25 Domain and range of quantifiers

[The Map proof of DomainChoice reads

L1:	Algebra $\gg$	$\varepsilon x\colon a \text{'} x$	;
L2:	AckermannChoice $\gg$	$\varepsilon x\colon \ell \text{'} x \wedge a \text{'} x$	;
L3:	Logic $\gg$	$\varepsilon x\colon \ell \text{'} x \wedge \approx a \text{'} x$	;
L4:	Rev $\rhd$ AckermannChoice $\gg$	$\varepsilon x\colon \approx a \text{'} x$	]*

[The Map proof of DomainExists reads

L1:	Algebra $\gg$	$\exists x\colon a \text{'} x$	;
L2:	Reduction $\gg$	$\approx a \text{'} \varepsilon x\colon a \text{'} x$	;
L3:	DomainChoice $\gg$	$\approx a \text{'} \varepsilon x\colon \approx a \text{'} x$	;
L4:	Logic $\gg$	$\approx\approx a \text{'} \varepsilon x\colon \approx a \text{'} x$	;
L5:	Reduction $\gg$	$\exists x\colon \approx a \text{'} x$	]*

[The Map proof of DomainAll reads

L1:	Algebra $\gg$	$\forall x\colon a \text{'} x$	;
L2:	Reflexivity $\gg$	$\neg\exists x\colon \neg a \text{'} x$	;
L3:	DomainExists $\gg$	$\neg\exists x\colon \approx\neg a \text{'} x$	;
L4:	Logic $\gg$	$\neg\exists x\colon \neg\approx a \text{'} x$	;
L5:	Reflexivity $\gg$	$\forall x\colon \approx a \text{'} x$	]*

[The Map proof of RangeEll reads

L01:	Block $\gg$	Begin	;
L02:	Algebra $\gg$	$\bar{\ell}\text{'}(\lambda x.\mathsf{T})$	;
L03:	Reduction $\gg$	$\lambda x.\mathrm{if}(x, \mathsf{T}, \approx \mathsf{E}p\colon \approx x \in_l p)$	;
L04:	RangePure $\gg$	$\lambda x.\mathrm{if}(x, \mathsf{T}, \approx ?\mathsf{E}p\colon \approx x \in_l p)$	;
L05:	Logic $\gg$	$\lambda x.\mathrm{if}(x, \mathsf{T}, \approx ?\mathsf{E}p\colon \approx x \in_l p) \downarrow \mathsf{T}$	;
L06:	RangePure $\gg$	$\lambda x.\mathrm{if}(x, \mathsf{T}, \approx \mathsf{E}p\colon \approx x \in_l p) \downarrow \mathsf{T}$	;
L07:	Reduction $\gg$	$\bar{\ell}\text{'}(\lambda x.\mathsf{T}) \downarrow \lambda x.\mathsf{T}$	;
L08:	Block $\gg$	End	;
L09:	Repetition $\rhd$ L7 $\gg$	$\bar{\ell}\text{'}(\lambda x.\mathsf{T}) \preceq \lambda x.\mathsf{T}$	;
L10:	Minimality $\rhd$ L9 $\gg$	$\ell \preceq \lambda x.\mathsf{T}$	;
L11:	Mono $\rhd$ L10 $\gg$	$\ell \text{'} x \preceq (\lambda x.\mathsf{T}) \text{'} x$	;
L12:	Transitivity $\rhd$ L11 $\rhd$ $\bar{\mathcal{R}}$ $\gg$	$\ell \text{'} x \preceq \mathsf{T}$	;
L13:	PureB $\rhd$ L12 $\gg$	$\ell \text{'} x \equiv ?\ell \text{'} x$	]*

[The Map proof of RangeExists reads

L1:	Algebra $\gg$	$\exists x\colon a \text{'} x$	;
L2:	Reflexivity $\gg$	$\approx(\lambda x.a \text{'} x) \text{'} \varepsilon x\colon a \text{'} x$	;
L3:	Logic $\gg$	$\approx\approx(\lambda x.a \text{'} x) \text{'} \varepsilon x\colon a \text{'} x$	;
L4:	Reflexivity $\gg$	$\approx\exists x\colon a \text{'} x$	]*

[The Map proof of RangeAll reads

L1:	Algebra $\gg$	$\forall x\colon a \text{'} x$	;
L2:	Reflexivity $\gg$	$\neg\exists x\colon \neg a \text{'} x$	;
L3:	Logic $\gg$	$\approx\neg\exists x\colon \neg a \text{'} x$	;
L4:	Reflexivity $\gg$	$\approx\forall x\colon a \text{'} x$	]*

A.26 Negation of quantifiers

[The Map proof of AllNot reads

L1:	Algebra $\gg$	$\forall x\colon \neg a\text{'}x$	;
L2:	Reflexivity $\gg$	$\neg\exists x\colon \neg\neg a\text{'}x$	;
L3:	Logic $\gg$	$\neg\exists x\colon \approx a\text{'}x$	;
L4:	DomainExists $\gg$	$\neg\exists x\colon a\text{'}x$	]*

[The Map proof of NotAll reads

L1:	Algebra $\gg$	$\neg\forall x\colon a\text{'}x$	;
L2:	Reflexivity $\gg$	$\neg\neg\exists x\colon \neg a\text{'}x$	;
L3:	Logic $\gg$	$\approx\exists x\colon \neg a\text{'}x$	;
L4:	RangeExists $\gg$	$\neg\exists x\colon a\text{'}x$	]*

[The Map proof of NotAllNot reads

L1:	Algebra $\gg$	$\neg\forall x\colon \neg a\text{'}x$	;
L2:	Replace $\rhd$ NotAll $\gg$	$\exists x\colon \neg\neg a\text{'}x$	;
L3:	Logic $\gg$	$\exists x\colon \approx a\text{'}x$	;
L4:	DomainExists $\gg$	$\exists x\colon a\text{'}x$	]*

A.27 Type and strictness of quantifiers

[The Map proof of StrictTypeExists reads

L1:	Algebra $\gg$	$!\exists x\colon a\text{'}x$	;
L2:	NotAllNot $\gg$	$!\neg\forall x\colon \neg a\text{'}x$	;
L3:	Logic $\gg$	$!\forall x\colon \neg a\text{'}x$	;
L4:	StrictTypeAll $\gg$	$\forall x\colon !\neg a\text{'}x$	;
L5:	Logic $\gg$	$\forall x\colon !_1 a$	]*

[The Map proof of TypeChoice reads

L1:	Hypothesis $\gg$	$!_1 a$	;
L2:	Transitivity $\rhd$ StrictTypeChoice $\rhd$ L1 $\gg$	$\ell\text{'}\varepsilon x\colon a\text{'}x$	]*

[The Map proof of TypeExists reads

L1:	Hypothesis $\gg$	$!_1 a$	;
L2:	Transitivity $\rhd$ StrictTypeExists $\rhd$ L1 $\gg$	$!\exists x\colon a\text{'}x$	]*

[The Map proof of TypeAll reads

L1:	Hypothesis $\gg$	$!_1 a$	;
L2:	Transitivity $\rhd$ StrictTypeAll $\rhd$ L1 $\gg$	$!\forall x\colon a\text{'}x$	]*

[The Map proof of StrictChoice reads

L1:	Hypothesis $\gg$	$\ell\text{'}\varepsilon x\colon a\text{'}x$	;
L2:	Trans $\rhd$ StrictTypeChoice $\rhd$ L1 $\gg$	$!_1 a$	]*

[The Map proof of StrictExists reads

L1:	Hypothesis $\gg$	$!\exists x\colon a\text{'}x$	;
L2:	Trans $\rhd$ StrictTypeExists $\rhd$ L1 $\gg$	$!_1 a$	]*

[The Map proof of StrictAll reads

L1:	Hypothesis $\gg$	$!\forall x\colon a\,'\,x$	;
L2:	Trans $\rhd$ StrictTypeAll $\rhd$ L1 $\gg$	$!_1 a$	]*

[The Map proof of StrictEll reads

L1:	Hypothesis $\gg$	$!\ell\,'\,x$	;
L2:	RangeEll $\gg$	$\ell\,'\,x \equiv ?\ell\,'\,x$	;
L3:	Replace' $\rhd$ L2 $\rhd$ L1 $\gg$	$!?\ell\,'\,x$	;
L4:	Logic $\unrhd$ L3 $\gg$	$\ell\,'\,x$	]*

A.28 Elimination of quantifiers

[The Map proof of ElimExists reads

L1:	Hypothesis $\gg$	$\exists x\colon a\,'\,x$	;
L2:	Repetition $\rhd$ L1 $\gg$	$\approx(\lambda x.a\,'\,x)\,'\,\varepsilon x\colon a\,'\,x$	;
L3:	Trans $\rhd$ Reduction $\rhd$ L2 $\gg$	$\approx a\,'\,\varepsilon x\colon a\,'\,x$	;
L4:	Logic $\rhd$ L3 $\gg$	$a\,'\,\varepsilon x\colon a\,'\,x$	]*

[The Map proof of ElimExistsEll reads

L1:	Hypothesis $\gg$	$\exists x\colon a\,'\,x$	;
L2:	Logic $\rhd$ L1 $\gg$	$!\exists x\colon a\,'\,x$	;
L3:	StrictExists $\unrhd$ L2 $\gg$	$!_1 a$	;
L4:	TypeChoice $\unrhd$ L3 $\gg$	$\ell\,'\,\varepsilon x\colon a\,'\,x$	]*

[The Map proof of ElimAll2 reads

L1:	Hypothesis $\gg$	$\forall u\colon\forall v\colon a\,'\,u\,'\,v$	;
L2:	Hypothesis $\gg$	$\ell\,'\,b$	;
L3:	Hypothesis $\gg$	$\ell\,'\,c$	;
L4:	ElimAll $\unrhd$ L1 $\unrhd$ L2 $\gg$	$\forall v\colon a\,'\,b\,'\,v$	;
L5:	ElimAll $\unrhd$ L4 $\unrhd$ L3 $\gg$	$a\,'\,b\,'\,c$	]*

[The Map proof of ElimAll3 reads

L1:	Hypothesis $\gg$	$\forall u\colon\forall v\colon\forall w\colon a\,'\,u\,'\,v\,'\,w$	;
L2:	Hypothesis $\gg$	$\ell\,'\,b$	;
L3:	Hypothesis $\gg$	$\ell\,'\,c$	;
L4:	Hypothesis $\gg$	$\ell\,'\,d$	;
L5:	ElimAll2 $\unrhd$ L1 $\unrhd$ L2 $\rhd$ L3 $\gg$	$\forall w\colon a\,'\,b\,'\,c\,'\,w$	;
L6:	ElimAll $\unrhd$ L5 $\unrhd$ L4 $\gg$	$a\,'\,b\,'\,c\,'\,d$	]*

[The Map proof of ElimAll4 reads

L1:	Hypothesis $\gg$	$\forall u\colon\forall v\colon\forall w\colon\forall x\colon a\,'\,u\,'\,v\,'\,w\,'\,x$	;
L2:	Hypothesis $\gg$	$\ell\,'\,b$	;
L3:	Hypothesis $\gg$	$\ell\,'\,c$	;
L4:	Hypothesis $\gg$	$\ell\,'\,d$	;
L5:	Hypothesis $\gg$	$\ell\,'\,e$	;
L6:	ElimAll3 $\unrhd$ L1 $\unrhd$ L2 $\unrhd$ L3 $\unrhd$ L4 $\gg$	$\forall x\colon a\,'\,b\,'\,c\,'\,d\,'\,x$	;
L7:	ElimAll $\unrhd$ L6 $\unrhd$ L5 $\gg$	$a\,'\,b\,'\,c\,'\,d\,'\,e$	]*

A.29 Ackermann rules

[The Map proof of AckermannExists reads

L01:	Algebra $\gg$	$\exists x\colon a\text{'}x$	;
L02:	RangeExists $\gg$	$\approx\exists x\colon a\text{'}x$	;
L03:	Logic $\gg$	$(!\exists x\colon a\text{'}x) \wedge \exists x\colon a\text{'}x$	;
L04:	StrictTypeExists $\gg$	$(\forall x\colon !a\text{'}x) \wedge \exists x\colon a\text{'}x$	;
L05:	StrictTypeChoice $\gg$	$(\ell\text{'}\varepsilon x\colon a\text{'}x) \wedge \exists x\colon a\text{'}x$	;
L06:	Reduction $\gg$	$(\ell\text{'}\varepsilon x\colon a\text{'}x) \wedge \approx a\text{'}\varepsilon x\colon a\text{'}x$	;
L07:	Logic $\gg$	$\approx((\ell\text{'}\varepsilon x\colon a\text{'}x) \wedge a\text{'}\varepsilon x\colon a\text{'}x)$	;
L08:	Reduction $\gg$	$\approx(\lambda x.\ell\text{'}x \wedge a\text{'}x)\text{'}\varepsilon x\colon a\text{'}x$	;
L09:	AckermannChoice $\gg$	$\approx(\lambda x.\ell\text{'}x \wedge a\text{'}x)\text{'}\varepsilon x\colon \ell\text{'}x \wedge a\text{'}x$	;
L10:	Reflexivity $\gg$	$\exists x\colon \ell\text{'}x \wedge a\text{'}x$	]*

[The Map proof of AckermannAll reads

L1:	Algebra $\gg$	$\forall x\colon a\text{'}x$	;
L2:	Reflexivity $\gg$	$\neg\exists x\colon \neg a\text{'}x$	;
L3:	AckermannExists $\gg$	$\neg\exists x\colon \ell\text{'}x \wedge \neg a\text{'}x$	;
L4:	Replace $\rhd$ RangeEll $\gg$	$\neg\exists x\colon ?\ell\text{'}x \wedge \neg a\text{'}x$	;
L5:	Logic $\gg$	$\neg\exists x\colon \neg(?\ell\text{'}x \wedge a\text{'}x)$	;
L6:	RangeEll $\gg$	$\neg\exists x\colon \neg(\ell\text{'}x \wedge a\text{'}x)$	;
L7:	Reflexivity $\gg$	$\forall x\colon \ell\text{'}x \wedge a\text{'}x$	]*

[The Map proof of Ackermann reads

L01:	Hypothesis $\gg$	$\forall x\colon a\text{'}x \Leftrightarrow b\text{'}x$	;
L02:	ElimAll $\unrhd$ L1 $\gg$	$\ell\text{'}x \rightarrow a\text{'}x \Leftrightarrow b\text{'}x$	;
L03:	Repetition $\rhd$ L2 $\gg$	$\ell\text{'}x \mathbin{\tilde{\wedge}} a\text{'}x \Leftrightarrow b\text{'}x \equiv \ell\text{'}x \mathbin{\tilde{\wedge}} \mathsf{T}$	;
L04:	Block $\gg$	Begin	;
L05:	Algebra $\gg$	$\varepsilon x\colon a\text{'}x$	;
L06:	AckermannChoice $\gg$	$\varepsilon x\colon \ell\text{'}x \wedge a\text{'}x$	;
L07:	Logic $\gg$	$\varepsilon x\colon (\ell\text{'}x \mathbin{\tilde{\wedge}} \mathsf{T}) \wedge a\text{'}x$	;
L08:	Replace $\rhd$ L3 $\gg$	$\varepsilon x\colon (\ell\text{'}x \mathbin{\tilde{\wedge}} a\text{'}x \Leftrightarrow b\text{'}x) \wedge a\text{'}x$	;
L09:	Logic $\gg$	$\varepsilon x\colon (\ell\text{'}x \mathbin{\tilde{\wedge}} a\text{'}x \Leftrightarrow b\text{'}x) \wedge b\text{'}x$	;
L10:	Replace $\rhd$ L3 $\gg$	$\varepsilon x\colon (\ell\text{'}x \mathbin{\tilde{\wedge}} \mathsf{T}) \wedge b\text{'}x$	;
L11:	Logic $\gg$	$\varepsilon x\colon \ell\text{'}x \wedge b\text{'}x$	;
L12:	AckermannChoice $\gg$	$\varepsilon x\colon b\text{'}x$	;
L13:	Block $\gg$	End	]*

A.30 Introduction of quantifiers

[The Map proof of IntroExists reads

L01:	Hypothesis $\gg$	$\forall x\colon !a\,\text{'}\,x$	;
L02:	Hypothesis $\gg$	$\ell\,\text{'}\,b$	;
L03:	TypeExists $\rhd$ L1 $\gg$	$!\exists x\colon a\,\text{'}\,x$	;
L04:	Ded:Import $\rhd$ L3 $\gg$	$!a\,\text{'}\,b \to\, !\exists x\colon a\,\text{'}\,x$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	$\quad\neg\exists x\colon a\,\text{'}\,x$	;
L07:	Replace' $\rhd$ AllNot $\rhd$ L6 $\gg$	$\quad\forall x\colon \neg a\,\text{'}\,x$	;
L08:	ElimAll $\unrhd$ L7 $\unrhd$ L2 $\gg$	$\quad\neg a\,\text{'}\,x$	;
L09:	Block $\gg$	End	;
L10:	Indirect' $\rhd$ L8 $\rhd$ L4 $\gg$	$a\,\text{'}\,b \to \exists x\colon a\,\text{'}\,x$	]*

[The Map proof of Gen reads

L01:	Premise $\gg$	$\ell\,\text{'}\,x \to \mathcal{A}$	;
L02:	Repetition $\rhd$ L1 $\gg$	$\ell\,\text{'}\,x \mathbin{\tilde{\wedge}} \mathcal{A} \equiv \ell\,\text{'}\,x \mathbin{\tilde{\wedge}} \mathsf{T}$	;
L03:	Block $\gg$	Begin	;
L04:	Algebra $\gg$	$\quad\forall x\colon \mathcal{A}$	;
L05:	AckermannAll $\gg$	$\quad\forall x\colon \ell\,\text{'}\,x \wedge \mathcal{A}$	;
L06:	Logic $\gg$	$\quad\forall x\colon \ell\,\text{'}\,x \wedge (\ell\,\text{'}\,x \mathbin{\tilde{\wedge}} \mathcal{A})$	;
L07:	Replace $\rhd$ L2 $\gg$	$\quad\forall x\colon \ell\,\text{'}\,x \wedge (\ell\,\text{'}\,x \mathbin{\tilde{\wedge}} \mathsf{T})$	;
L08:	Logic $\gg$	$\quad\forall x\colon \ell\,\text{'}\,x \wedge \mathsf{T}$	;
L09:	AckermannAll $\gg$	$\quad\forall x\colon \mathsf{T}$	;
L10:	Reduction $\gg$	$\quad\mathsf{T}$	;
L11:	Block $\gg$	End	]*

[The Map proof of Gen2 reads

L1:	Premise $\gg$	$\ell\,\text{'}\,x \to \ell\,\text{'}\,y \to \mathcal{A}$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad\ell\,\text{'}\,x$	;
L4:	L1 $\unrhd$ L3 $\gg$	$\quad\ell\,\text{'}\,y \to \mathcal{A}$	;
L5:	Gen $\rhd$ L4 $\gg$	$\quad\forall y\colon \mathcal{A}$	;
L6:	Block $\gg$	End	;
L7:	Gen $\rhd$ L5 $\gg$	$\forall x\colon \forall y\colon \mathcal{A}$	]*

[The Map proof of Gen3 reads

L1:	Premise $\gg$	$\ell\,\text{'}\,x \to \ell\,\text{'}\,y \to \ell\,\text{'}\,z \to \mathcal{A}$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad\ell\,\text{'}\,x$	;
L4:	L1 $\unrhd$ L3 $\gg$	$\quad\ell\,\text{'}\,y \to \ell\,\text{'}\,z \to \mathcal{A}$	;
L5:	Gen2 $\rhd$ L4 $\gg$	$\quad\forall y\colon \forall z\colon \mathcal{A}$	;
L6:	Block $\gg$	End	;
L7:	Gen $\rhd$ L5 $\gg$	$\forall x\colon \forall y\colon \forall z\colon \mathcal{A}$	]*

A.31 Transfinite Induction

The key step in the proof of [Transfinite] is to conclude

$$[\,\mathsf{Y}\,\text{'}\,\bar{\ell} \preceq \lambda x.\mathcal{A}\,]$$

from

$$[\,\bar{\ell}\,\text{'}\,\lambda x.\mathcal{A} \preceq \lambda x.\mathcal{A}\,]$$

by [Minimality]. This happens in step [$\mathsf{L23}$] in the proof of [Transfinite]. The rest of the proof contains chores for proving [$\bar{\ell}\,\text{'}\,\lambda x.\mathcal{A} \preceq \lambda x.\mathcal{A}$] by [QND] and for concluding [$\forall x\colon \mathcal{A}$] from [$\mathsf{Y}\,\text{'}\,\bar{\ell} \preceq \lambda x.\mathcal{A}$].

[The Map proof of Transfinite reads

L01:	Premise $\gg$	$\mathcal{A}_{\mathsf{T}}$	;
L02:	Premise $\gg$	$\neg x \to \forall y\colon \mathcal{A}_{xy} \to$	
		$\mathsf{E}y\colon \mathcal{A}_y \wedge x \in_l y \to \mathcal{A}$	;
L03:	Define $\gg$	$f \doteq \lambda x.\mathcal{A}$	;
L04:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L1 $\gg$	$f\,'\,\mathsf{T} \equiv \mathsf{T}$	;
L05:	L7.4.1 $\gg$	$\mathsf{T} \preceq \mathsf{T}$	;
L06:	Replace' $\rhd$ L4 $\rhd$ L5 $\gg$	$\mathsf{T} \preceq f\,'\,\mathsf{T}$	;
L07:	T4 $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L6 $\rhd$ $\bar{\mathcal{R}}$ $\gg$	$?\mathrm{if}(\mathsf{T}, \mathsf{T}, (\forall y\colon f\,'(\mathsf{T}\,'\,y)) \wedge$	
		$\mathsf{E}y\colon f\,'\,y \wedge \mathsf{T} \in_l y) \preceq f\,'\,\mathsf{T}$	;
L08:	T4 $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L2 $\rhd$ $\bar{\mathcal{R}}$ $\gg$	$\neg x \to \forall y\colon f\,'(x\,'\,y) \to$	
		$\mathsf{E}y\colon f\,'\,y \wedge x \in_l y \to f\,'\,x$	;
L09:	Instantiation $\rhd$ L8 $\gg$	$\neg \Lambda x \to \forall y\colon f\,'((\Lambda x)\,'\,y) \to$	
		$\mathsf{E}y\colon f\,'\,y \wedge \Lambda x \in_l y \to f\,'\,\Lambda x$	;
L10:	Block $\gg$	Begin	;
L11:	Hypothesis $\gg$	$\mathrm{if}(\Lambda x, \mathsf{T}, (\forall y\colon f\,'((\Lambda x)\,'\,y)) \wedge$	
		$\mathsf{E}y\colon f\,'\,y \wedge \Lambda x \in_l y)$	;
L12:	$\bar{\mathcal{R}}$ $\gg$	$\neg \Lambda x$	;
L13:	Logic $\rhd$ L11 $\gg$	$\forall y\colon f\,'((\Lambda x)\,'\,y))$	;
L14:	Logic $\rhd$ L11 $\gg$	$\mathsf{E}y\colon f\,'\,y \wedge \Lambda x \in_l y)$	;
L15:	L9 $\unrhd$ L12 $\unrhd$ L13 $\unrhd$ L14 $\gg$	$f\,'\,\Lambda x$	;
L16:	Block $\gg$	End	;
L17:	L7.8.2 $\rhd$ L15 $\gg$	$?\mathrm{if}(\Lambda x, \mathsf{T}, (\forall y\colon f\,'((\Lambda x)\,'\,y)) \wedge$	
		$\mathsf{E}y\colon f\,'\,y \wedge \Lambda x \in_l y) \preceq f\,'\,\Lambda x$	;
L18:	L7.6.1 $\gg$	$\bot \preceq f\,'\,\bot$	;
L19:	T4 $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L18 $\rhd$ $\bar{\mathcal{R}}$ $\gg$	$?\mathrm{if}(\bot, \mathsf{T}, (\forall y\colon f\,'(\bot\,'\,y)) \wedge$	
		$\mathsf{E}y\colon f\,'\,y \wedge \bot \in_l y) \preceq f\,'\,x$	;
L20:	Q'(x) $\rhd$ L7 $\rhd$ L17 $\rhd$ L19 $\gg$	$?\mathrm{if}(x, \mathsf{T}, (\forall y\colon f\,'(x\,'\,y)) \wedge$	
		$\mathsf{E}y\colon f\,'\,y \wedge x \in_l y) \preceq f\,'\,x$	;
L21:	L7.5.1 $\rhd$ L20 $\gg$	$\lambda x.?\mathrm{if}(x, \mathsf{T}, (\forall y\colon f\,'(x\,'\,y)) \wedge$	
		$\mathsf{E}y\colon f\,'\,y \wedge x \in_l y) \preceq \lambda x.f'x$	;
L22:	T4 $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L21 $\rhd$ $\bar{\mathcal{R}}$ $\gg$	$\bar{\ell}\,'\,f \preceq f$	;
L23:	Minimality $\rhd$ L22 $\gg$	$\ell \preceq f$	;
L24:	Mono $\rhd$ L23 $\gg$	$\ell\,'\,x \preceq f\,'\,x$	;
L25:	Transitivity $\rhd$ L24 $\rhd$ $\bar{\mathcal{R}}$ $\gg$	$\ell\,'\,x \preceq \mathcal{A}$	;
L26:	Block $\gg$	Begin	;
L27:	Hypothesis $\gg$	$\ell\,'\,x$	;
L28:	Replace' $\rhd$ L27 $\rhd$ L25 $\gg$	$\mathsf{T} \preceq \mathcal{A}$	;
L29:	L7.6.3 $\rhd$ L28 $\gg$	$\mathcal{A} \equiv \mathsf{T}$	;
L30:	Block $\gg$	End	]*

[The Map proof of Transfinite" reads

L01:	Premise $\gg$	$\mathcal{A}_{\mathsf{T}}$	;
L02:	Premise $\gg$	$\neg x \to \ell\text{'}x \to \forall y\colon \mathcal{A}_{xy} \to$	
		$\mathsf{E}y\colon \ell\text{'}y \wedge \mathcal{A}_y \wedge x \in_l y \to \mathcal{A}$	;
L03:	Define $\gg$	$a \doteq \lambda x.\mathcal{A}$	;
L04:	Logic $\trianglerighteq$ L1 $\gg$	$\ell\text{'}\mathsf{T} \wedge a\text{'}\mathsf{T}$	;
L05:	T4 $\triangleright \bar{\mathcal{R}} \triangleright$ L2 $\triangleright \bar{\mathcal{R}} \gg$	$\neg x \to \ell\text{'}x \to \forall y\colon a\text{'}(x\text{'}y) \to$	
		$\mathsf{E}y\colon \ell\text{'}y \wedge a\text{'}y \wedge x \in_l y \to a\text{'}x$	;
L06:	Block $\gg$	Begin	;
L07:	Hypothesis $\gg$	$\neg x$	;
L08:	Hypothesis $\gg$	$\forall y\colon \ell\text{'}(x\text{'}y) \wedge a\text{'}(x\text{'}y)$	;
L09:	Hypothesis $\gg$	$\mathsf{E}y\colon \ell\text{'}y \wedge a\text{'}y \wedge x \in_l y$	;
L10:	Block $\gg$	Begin	;
L11:	Hypothesis $\gg$	$\ell\text{'}y$	;
L12:	ElimAll $\trianglerighteq$ L08 $\trianglerighteq$ L11 $\gg$	$\ell\text{'}(x\text{'}y) \wedge a\text{'}(x\text{'}y)$	;
L13:	Logic $\trianglerighteq$ L12 $\gg$	$\ell\text{'}(x\text{'}y)$	;
L14:	Block $\gg$	End	;
L15:	Gen $\triangleright$ L13 $\gg$	$\forall y\colon \ell\text{'}(x\text{'}y)$	;
L16:	Block $\gg$	Begin	;
L17:	Hypothesis $\gg$	$\ell\text{'}y \wedge a\text{'}y \wedge x \in_l y$	;
L18:	Logic $\trianglerighteq$ L17 $\gg$	$\ell\text{'}y \wedge x \in_l y$	;
L19:	Block $\gg$	End	;
L20:	ImplyPure $\trianglerighteq$ L9 $\trianglerighteq$ L18 $\gg$	$\mathsf{E}y\colon \ell\text{'}y \wedge x \in_l y$	;
L21:	Logic $\trianglerighteq$ L7 $\trianglerighteq$ L15 $\trianglerighteq$ L20 $\gg$	$\mathrm{if}(x, \mathsf{T}, \forall y\colon \ell\text{'}(x\text{'}y) \wedge$	
		$\mathsf{E}y\colon \ell\text{'}y \wedge x \in_l y)$	;
L22:	Trans $\triangleright \bar{\mathcal{R}} \triangleright$ L21 $\gg$	$\ell\text{'}x$	;
L23:	Block $\gg$	Begin	;
L24:	Hypothesis $\gg$	$\ell\text{'}y$	;
L25:	ElimAll $\trianglerighteq$ L8 $\trianglerighteq$ L24 $\gg$	$\ell\text{'}(x\text{'}y) \wedge a\text{'}(x\text{'}y)$	;
L26:	Logic $\trianglerighteq$ L25 $\gg$	$a\text{'}(x\text{'}y)$	;
L27:	Block $\gg$	End	;
L28:	Gen $\triangleright$ L26 $\gg$	$\forall y\colon a\text{'}(x\text{'}y)$	;
L29:	L5 $\trianglerighteq$ L7 $\trianglerighteq$ L22 $\trianglerighteq$ L28 $\trianglerighteq$ L9 $\gg$	$a\text{'}x$	;
L30:	Logic $\triangleright$ L22 $\triangleright$ L29 $\gg$	$\ell\text{'}x \wedge a\text{'}x$	;
L31:	Block $\gg$	End	;
L32:	Gen $\triangleright$ (Transfinite $\triangleright$ L4 $\triangleright$ L30) $\gg$	$\forall x\colon \ell\text{'}x \wedge a\text{'}x$	;
L33:	Replace'$\triangleright$AckermannAll$\triangleright$L32 $\gg$	$\forall x\colon a\text{'}x$	;
L34:	Transitivity $\triangleright \bar{\mathcal{R}} \triangleright$ L33 $\gg$	$\forall x\colon \mathcal{A}$	]*

[The Map proof of Transfinite’ reads

L01:	Premise $\gg$	$\mathcal{A}_{\mathsf{T}}$	;
L02:	Premise $\gg$	$\neg x \to \ell\,'\,x \to \forall y\colon \mathcal{A}_{xy} \to \mathcal{A}$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\neg x$	;
L05:	Hypothesis $\gg$	$\ell\,'\,x$	;
L06:	Hypothesis $\gg$	$\forall y\colon \mathcal{A}_{xy}$	;
L07:	Hypothesis $\gg$	$\mathsf{E}y\colon \ell\,'\,y \wedge \mathcal{A}_y \wedge x \in_l y$	;
L08:	L2 $\unrhd$ L4 $\unrhd$ L5 $\unrhd$ L6 $\gg$	$\mathcal{A}$	;
L09:	Block $\gg$	End	;
L10:	Transfinite” $\rhd$ L1 $\rhd$ L8 $\gg$	$\forall x\colon \mathcal{A}$	]*

A.32 Type and subtype rules

[The Map proof of TypeT reads

L1:	Reduction $\gg$	$\ell\,'\,\mathsf{T}$	]*

[The Map proof of TypeK reads

L1:	Block $\gg$	Begin	;
L2:	Hypothesis $\gg$	$\ell\,'\,x$	;
L3:	Hypothesis $\gg$	$\ell\,'\,y$	;
L4:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L2 $\gg$	$\ell\,'(\mathsf{K}\,'\,x\,'\,y)$	;
L5:	Block $\gg$	End	;
L6:	Gen2 $\rhd$ L4 $\gg$	$\ell_2\mathsf{K}$	]*

[The Map proof of TypeIfBLL reads

L01:	Hypothesis $\gg$	$!x$	;
L02:	Hypothesis $\gg$	$\ell\,'\,y$	;
L03:	Hypothesis $\gg$	$\ell\,'\,z$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	x	;
L06:	Logic $\unrhd$ L5 $\gg$	$\ell\,'\,\mathrm{if}(x,y,z) \equiv \ell\,'\,y$	;
L07:	Transitivity $\rhd$ L6 $\rhd$ L2 $\gg$	$\ell\,'\,\mathrm{if}(x,y,z)$	;
L08:	Block $\gg$	End	;
L09:	Block $\gg$	Begin	;
L10:	Hypothesis $\gg$	$\neg x$	;
L11:	Logic $\unrhd$ L10 $\gg$	$\ell\,'\,\mathrm{if}(x,y,z) \equiv \ell\,'\,z$	;
L12:	Transitivity $\rhd$ L11 $\rhd$ L3 $\gg$	$\ell\,'\,\mathrm{if}(x,y,z)$	;
L13:	Block $\gg$	End	;
L14:	TND $\rhd$ L1 $\rhd$ L7 $\rhd$ L12 $\gg$	$\ell\,'\,\mathrm{if}(x,y,z)$	]*

[The Map proof of SubtypeLB reads

L1:	Hypothesis $\gg$	$\ell\,'\,x$	;
L2:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L1 $\gg$	$\mathrm{if}(x, \mathsf{T}, \forall y\colon \ell\,'(x\,'\,y) \wedge \mathsf{E}p\colon \ell\,'\,p \wedge x \in_l p)$	;
L3:	Logic $\unrhd$ L2 $\gg$	$!x$	]*

[The Map proof of SubtypeLL1 reads

L1:	Hypothesis $\gg$	$\ell ' a$	;
L2:	SubtypeLB $\trianglerighteq$ L1 $\gg$	$!a$	;
L3:	Logic $\gg$	$a \rightarrow \ell_1 a$	;
L4:	Block $\gg$	Begin	;
L5:	Hypothesis $\gg$	$\quad \neg a$	;
L6:	Trans $\triangleright \bar{\mathcal{R}} \triangleright$ L1 $\gg$	$\quad \mathrm{if}(a, \mathsf{T}, \forall x \colon \ell '(a ' x) \wedge \mathsf{E} p \colon \ell ' p \wedge a \in_l p)$	;
L7:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$\quad \forall a \colon \ell '(a ' x)$	;
L8:	Block $\gg$	End	;
L9:	TND $\triangleright$ L2 $\triangleright$ L3 $\triangleright$ L7 $\gg$	$\ell_1 a$	]*

[The Map proof of TypeApply reads

L1:	Hypothesis $\gg$	$\ell ' a$	;
L2:	Hypothesis $\gg$	$\ell ' b$	;
L3:	SubtypeLL1 $\trianglerighteq$ L1 $\gg$	$\ell_1 a$	;
L4:	ElimAll $\trianglerighteq$ L3 $\trianglerighteq$ L2 $\gg$	$\ell '(a ' b)$	]*

[The Map proof of SubtypeL1L2 reads

L1:	Hypothesis $\gg$	$\ell_1 a$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad \ell ' x$	;
L4:	Hypothesis $\gg$	$\quad \ell ' y$	;
L5:	ElimAll $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$\quad \ell '(a ' x)$	;
L6:	TypeApply $\trianglerighteq$ L5 $\trianglerighteq$ L4 $\gg$	$\quad \ell '(a ' x ' y)$	;
L7:	Block $\gg$	End	;
L8:	Gen2 $\triangleright$ L6 $\gg$	$\ell_2 a$	]*

[The Map proof of SubtypeL2B2 reads

L1:	Hypothesis $\gg$	$\ell_2 a$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad \ell ' x$	;
L4:	Hypothesis $\gg$	$\quad \ell ' y$	;
L5:	ElimAll $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$\quad \ell '(a ' x)$	;
L6:	ElimAll $\trianglerighteq$ L5 $\trianglerighteq$ L4 $\gg$	$\quad \ell '(a ' x ' y)$	;
L7:	SubtypeLB $\trianglerighteq$ L6 $\gg$	$\quad !a ' x ' y$	;
L8:	Block $\gg$	End	;
L9:	Gen2 $\triangleright$ L6 $\gg$	$!_2 a$	]*

[The Map proof of SubtypeB1B reads

L1:	Logic $\gg$	$!_1 a \rightarrow !a$	]*

[The Map proof of SubtypeB2B1 reads

L1:	Hypothesis $\gg$	$!_2 a$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad \ell ' x$	;
L4:	ElimAll $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$\quad !_1 a ' x$	;
L5:	SubtypeB1B $\trianglerighteq$ L4 $\gg$	$\quad !a ' x$	;
L6:	Block $\gg$	End	;
L7:	Gen $\triangleright$ L5 $\gg$	$!_1 a$	]*

[The Map proof of SubtypeLL2 reads
L1: Hypothesis $\gg$ $\ell\,\text{'}\,a$;
L2: SubtypeLL1 $\trianglerighteq$ L1 $\gg$ $\ell_1 a$;
L3: SubtypeL1L2 $\trianglerighteq$ L2 $\gg$ $\ell_2 a$]*

[The Map proof of SubtypeL1B reads
L1: Hypothesis $\gg$ $\ell_1 a$;
L2: SubtypeL1L2 $\trianglerighteq$ L1 $\gg$ $\ell_2 a$;
L3: SubtypeL2B2 $\trianglerighteq$ L2 $\gg$ $!_2 a$;
L4: SubtypeB2B1 $\trianglerighteq$ L3 $\gg$ $!_1 a$;
L5: SubtypeB1B $\trianglerighteq$ L4 $\gg$ $!a$]*

A.33 Rules about $[\, x =_p y \,]$

[The Map proof of TypeEquiv reads

```
L01: Hypothesis $\gg$                                      $\ell_2 p$                                              ;
L02: Block $\gg$                                           Begin                                                   ;
L03: Hypothesis $\gg$                                        $\ell \text{'} y$                                     ;
L04: SubtypeLB $\trianglerighteq$ L3 $\gg$                   $!y$                                                  ;
L05: Logic $\trianglerighteq$ L4 $\gg$                       $!(\mathsf{T} =_p y)$                                 ;
L06: Block $\gg$                                           End                                                     ;
L07: Gen $\triangleright$ L5 $\gg$                         $\forall y\colon !(\mathsf{T} =_p y)$                   ;
L08: Block $\gg$                                           Begin                                                   ;
L09: Hypothesis $\gg$                                        $\neg x$                                              ;
L10: Hypothesis $\gg$                                        $\ell \text{'} x$                                     ;
L11: Hypothesis $\gg$                                        $\forall a\colon \forall y\colon !(x \text{'} a =_p y)$ ;
L12: Block $\gg$                                             Begin                                                 ;
L13: Hypothesis $\gg$                                          $\ell \text{'} y$                                   ;
L14: SubtypeLB $\trianglerighteq$ L13 $\gg$                    $!y$                                                ;
L15: Logic $\trianglerighteq$ L9 $\gg$                         $y \to !(x =_p y)$                                  ;
L16: Block $\gg$                                               Begin                                               ;
L17: Hypothesis $\gg$                                            $\neg y$                                          ;
L18: Block $\gg$                                                 Begin                                             ;
L19: Hypothesis $\gg$                                              $\ell \text{'} u$                               ;
L20: Block $\gg$                                                   Begin                                           ;
L21: Hypothesis $\gg$                                                $\ell \text{'} v$                             ;
L22: ElimAll2$\trianglerighteq$L1$\trianglerighteq$L19$\trianglerighteq$L21 $\gg$   $\ell \text{'}(p \text{'} u \text{'} v)$   ;
L23: TypeApply $\triangleright$ L13 $\triangleright$ L22 $\gg$       $\ell \text{'}(y \text{'}(p \text{'} u \text{'} v))$   ;
L24: ElimAll2$\triangleright$L11$\triangleright$L22$\triangleright$L23$\gg$   $!x\text{'}(p\text{'}u\text{'}v)=_p y\text{'}(p\text{'}u\text{'}v)$   ;
L25: Block $\gg$                                                   End                                             ;
L26: TypeAll $\trianglerighteq$ (Gen $\triangleright$ L24) $\gg$   $!\forall v\colon x\text{'}(p\text{'}u\text{'}v)=_p y\text{'}(p\text{'}u\text{'}v)$   ;
L27: Block $\gg$                                                 End                                               ;
L28: TypeAll $\trianglerighteq$ (Gen $\triangleright$ L26) $\gg$ $!\forall u\colon \forall v\colon x\text{'}(p\text{'}u\text{'}v)=_p y\text{'}(p\text{'}u\text{'}v)$   ;
L29: Logic $\trianglerighteq$ L9 $\trianglerighteq$ L17 $\trianglerighteq$ L28 $\gg$   $!(x =_p y)$             ;
L30: Block $\gg$                                               End                                                 ;
L31: TND $\triangleright$ L14 $\triangleright$ L15 $\triangleright$ L29 $\gg$   $!(x =_p y)$                     ;
L32: Block $\gg$                                             End                                                   ;
L33: Gen $\triangleright$ L31 $\gg$                          $\forall y\colon !(x =_p y)$                          ;
L34: Block $\gg$                                           End                                                     ;
L35: Transfinite' $\triangleright$ L7 $\triangleright$ L33 $\gg$   $\forall x \forall y\colon !(x =_p y)$          ;
L36: ElimAll2 $\trianglerighteq$ L35 $\gg$                 $\ell \text{'} x \to \ell \text{'} y \to !(x =_p y)$    ]*
```

[The Map proof of ElimEquiv reads

```
L01: Hypothesis $\gg$                     $\ell\,'u$                                       ;
L02: Hypothesis $\gg$                     $\ell\,'v$                                       ;
L03: Hypothesis $\gg$                     $a =_p b$                                        ;
L04: Logic $\unrhd$ L3 $\gg$              $!a$                                             ;
L05: Logic $\unrhd$ L3 $\gg$              $a \to a\,'(p\,'u\,'v) =_p b\,'(p\,'u\,'v)$      ;
L06: Block $\gg$                          Begin                                            ;
L07: Hypothesis $\gg$                       $\neg a$                                       ;
L08: Logic $\unrhd$ L3 $\unrhd$ L7 $\gg$    $\neg b$                                       ;
L09: Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L3 $\gg$   $\mathrm{if}(a, \mathrm{if}(b, \mathsf{T}, \mathsf{F}), \mathrm{if}(b, \mathsf{F},$
                                            $\forall u\colon \forall v\colon a\,'(p\,'u\,'v) =_p b\,'(p\,'u\,'v))$   ;
L10: Logic $\unrhd$ L7 $\unrhd$ L8 $\unrhd$ L9 $\gg$   $\forall u\colon \forall v\colon a\,'(p\,'u\,'v) =_p b\,'(p\,'u\,'v)$   ;
L11: ElimAll2$\unrhd$L10$\unrhd$L1$\unrhd$L2$\gg$   $a\,'(p\,'u\,'v) =_p b\,'(p\,'u\,'v)$   ;
L12: Block $\gg$                          End                                              ;
L13: TND $\rhd$ L4 $\rhd$ L5 $\rhd$ L11 $\gg$   $a\,'(p\,'u\,'v) =_p b\,'(p\,'u\,'v)$      ]*
```

[The Map proof of IntroEquiv reads

```
L1: Premise $\gg$                         $\neg a$                                         ;
L2: Premise $\gg$                         $\neg b$                                         ;
L3: Premise $\gg$                         $\ell u, v\colon a\,'(p\,'u\,'v) =_p b\,'(p\,'u\,'v)$   ;
L4: Gen2 $\rhd$ L3 $\gg$                  $\forall u\colon \forall v\colon a\,'(p\,'u\,'v) =_p b\,'(p\,'u\,'v)$   ;
L5: Logic $\unrhd$ L1 $\unrhd$ L2 $\unrhd$ L4 $\gg$   $\mathrm{if}(a, \mathrm{if}(b, \mathsf{T}, \mathsf{F}), \mathrm{if}(b, \mathsf{F},$
                                          $\forall u\colon \forall v\colon a\,'(p\,'u\,'v) =_p b\,'(p\,'u\,'v)))$   ;
L6: Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L5 $\gg$   $a =_p b$                            ]*
```

[The Map proof of RefEquiv reads

```
L01: Hypothesis $\gg$                     $\ell_2 p$                                       ;
L02: Reduction $\gg$                      $\mathsf{T} =_p \mathsf{T}$                      ;
L03: Block $\gg$                          Begin                                            ;
L04: Hypothesis $\gg$                       $\neg a$                                       ;
L05: Hypothesis $\gg$                       $\ell\,'a$                                     ;
L06: Hypothesis $\gg$                       $\forall y\colon a\,'y =_p a\,'y$              ;
L07: Block $\gg$                            Begin                                          ;
L08: Hypothesis $\gg$                         $\ell\,'u$                                   ;
L09: Hypothesis $\gg$                         $\ell\,'v$                                   ;
L10: ElimAll2 $\unrhd$ L1 $\unrhd$ L8 $\unrhd$ L9 $\gg$   $\ell\,'(p\,'u\,'v)$             ;
L11: ElimAll $\unrhd$ L6 $\unrhd$ L10 $\gg$   $a\,'(p\,'u\,'v) =_p a\,'(p\,'u\,'v)$       ;
L12: Block $\gg$                            End                                            ;
L13: IntroEquiv $\rhd$ L4 $\rhd$ L4 $\rhd$ L11 $\gg$   $a =_p a$                           ;
L14: Block $\gg$                          End                                              ;
L15: Transfinite' $\rhd$ L2 $\rhd$ L13 $\gg$   $\forall a\colon a =_p a$                   ;
L16: ElimAll $\unrhd$ L15 $\gg$           $\ell\,'a \to a =_p a$                          ]*
```

[The Map proof of ComEquiv reads

L01:	Hypothesis ≫	$\ell_2 p$	;
L02:	Hypothesis ≫	$\ell\text{'}a$	;
L03:	Hypothesis ≫	$\ell\text{'}b$	;
L04:	Hypothesis ≫	$a =_p b$	;
L05:	Block ≫	Begin	;
L06:	Hypothesis ≫	$\quad \ell\text{'}y$	;
L07:	TypeT ≫	$\quad \ell\text{'}\mathsf{T}$	;
L08:	TypeEquiv ⊵ L1 ⊵ L7 ⊵ L6 ≫	$\quad !\mathsf{T} =_p b$	;
L09:	TypeEquiv ⊵ L1 ⊵ L6 ⊵ L7 ≫	$\quad !b =_p \mathsf{T}$	;
L12:	Logic ≫	$\quad \mathsf{T} =_p b \rightarrow b =_p \mathsf{T}$	;
L14:	CDeduction▷L8▷L9▷L12 ≫	$\quad \mathsf{T} =_p b \Rightarrow b =_p \mathsf{T}$	;
L15:	Block ≫	End	;
L16:	Gen ▷ L14 ≫	$\forall y\colon (\mathsf{T} =_p b \Rightarrow b =_p \mathsf{T})$	;
L17:	Block ≫	Begin	;
L18:	Hypothesis ≫	$\quad \neg a$	;
L19:	Hypothesis ≫	$\quad \ell\text{'}a$	;
L20:	Hypothesis ≫	$\quad \forall y\colon \forall b\colon (a\text{'}y =_p b \Rightarrow b =_p a\text{'}y)$	;
L21:	Block ≫	$\quad$ Begin	;
L22:	Hypothesis ≫	$\quad\quad \ell\text{'}b$	;
L23:	TypeEquiv⊵L1⊵L19⊵L22 ≫	$\quad\quad !a =_p b$	;
L24:	TypeEquiv⊵L1⊵L22⊵L19 ≫	$\quad\quad !b =_p a$	;
L25:	Block ≫	$\quad\quad$ Begin	;
L26:	Hypothesis ≫	$\quad\quad\quad a =_p b$	;
L27:	Logic ⊵ L18 ⊵ L26 ≫	$\quad\quad\quad \neg b$	;
L28:	Block ≫	$\quad\quad\quad$ Begin	;
L29:	Hypothesis ≫	$\quad\quad\quad\quad \ell\text{'}u$	;
L30:	Hypothesis ≫	$\quad\quad\quad\quad \ell\text{'}v$	;
L31:	ElimAll2 ⊵ L1 ⊵ L29 ⊵ L30 ≫	$\quad\quad\quad\quad \ell\text{'}(p\text{'}u\text{'}v)$	;
L32:	TypeApply ⊵ L22 ⊵ L31 ≫	$\quad\quad\quad\quad \ell\text{'}b\text{'}(p\text{'}u\text{'}v)$	;
L33:	ElimAll2⊵L20⊵L31⊵L32 ≫	$\quad\quad\quad\quad a\text{'}(p\text{'}u\text{'}v) =_p b\text{'}(p\text{'}u\text{'}v) \Rightarrow$ $b\text{'}(p\text{'}u\text{'}v) =_p a\text{'}(p\text{'}u\text{'}v)$	;
L34:	ElimEquiv⊵L29⊵L30⊵L26≫	$\quad\quad\quad\quad a\text{'}(p\text{'}u\text{'}v) =_p b\text{'}(p\text{'}u\text{'}v)$	;
L35:	Logic ⊵ L33 ⊵ L34 ≫	$\quad\quad\quad\quad b\text{'}(p\text{'}u\text{'}v) =_p a\text{'}(p\text{'}u\text{'}v)$	;
L36:	Block ≫	$\quad\quad\quad$ End	;
L37:	IntroEquiv▷L28▷L18▷L35≫	$\quad\quad\quad b =_p a$	;
L38:	Block ≫	$\quad\quad$ End	;
L39:	CDeduction▷L23▷L24▷L37≫	$\quad\quad a =_p b \Rightarrow b =_p a$	;
L40:	Block ≫	$\quad$ End	;
L41:	Gen ▷ L39 ≫	$\quad \forall b\colon (a =_p b \Rightarrow b =_p a)$	;
L42:	Block ≫	End	;
L43:	Transfinite' ▷ L16 ▷ L41 ≫	$\forall a\colon \forall b\colon (a =_p b \Rightarrow b =_p a)$	;
L44:	ElimAll2 ⊵ L43 ⊵ L2 ⊵ L3 ≫	$a =_p b \Rightarrow b =_p a$	;
L45:	Logic ⊵ L4 ⊵ L44 ≫	$b =_p a$	]*

[Map lemma

TransEquiv1$_{155}$:	$\ell_2 p$: $\forall b$: $\forall c$: $(\mathsf{T} =_p b \wedge b =_p c \Rightarrow \mathsf{T} =_p c)$	;
TransEquiv2$_{155}$:	$\ell_2 p$: $\neg a \rightarrow \ell\,\text{'}\,a \rightarrow$	
	$\forall y$: $\forall a$: $\forall b$: $\forall c$: $(a\,\text{'}\,y =_p b \wedge b =_p c \Rightarrow a\,\text{'}\,y =_p c) \rightarrow$	
	$\forall a$: $\forall b$: $\forall c$: $(a =_p b \wedge b =_p c \Rightarrow a =_p c)$	]*

[The Map proof of TransEquiv1 reads

L01:	Hypothesis $\gg$	$\ell_2 p$	;
L02:	TypeT $\gg$	$\ell\,\text{'}\,\mathsf{T}$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\ell\,\text{'}\,b$	;
L05:	Hypothesis $\gg$	$\ell\,\text{'}\,c$	;
L06:	TypeEquiv $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\gg$	$!\mathsf{T} =_p b$	;
L07:	TypeEquiv $\trianglerighteq$ L1 $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\gg$	$!b =_p c$	;
L08:	TypeEquiv $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L5 $\gg$	$!\mathsf{T} =_p c$	;
L09:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$!(\mathsf{T} =_p b \wedge b =_p c)$	;
L10:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\trianglerighteq$ L8 $\gg$	$\mathsf{T} =_p b \wedge b =_p c \rightarrow \mathsf{T} =_p c$	;
L11:	CDeduction $\triangleright$ L9 $\triangleright$ L8 $\triangleright$ L10 $\gg$	$\mathsf{T} =_p b \wedge b =_p c \Rightarrow \mathsf{T} =_p c$	;
L12:	Block $\gg$	End	;
L13:	Gen2 $\triangleright$ L11 $\gg$	$\forall b$: $\forall c$: $(\mathsf{T}=_p b \wedge b=_p c \Rightarrow \mathsf{T}=_p c)$	]*

[The Map proof of TransEquiv2 reads

L01:	Hypothesis ≫	$\ell_2 p$	;
L02:	Hypothesis ≫	$\neg a$	;
L03:	Hypothesis ≫	$\ell\,'\,a$	;
L04:	Hypothesis ≫	$\forall y\colon \forall a\colon \forall b\colon \forall c\colon (a\,'\,y =_p b \wedge b =_p c \Rightarrow a\,'\,y =_p c)$	;
L05:	Block ≫	Begin	;
L06:	Hypothesis ≫	$\quad \ell\,'\,a$	;
L07:	Hypothesis ≫	$\quad \ell\,'\,b$	;
L08:	Hypothesis ≫	$\quad \ell\,'\,c$	;
L09:	TypeEquiv ⊵ L1 ⊵ L6 ⊵ L7 ≫	$\quad !a =_p b$	;
L10:	TypeEquiv ⊵ L1 ⊵ L6 ⊵ L8 ≫	$\quad !a =_p c$	;
L11:	TypeEquiv ⊵ L1 ⊵ L7 ⊵ L8 ≫	$\quad !b =_p c$	;
L12:	Logic ⊵ L9 ⊵ L10 ≫	$\quad !(a =_p b \wedge b =_p c)$	;
L13:	Block ≫	$\quad$ Begin	;
L14:	Hypothesis ≫	$\quad\quad a =_p b \wedge b =_p c$	;
L15:	Logic ⊵ L2 ⊵ L14 ≫	$\quad\quad \neg b$	;
L16:	Logic ⊵ L2 ⊵ L14 ≫	$\quad\quad \neg c$	;
L17:	Block ≫	$\quad\quad$ Begin	;
L18:	Hypothesis ≫	$\quad\quad\quad \ell\,'\,u$	;
L19:	Hypothesis ≫	$\quad\quad\quad \ell\,'\,v$	;
L20:	Logic ⊵ L14 ≫	$\quad\quad\quad a =_p b$	;
L21:	ElimEquiv⊵L18⊵L19⊵L20≫	$\quad\quad\quad a\,'(p'u'v) =_p b\,'(p'u'v)$	;
L22:	Logic ⊵ L14 ≫	$\quad\quad\quad b =_p c$	;
L23:	ElimEquiv⊵L18⊵L19⊵L22≫	$\quad\quad\quad b\,'(p'u'v) =_p c\,'(p'u'v)$	;
L24:	ElimAll2 ⊵ L1 ⊵ L18 ⊵ L19 ≫	$\quad\quad\quad \ell\,'(p'u'v)$	;
L25:	TypeApply ⊵ L7 ⊵ L24 ≫	$\quad\quad\quad \ell\,'(b\,'(p'u'v))$	;
L26:	TypeApply ⊵ L8 ⊵ L24 ≫	$\quad\quad\quad \ell\,'(c\,'(p'u'v))$	;
L27:	ElimAll4 ⊵ L4 ⊵ L24 ⊵ L6 ⊵ L25 ⊵ L26 ≫	$\quad\quad\quad a\,'(p'u'v) =_p b\,'(p'u'v) \wedge b\,'(p'u'v) =_p c\,'(p'u'v) \Rightarrow a\,'(p'u'v) =_p c\,'(p'u'v)$	;
L28:	Logic ⊵ L21 ⊵ L23 ⊵ L27 ≫	$\quad\quad\quad a\,'(p'u'v) =_p c\,'(p'u'v)$	;
L29:	Block ≫	$\quad\quad$ End	;
L30:	IntroEquiv▷L2▷L16▷L28≫	$\quad\quad a =_p c$	;
L31:	Block ≫	$\quad$ End	;
L32:	CDeduction▷L12▷L11▷L30≫	$\quad a =_p b \wedge b =_p c \Rightarrow a =_p c$	;
L33:	Block ≫	End	;
L34:	Gen2 ▷ L32 ≫	$\forall b\colon \forall c\colon (a{=_p}b{\wedge}b{=_p}c{\Rightarrow}a{=_p}c)$	]*

[The Map proof of TransEquiv reads

L1:	Hypothesis $\gg$	$\ell_2 p$	;
L2:	Hypothesis $\gg$	ℓ'a	;
L3:	Hypothesis $\gg$	ℓ'b	;
L4:	Hypothesis $\gg$	ℓ'c	;
L5:	Hypothesis $\gg$	$a =_p b$	;
L6:	Hypothesis $\gg$	$b =_p c$	;
L7:	Transfinite' $\rhd$ (TransEquiv1 $\unrhd$ L1)$\rhd$ (TransEquiv2 $\unrhd$ L1) $\gg$	$\forall a\colon \forall b\colon \forall c\colon (a{=_p}b \wedge b{=_p}c \Rightarrow a{=_p}c)$	;
L8:	ElimAll3 $\unrhd$ L7 $\unrhd$ L2 $\unrhd$ L3 $\unrhd$ L4 $\gg$	$a{=_p}b \wedge b{=_p}c \Rightarrow a{=_p}c$	;
L9:	Logic $\unrhd$ L5 $\unrhd$ L6 $\unrhd$ L8 $\gg$	$a =_p c$	]*

A.34 Lemmas about $[\, x \sim_p y \,]$

[The Map proof of TypeEquivP reads

L1:	Hypothesis $\gg$	ℓ'p	;
L2:	Hypothesis $\gg$	ℓ'x	;
L3:	Hypothesis $\gg$	ℓ'y	;
L4:	SubtypeLL2 $\unrhd$ L1 $\gg$	$\ell_2 p$	;
L5:	TypeEquiv $\unrhd$ L4 $\unrhd$ L2 $\unrhd$ L3 $\gg$	$!x =_p y$	;
L6:	Logic $\unrhd$ L4 $\unrhd$ L2 $\unrhd$ L3 $\unrhd$ L5 $\gg$	$!x \sim_p y$	]*

[The Map proof of StrictEquivPX reads

L1:	Hypothesis $\gg$	$!x \sim_p y$	;
L2:	Logic $\unrhd$ L1 $\gg$	$!\ell$'x	;
L3:	StrictEll $\unrhd$ L2 $\gg$	ℓ'x	]*

[The Map proof of StrictEquivPY reads

L1:	Hypothesis $\gg$	$!x \sim_p y$	;
L2:	Logic $\unrhd$ L1 $\gg$	$!\ell$'y	;
L3:	StrictEll $\unrhd$ L2 $\gg$	ℓ'y	]*

[The Map proof of StrictEquivPP reads

L1:	Hypothesis $\gg$	$!x \sim_p y$	;
L2:	Logic $\unrhd$ L1 $\gg$	$!\ell$'p	;
L3:	StrictEll $\unrhd$ L2 $\gg$	ℓ'p	]*

[The Map proof of RefEquivP reads

L1:	Hypothesis $\gg$	ℓ'p	;
L2:	Hypothesis $\gg$	ℓ'a	;
L3:	SubtypeLL2 $\unrhd$ L1 $\gg$	$\ell_2 p$	;
L4:	RefEquiv $\unrhd$ L3 $\unrhd$ L2 $\gg$	$a =_p a$	;
L5:	Logic $\unrhd$ L2 $\unrhd$ L4 $\gg$	$a \sim_p a$	]*

[The Map proof of ComEquivP reads

L1:	Hypothesis $\gg$	$a \sim_p b$	;
L2:	Logic $\trianglerighteq$ L1 $\gg$	$\ell\text{'}p$	;
L3:	SubtypeLL2 $\trianglerighteq$ L2 $\gg$	$\ell_2 p$	;
L4:	Logic $\trianglerighteq$ L1 $\gg$	$\ell\text{'}a$	;
L5:	Logic $\trianglerighteq$ L1 $\gg$	$\ell\text{'}b$	;
L6:	Logic $\trianglerighteq$ L1 $\gg$	$a =_p b$	;
L7:	ComEquiv $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L $\trianglerighteq$ L6 $\gg$	$b =_p a$	;
L8:	Logic $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L7 $\gg$	$b \sim_p a$	]*

[The Map proof of TransEquivP reads

L01:	Hypothesis $\gg$	$a \sim_p b$	;
L02:	Hypothesis $\gg$	$b \sim_p c$	;
L03:	Logic $\trianglerighteq$ L1 $\gg$	$\ell\text{'}p$	;
L04:	SubtypeLL2 $\trianglerighteq$ L3 $\gg$	$\ell_2 p$	;
L05:	Logic $\trianglerighteq$ L1 $\gg$	$\ell\text{'}a$	;
L06:	Logic $\trianglerighteq$ L2 $\gg$	$\ell\text{'}b$	;
L07:	Logic $\trianglerighteq$ L2 $\gg$	$\ell\text{'}c$	;
L08:	Logic $\trianglerighteq$ L1 $\gg$	$a =_p b$	;
L09:	Logic $\trianglerighteq$ L2 $\gg$	$b =_p c$	;
L10:	TransEquiv $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\trianglerighteq$ L8 $\trianglerighteq$ L9 $\gg$	$a =_p c$	;
L11:	Logic $\trianglerighteq$ L3 $\trianglerighteq$ L5 $\trianglerighteq$ L7 $\trianglerighteq$ L10 $\gg$	$a \sim_p c$	]*

A.35 Lemmas about $[\, x \sim y \,]$

[The Map proof of TypeEquivK reads

L1:	Hypothesis $\gg$	$\ell\text{'}x$	;
L2:	Hypothesis $\gg$	$\ell\text{'}y$	;
L3:	TypeK $\gg$	$\ell_2\mathsf{K}$	;
L4:	TypeEquiv $\trianglerighteq$ L3 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$!x =_{\mathsf{K}} y$	;
L5:	Logic $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\gg$	$!x \sim y$	]*

[The Map proof of StrictEquivKX reads

L1:	Hypothesis $\gg$	$!x \sim y$	;
L2:	Logic $\trianglerighteq$ L1 $\gg$	$!\ell\text{'}x$	;
L3:	StrictEll $\trianglerighteq$ L2 $\gg$	$\ell\text{'}x$	]*

[The Map proof of StrictEquivKY reads

L1:	Hypothesis $\gg$	$!x \sim y$	;
L2:	Logic $\trianglerighteq$ L1 $\gg$	$!\ell\text{'}y$	;
L3:	StrictEll $\trianglerighteq$ L2 $\gg$	$\ell\text{'}y$	]*

[The Map proof of ElimEquivK reads

L01:	Hypothesis $\gg$	$\ell\,\text{'}\,u$	;
L02:	Hypothesis $\gg$	$a \sim b$	;
L03:	Logic $\trianglerighteq$ L2 $\gg$	$\ell\,\text{'}\,a$	;
L04:	Logic $\trianglerighteq$ L2 $\gg$	$\ell\,\text{'}\,b$	;
L05:	TypeApply $\trianglerighteq$ L3 $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}(a\,\text{'}\,u)$	;
L06:	TypeApply $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}(b\,\text{'}\,u)$	;
L07:	Logic $\trianglerighteq$ L3 $\gg$	$a =_{\mathsf{K}} b$	;
L08:	ElimEquiv $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\trianglerighteq$ L8 $\gg$	$a\,\text{'}(\mathsf{K}\,\text{'}\,u\,\text{'}\,u) =_{\mathsf{K}} (\mathsf{K}\,\text{'}\,u\,\text{'}\,u)$	;
L09:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L8 $\gg$	$a\,\text{'}\,u =_{\mathsf{K}} b\,\text{'}\,u$	;
L10:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\trianglerighteq$ L9 $\gg$	$a\,\text{'}\,u \sim b\,\text{'}\,u$	]*

[The Map proof of RefEquivK reads

L1:	Hypothesis $\gg$	$\ell\,\text{'}\,a$	;
L2:	TypeK $\gg$	$\ell_2\mathsf{K}$	;
L3:	RefEquiv $\trianglerighteq$ L2 $\trianglerighteq$ L1 $\gg$	$a =_{\mathsf{K}} a$	;
L4:	Logic $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$a \sim a$	]*

[The Map proof of ComEquivK reads

L1:	Hypothesis $\gg$	$a \sim b$	;
L2:	TypeK $\gg$	$\ell_2\mathsf{K}$	;
L3:	Logic $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}\,a$	;
L4:	Logic $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}\,b$	;
L5:	Logic $\trianglerighteq$ L1 $\gg$	$a =_{\mathsf{K}} b$	;
L6:	ComEquiv $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\gg$	$b =_{\mathsf{K}} a$	;
L7:	Logic $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\trianglerighteq$ L6 $\gg$	$b \sim a$	]*

[The Map proof of TransEquivK reads

L01:	Hypothesis $\gg$	$a \sim b$	;
L02:	Hypothesis $\gg$	$b \sim c$	;
L03:	TypeK $\gg$	$\ell_2\mathsf{K}$	;
L04:	Logic $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}\,a$	;
L05:	Logic $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}\,b$	;
L06:	Logic $\trianglerighteq$ L2 $\gg$	$\ell\,\text{'}\,c$	;
L07:	Logic $\trianglerighteq$ L1 $\gg$	$a =_{\mathsf{K}} b$	;
L08:	Logic $\trianglerighteq$ L2 $\gg$	$b =_{\mathsf{K}} c$	;
L09:	TransEquiv $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\trianglerighteq$ L8 $\gg$	$a =_{\mathsf{K}} c$	;
L10:	Logic $\trianglerighteq$ L4 $\trianglerighteq$ L6 $\trianglerighteq$ L9 $\gg$	$a \sim c$	]*

A.36 Lemmas about $[\, x \in_\ell p \,]$

[The Map proof of TypeLim reads

```
L01: Hypothesis $\gg$                                  $\ell_1 x$                                                        ;
L02: Hypothesis $\gg$                                  $\ell\,'p$                                                        ;
L03: SubtypeLL2 $\unrhd$ L2 $\gg$                      $\ell_2 p$                                                        ;
L04: TypeK $\gg$                                       $\ell_2 \mathsf{K}$                                               ;
L05: Block $\gg$                                       Begin                                                             ;
L06: Hypothesis $\gg$                                    $\ell\,'u$                                                      ;
L07: Block $\gg$                                         Begin                                                           ;
L08: Hypothesis $\gg$                                      $\ell\,'v$                                                    ;
L09: TypeEquiv $\unrhd$ L3 $\unrhd$ L6 $\unrhd$ L8 $\gg$   $!u =_p v$                                                    ;
L10: ElimAll $\unrhd$ L1 $\unrhd$ L6 $\gg$                 $\ell\,'(x\,'u)$                                              ;
L11: ElimAll $\unrhd$ L1 $\unrhd$ L8 $\gg$                 $\ell\,'(x\,'v)$                                              ;
L12: TypeEquiv$\unrhd$L4$\unrhd$L10$\unrhd$L11 $\gg$       $!x\,'u =_{\mathsf{K}} x\,'v$                                 ;
L13: Logic $\unrhd$ L9 $\unrhd$ L12 $\gg$                  $!(u =_p v \Rightarrow x\,'u =_{\mathsf{K}} x\,'v)$           ;
L14: Block $\gg$                                         End                                                             ;
L15: TypeAll $\unrhd$ (Gen $\rhd$ L13) $\gg$             $!\forall v\colon (u =_p v \Rightarrow x\,'u =_{\mathsf{K}} x\,'v)$   ;
L16: Block $\gg$                                       End                                                               ;
L17: TypeAll $\unrhd$ (Gen $\rhd$ L15) $\gg$           $!\forall u \forall v\colon (u =_p v \Rightarrow x\,'u =_{\mathsf{K}} x\,'v)$   ;
L18: Repetition $\rhd$ L17 $\gg$                       $!x \in_l p$                                                      ]*
```

[The Map proof of TypeInEll reads

```
L1: Hypothesis $\gg$                              $\ell_1 x$        ;
L2: Hypothesis $\gg$                              $\ell\,'p$        ;
L3: TypeLim $\unrhd$ L1 $\unrhd$ L2 $\gg$         $!x \in_l p$      ;
L4: Logic $\unrhd$ L1 $\unrhd$ L2 $\unrhd$ L3 $\gg$   $!x \in_\ell p$   ]*
```

[The Map proof of StrictInEllX reads

```
L1: Hypothesis $\gg$        $!x \in_\ell p$   ;
L2: Logic $\unrhd$ L1 $\gg$   $\ell_1 x$        ]*
```

[The Map proof of StrictInEllP reads

```
L1: Hypothesis $\gg$        $!x \in_\ell p$   ;
L2: Logic $\unrhd$ L1 $\gg$   $\ell\,'p$        ]*
```

[The Map proof of ElimInEll reads

L01:	Hypothesis $\gg$	$x \in_\ell p$	;
L02:	Hypothesis $\gg$	$u \sim_p v$	;
L03:	StrictEquivPX $\trianglerighteq$ L2 $\gg$	$\ell\,\text{'}\,u$	;
L04:	StrictEquivPY $\trianglerighteq$ L2 $\gg$	$\ell\,\text{'}\,v$	;
L05:	Logic $\trianglerighteq$ L1 $\gg$	$x \in_l p$	;
L06:	ElimAll2 $\trianglerighteq$ L5 $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$u =_p v \Rightarrow x\,\text{'}\,u =_K x\,\text{'}\,v$	;
L07:	Logic $\trianglerighteq$ L2 $\gg$	$u =_p v$	;
L08:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$x\,\text{'}\,u =_K x\,\text{'}\,v$	;
L09:	StrictInEllX $\trianglerighteq$ L1 $\gg$	$\ell_1 x$	;
L10:	ElimAll $\trianglerighteq$ L9 $\trianglerighteq$ L3 $\gg$	$\ell\,\text{'}(x\,\text{'}\,u)$	;
L11:	ElimAll $\trianglerighteq$ L9 $\trianglerighteq$ L4 $\gg$	$\ell\,\text{'}(x\,\text{'}\,v)$	;
L12:	Logic $\trianglerighteq$ L8 $\trianglerighteq$ L9 $\trianglerighteq$ L10 $\trianglerighteq$ L11 $\gg$	$x\,\text{'}\,u \sim x\,\text{'}\,v$	]*

[The Map proof of IntroInEll reads

L01:	Premise $\gg$	$\ell\,\text{'}\,p$	;
L02:	Premise $\gg$	$u \sim_p v \to x\,\text{'}\,u \sim x\,\text{'}\,v$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\quad \ell\,\text{'}\,u$	;
L05:	RefEquivP $\trianglerighteq$ L1 $\trianglerighteq$ L4 $\gg$	$\quad u \sim_p u$	;
L06:	L2 $\trianglerighteq$ L5 $\gg$	$\quad x\,\text{'}\,u \sim x\,\text{'}\,u$	;
L07:	StrictEquivKX $\trianglerighteq$ L6 $\gg$	$\quad \ell\,\text{'}(x\,\text{'}\,u)$	;
L08:	Block $\gg$	End	;
L09:	Gen $\triangleright$ L $\gg$	$\ell_1 x$	;
L10:	Block $\gg$	Begin	;
L11:	Hypothesis $\gg$	$\quad \ell\,\text{'}\,u$	;
L12:	Hypothesis $\gg$	$\quad \ell\,\text{'}\,v$	;
L13:	TypeEquiv $\trianglerighteq$ L1 $\trianglerighteq$ L11 $\trianglerighteq$ L12 $\gg$	$\quad !u =_p v$	;
L14:	TypeK $\gg$	$\quad \ell_2 \mathsf{K}$	;
L15:	ElimAll $\trianglerighteq$ L9 $\trianglerighteq$ L11 $\gg$	$\quad \ell\,\text{'}(x\,\text{'}\,u)$	;
L16:	ElimAll $\trianglerighteq$ L9 $\trianglerighteq$ L12 $\gg$	$\quad \ell\,\text{'}(x\,\text{'}\,v)$	;
L17:	TypeEquiv $\trianglerighteq$ L14 $\trianglerighteq$ L15 $\trianglerighteq$ L16 $\gg$	$\quad !x\,\text{'}\,u =_{\mathsf{K}} x\,\text{'}\,v$	;
L18:	Block $\gg$	$\quad$ Begin	;
L19:	Hypothesis $\gg$	$\qquad u =_p v$	;
L20:	Logic $\trianglerighteq$ L11 $\trianglerighteq$ L12 $\trianglerighteq$ L19 $\gg$	$\qquad u \sim_p v$	;
L21:	L2 $\trianglerighteq$ L20 $\gg$	$\qquad x\,\text{'}\,u \sim x\,\text{'}\,v$	;
L22:	Block $\gg$	$\quad$ End	;
L23:	CDeduction $\triangleright$ L13 $\triangleright$ L17 $\triangleright$ L21 $\gg$	$\quad u \sim_p v \Rightarrow x\,\text{'}\,u \sim x\,\text{'}\,v$	;
L24:		;	
L25:	Gen2 $\triangleright$ L23 $\gg$	$\quad x \in_l p$	;
L26:	Logic $\trianglerighteq$ L9 $\trianglerighteq$ L1 $\trianglerighteq$ L25 $\gg$	$\quad x \in_\ell p$	]*

[The Map proof of IntroInEllT reads

L1:	Hypothesis $\gg$	$\ell\,'\,p$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad u \sim_p v$	;
L4:	Reduction $\gg$	$\quad \mathsf{T}\,'\,u \sim \mathsf{T}\,'\,v$	;
L5:	Block $\gg$	End	;
L6:	IntroInEll $\triangleright$ L1 $\triangleright$ L4 $\gg$	$\mathsf{T} \in_\ell p$	]*

A.37 Lemmas about $[\,\ell\,]$

[The Map proof of ElimEll reads

L01:	Hypothesis $\gg$	$\ell\,'\,x$	;
L02:	SubtypeLL1 $\triangleright$ L1 $\gg$	$\ell_1 x$	;
L03:	SubtypeLB $\triangleright$ L1 $\gg$	$!x$	;
L04:	TypeLim $\trianglerighteq$ L2 $\gg$	$\ell\,'\,p \to\, !x \in_l p$	;
L05:	Gen $\triangleright$ L4 $\gg$	$\forall p\colon !x \in_l p$	;
L06:	Block $\gg$	Begin	;
L07:	Hypothesis $\gg$	$\quad x$	;
L08:	TypeT $\gg$	$\quad \ell\,'\,\mathsf{T}$	;
L09:	Block $\gg$	$\quad$ Begin	;
L10:	Hypothesis $\gg$	$\quad\quad \ell\,'\,u$	;
L11:	Hypothesis $\gg$	$\quad\quad \ell\,'\,v$	;
L12:	SubtypeLL2 $\trianglerighteq$ L8 $\gg$	$\quad\quad \ell_2 \mathsf{T}$	;
L13:	TypeEquiv $\trianglerighteq$ L12 $\trianglerighteq$ L10 $\trianglerighteq$ L11 $\gg$	$\quad\quad !u =_{\mathsf{T}} v$	;
L14:	Logic $\trianglerighteq$ L7 $\trianglerighteq$ L13 $\gg$	$\quad\quad u =_{\mathsf{T}} v \Rightarrow x\,'\,u =_{\mathsf{K}} x\,'\,v$	;
L15:	Block $\gg$	$\quad$ End	;
L16:	Gen2 $\triangleright$ L14 $\gg$	$\quad x \in_l \mathsf{T}$	;
L17:	IntroExists $\trianglerighteq$ L5 $\trianglerighteq$ L8 $\trianglerighteq$ L16 $\gg$	$\quad \exists p\colon x \in_l p$	;
L18:	Block $\gg$	End	;
L19:	Block $\gg$	Begin	;
L20:	Hypothesis $\gg$	$\quad \neg x$	;
L21:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L1 $\gg$	$\quad \mathrm{if}(x, \mathsf{T}, \forall y\colon \ell\,'(x\,'\,y) \wedge$ $\mathsf{E}p\colon \ell\,'\,p \wedge x \in_l p)$	;
L22:	Logic $\trianglerighteq$ L21 $\gg$	$\quad \mathsf{E}p\colon \ell\,'\,p \wedge x \in_l p$	;
L23:	Block $\gg$	$\quad$ Begin	;
L24:	Hypothesis $\gg$	$\quad\quad \ell\,'\,p \wedge x \in_l p$	;
L25:	Logic $\trianglerighteq$ L24 $\gg$	$\quad\quad \ell\,'\,p$	;
L26:	Logic $\trianglerighteq$ L24 $\gg$	$\quad\quad x \in_l p$	;
L27:	IntroExists $\trianglerighteq$ L5 $\trianglerighteq$ L25 $\trianglerighteq$ L26 $\gg$	$\quad\quad \exists p\colon x \in_l p$	;
L28:	Block $\gg$	$\quad$ End	;
L29:	ElimPure $\triangleright$ L22 $\triangleright$ L27 $\gg$	$\quad \exists p\colon x \in_l p$	;
L30:	Block $\gg$	End	;
L31:	TND $\triangleright$ L3 $\triangleright$ L17 $\triangleright$ L29 $\gg$	$\exists p\colon x \in_l p$	;
L32:	ElimExistsEll $\trianglerighteq$ L31 $\gg$	$\ell\,'\,\mathcal{I}x$	;
L33:	ElimExists $\trianglerighteq$ L31 $\gg$	$x \in_l \mathcal{I}x$	;
L34:	Logic $\triangleright$ L2 $\trianglerighteq$ L32 $\trianglerighteq$ L33 $\gg$	$x \in_\ell \mathcal{I}x$	]*

[The Map proof of IntroEll reads

L1:	Hypothesis $\gg$	$x \in_\ell p$	;
L2:	StrictInEllX $\trianglerighteq$ L1 $\gg$	$\ell_1 x$	;
L3:	SubtypeL1B $\trianglerighteq$ L2 $\gg$	$!x$	;
L4:	Logic $\trianglerighteq$ L1 $\gg$	$\ell\text{'}p \wedge x \in_l p$	;
L5:	IntroPure(p) $\trianglerighteq$ L4 $\gg$	$\mathsf{E}p\colon \ell\text{'}p \wedge x \in_l p$	;
L6:	Logic $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\trianglerighteq$ L5 $\gg$	$\text{if}(x, \mathsf{T}, \forall y\colon \ell\text{'}(x\text{'}y) \wedge$ $\mathsf{E}p\colon \ell\text{'}p \wedge x \in_l p$	;
L7:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L6 $\gg$	$\ell\text{'}x$	]*

A.38 Classical Extensionality

[The Map proof of IntroEquivK reads

L01:	Hypothesis $\gg$	$\neg a$	;
L02:	Hypothesis $\gg$	$\neg b$	;
L03:	Hypothesis $\gg$	$\ell\text{'}a$	;
L04:	Hypothesis $\gg$	$\forall x\colon a\text{'}x \sim b\text{'}x$	;
L05:	Logic $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\gg$	$\text{if}(a, \text{if}(b, \mathsf{T}, \mathsf{F}), \text{if}(b, \mathsf{F}, \forall x\colon a\text{'}x{\sim}b\text{'}x))$	;
L06:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L5 $\gg$	$a =_{\mathsf{K}} b$	;
L07:	ElimEll $\trianglerighteq$ L3 $\gg$	$a \in_\ell \mathcal{I}a$	;
L08:	StrictInEllP $\trianglerighteq$ L7 $\gg$	$\ell\text{'}\mathcal{I}a$	;
L09:	Block $\gg$	Begin	;
L10:	Hypothesis $\gg$	$u \sim_{\mathcal{I}a} v$	;
L11:	StrictEquivPX $\trianglerighteq$ L10 $\gg$	$\ell\text{'}u$	;
L12:	StrictEquivPY $\trianglerighteq$ L10 $\gg$	$\ell\text{'}v$	;
L13:	ElimInEll $\trianglerighteq$ L7 $\trianglerighteq$ L10 $\gg$	$a\text{'}u \sim a\text{'}v$	;
L14:	ElimAll $\trianglerighteq$ L4 $\trianglerighteq$ L11 $\gg$	$a\text{'}u \sim b\text{'}u$	;
L15:	ElimAll $\trianglerighteq$ L4 $\trianglerighteq$ L12 $\gg$	$a\text{'}v \sim b\text{'}v$	;
L16:	ComEquivK $\trianglerighteq$ L13 $\gg$	$b\text{'}u \sim a\text{'}u$	;
L17:	TransEquivK$\trianglerighteq$L16$\trianglerighteq$L13 $\gg$	$b\text{'}u \sim a\text{'}v$	;
L18:	TransEquivK$\trianglerighteq$L17$\trianglerighteq$L15 $\gg$	$b\text{'}u \sim b\text{'}v$	;
L19:	Block $\gg$	End	;
L20:	IntroInEll $\triangleright$ L8 $\triangleright$ L18 $\gg$	$b \in_\ell \mathcal{I}a$	;
L21:	IntroEll $\trianglerighteq$ L20 $\gg$	$\ell\text{'}b$	;
L22:	Logic $\trianglerighteq$ L3 $\trianglerighteq$ L6 $\trianglerighteq$ L21 $\gg$	$a \sim b$	]*

[The Map proof of EllOrderA reads

L01:	Block $\gg$	Begin	;
L02:	Hypothesis $\gg$	$\ell\,'\,y$	;
L03:	SubtypeLB $\trianglerighteq$ L2 $\gg$	$!y$	;
L04:	Logic $\trianglerighteq$ L3 $\gg$	$\mathsf{T} \sim y \dot{\Rightarrow} \mathsf{T} \sim \mathsf{T} \downarrow y$	;
L05:	Block $\gg$	End	;
L06:	Gen $\triangleright$ L4 $\gg$	$\forall y\colon (\mathsf{T} \sim y \dot{\Rightarrow} \mathsf{T} \sim \mathsf{T} \downarrow y)$	;
L07:	Block $\gg$	Begin	;
L08:	Hypothesis $\gg$	$\neg x$	;
L09:	Hypothesis $\gg$	$\ell\,'\,x$	;
L10:	Hypothesis $\gg$	$\forall z\colon\forall y\colon (x\,'\,z \sim y \dot{\Rightarrow}$ $x\,'\,z \sim x\,'\,z \downarrow y$	;
L11:	Block $\gg$	Begin	;
L12:	Hypothesis $\gg$	$\ell\,'\,y$	;
L13:	TypeEquivK $\trianglerighteq$ L9 $\trianglerighteq$ L12 $\gg$	$!x \sim y$	;
L14:	Block $\gg$	Begin	;
L15:	Hypothesis $\gg$	$x \sim y$	;
L16:	Logic $\trianglerighteq$ L8 $\trianglerighteq$ L15 $\gg$	$\neg y$	;
L17:	Block $\gg$	Begin	;
L18:	Hypothesis $\gg$	$\ell\,'\,z$	;
L19:	TypeApply $\trianglerighteq$ L12 $\trianglerighteq$ L18 $\gg$	$\ell\,'(y\,'\,z)$	;
L20:	ElimAll2 $\trianglerighteq$ L10 $\trianglerighteq$ L18 $\trianglerighteq$ L19 $\gg$	$x\,'\,z \sim y\,'\,z \dot{\Rightarrow}$ $x\,'\,z \sim x\,'\,z \downarrow y\,'\,z$	;
L21:	ElimEquivK $\trianglerighteq$ L18 $\trianglerighteq$ L15 $\gg$	$x\,'\,z \sim y\,'\,z$	;
L22:	Logic $\trianglerighteq$ L20 $\trianglerighteq$ L21 $\gg$	$x\,'\,z \sim x\,'\,z \downarrow y\,'\,z$	;
L23:	Logic $\trianglerighteq$ L8 $\trianglerighteq$ L16 $\gg$	$x'z \downarrow y'z \equiv (x \downarrow y)'z$	;
L24:	Replace' $\triangleright$ L23 $\triangleright$ L22 $\gg$	$x\,'\,z \sim (x \downarrow y)\,'\,z$	;
L25:	Block $\gg$	End	;
L26:	Gen $\triangleright$ L24 $\gg$	$\forall z\colon x\,'\,z \sim (x \downarrow y)\,'\,z$	;
L27:	Logic $\trianglerighteq$ L8 $\trianglerighteq$ L16 $\gg$	$\neg(x \downarrow y)$	;
L28:	IntroEquivK$\trianglerighteq$L8$\trianglerighteq$L27$\trianglerighteq$L9$\trianglerighteq$L26$\gg$	$x \sim x \downarrow y$	;
L29:	Block $\gg$	End	;
L30:	Deduction" $\triangleright$ L13 $\triangleright$ L28 $\gg$	$x \sim y \dot{\Rightarrow} x \sim x \downarrow y$	;
L31:	Block $\gg$	End	;
L32:	Gen $\triangleright$ L30 $\gg$	$\forall y\colon (x \sim y \dot{\Rightarrow} x \sim x \downarrow y)$	;
L33:	Block $\gg$	End	;
L34:	Transfinite' $\triangleright$ L6 $\triangleright$ L32 $\gg$	$\forall x\colon\forall y\colon (x \sim y \dot{\Rightarrow} x \sim x \downarrow y)$	;
L35:	Hypothesis $\gg$	$x \sim y$	;
L36:	StrictEquivKX $\trianglerighteq$ L35 $\gg$	$\ell\,'\,x$	;
L37:	StrictEquivKY $\trianglerighteq$ L35 $\gg$	$\ell\,'\,y$	;
L38:	ElimAll2 $\trianglerighteq$ L34 $\trianglerighteq$ L36 $\trianglerighteq$ L37 $\gg$	$x \sim y \dot{\Rightarrow} x \sim x \downarrow y$	;
L39:	Logic $\trianglerighteq$ L36 $\trianglerighteq$ L38 $\gg$	$x \sim x \downarrow y$	;
L40:	StrictEquivKY $\trianglerighteq$ L39 $\gg$	$\ell\,'(x \downarrow y)$	]*

[The Map proof of Ext reads

L01:	Hypothesis $\gg$	$\ell_1 f$	;
L02:	Hypothesis $\gg$	$x \sim y$	;
L03:	EllOrderA $\trianglerighteq$ L2 $\gg$	$\ell\text{'}(x \downarrow y)$	;
L04:	ElimAll $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$\ell\text{'}(f\text{'}(x \downarrow y))$	;
L05:	RefEquivK $\trianglerighteq$ L4 $\gg$	$f\text{'}(x \downarrow y) \sim f\text{'}(x \downarrow y)$	;
L06:	L7.7.1 $\gg$	$x \downarrow y \preceq x$	;
L07:	L7.7.2 $\gg$	$x \downarrow y \preceq y$	;
L08:	Mono $\triangleright$ L6 $\gg$	$f\text{'}(x{\downarrow}y) \sim f\text{'}(x{\downarrow}y) \preceq f\text{'}x \sim f\text{'}(x{\downarrow}y)$	;
L09:	Mono $\triangleright$ L7 $\gg$	$f\text{'}(x{\downarrow}y) \sim f\text{'}(x{\downarrow}y) \preceq f\text{'}(x{\downarrow}y) \sim f\text{'}y$	;
L10:	MP' $\triangleright$ L5 $\triangleright$ L8 $\gg$	$f\text{'}x \sim f\text{'}(x \downarrow y)$	;
L11:	MP' $\triangleright$ L5 $\triangleright$ L9 $\gg$	$f\text{'}(x \downarrow y) \sim f\text{'}y$	;
L12:	TransEquivK$\trianglerighteq$L10$\trianglerighteq$L11$\gg$	$f\text{'}x \sim f\text{'}y$	]*

[The Map proof of ExtBool reads

L01:	Hypothesis $\gg$	$!_1 f$	;
L02:	Hypothesis $\gg$	$x \sim y$	;
L03:	EllOrderA $\trianglerighteq$ L2 $\gg$	$\ell\text{'}(x \downarrow y)$	;
L04:	ElimAll $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$!f\text{'}(x \downarrow y)$	;
L05:	Logic $\trianglerighteq$ L4 $\gg$	$f\text{'}(x \downarrow y) \Leftrightarrow f\text{'}(x \downarrow y)$	;
L06:	L7.7.1 $\gg$	$x \downarrow y \preceq x$	;
L07:	L7.7.2 $\gg$	$x \downarrow y \preceq y$	;
L08:	Mono $\triangleright$ L6 $\gg$	$f\text{'}(x{\downarrow}y) \Leftrightarrow f\text{'}(x{\downarrow}y) \preceq f\text{'}x \Leftrightarrow f\text{'}(x{\downarrow}y)$	;
L09:	Mono $\triangleright$ L7 $\gg$	$f\text{'}(x{\downarrow}y) \Leftrightarrow f\text{'}(x{\downarrow}y) \preceq f\text{'}(x{\downarrow}y) \Leftrightarrow f\text{'}y$	;
L10:	MP' $\triangleright$ L5 $\triangleright$ L8 $\gg$	$f\text{'}x \Leftrightarrow f\text{'}(x \downarrow y)$	;
L11:	MP' $\triangleright$ L5 $\triangleright$ L9 $\gg$	$f\text{'}(x \downarrow y) \Leftrightarrow f\text{'}y$	;
L12:	Logic$\trianglerighteq$L10$\trianglerighteq$L11$\gg$	$f\text{'}x \Leftrightarrow f\text{'}y$	]*

[The Map proof of ExtBool' reads

L01:	Hypothesis $\gg$	$!_1 f$	;
L02:	Hypothesis $\gg$	$x \sim y$	;
L03:	Hypothesis $\gg$	$f\text{'}x$	;
L04:	ExtBool $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$f\text{'}x \Leftrightarrow f\text{'}y$	;
L05:	Logic $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$f\text{'}y$	]*

[The Map proof of ExtBool" reads

L01:	Hypothesis $\gg$	$!_1 f$	;
L02:	Hypothesis $\gg$	$x \sim y$	;
L03:	Hypothesis $\gg$	$f\text{'}y$	;
L04:	ExtBool $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$f\text{'}x \Leftrightarrow f\text{'}y$	;
L05:	Logic $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$f\text{'}x$	]*

A.39 Primitive membership

[The Map proof of TypeInTwo reads

L01:	Hypothesis $\gg$	$\ell\text{'}\,x$	;
L02:	Hypothesis $\gg$	$\ell_2 y$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\quad \ell\text{'}\,s$	;
L05:	Block $\gg$	$\quad$ Begin	;
L06:	Hypothesis $\gg$	$\qquad \ell\text{'}\,t$	;
L07:	ElimAll2 $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\trianglerighteq$ L6 $\gg$	$\qquad \ell\text{'}(y\text{'}\,s\text{'}\,t)$	;
L08:	TypeEquivK $\trianglerighteq$ L1 $\trianglerighteq$ L7 $\gg$	$\qquad !x \sim y\text{'}\,s\text{'}\,t$	;
L09:	Block $\gg$	$\quad$ End	;
L10:	TypeExists $\trianglerighteq$ (Gen $\triangleright$ L8) $\gg$	$\quad !\exists t\colon x \sim y\text{'}\,s\text{'}\,t$	;
L11:	Block $\gg$	End	;
L12:	TypeExists $\trianglerighteq$ (Gen $\triangleright$ L10) $\gg$	$!\exists s\colon \exists t\colon x \sim y\text{'}\,s\text{'}\,t$	;
L13:	Repetition $\triangleright$ L12 $\gg$	$!x \in_2 y$	]*

[The Map proof of StrictInTwoX reads

L1:	Hypothesis $\gg$	$!x \in_2 y$	;
L2:	TypeT $\gg$	$\ell\text{'}\,\mathsf{T}$	;
L3:	Repetition $\triangleright$ L1 $\gg$	$!\exists s\colon \exists t\colon x \sim y\text{'}\,s\text{'}\,t$	;
L4:	ElimAll $\trianglerighteq$ (StrictExists $\trianglerighteq$ L3) $\trianglerighteq$ L2 $\gg$	$!\exists t\colon x \sim y\text{'}\,\mathsf{T}\text{'}\,t$	;
L5:	ElimAll $\trianglerighteq$ (StrictExists $\trianglerighteq$ L4) $\trianglerighteq$ L2 $\gg$	$!x \sim y\text{'}\,\mathsf{T}\text{'}\,\mathsf{T}$	;
L6:	StrictEquivKX $\trianglerighteq$ L5 $\gg$	$\ell\text{'}\,x$	]*

[The Map proof of StrictInTwoY reads

L01:	Hypothesis $\gg$	$!x \in_2 y$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\quad \ell\text{'}\,s$	;
L04:	Hypothesis $\gg$	$\quad \ell\text{'}\,t$	;
L05:	Repetition $\triangleright$ L1 $\gg$	$\quad !\exists s\colon \exists t\colon x \sim y\text{'}\,s\text{'}\,t$	;
L06:	ElimAll $\trianglerighteq$ (StrictExists $\trianglerighteq$ L5) $\trianglerighteq$ L3 $\gg$	$\quad !\exists t\colon x \sim y\text{'}\,s\text{'}\,t$	;
L07:	ElimAll $\trianglerighteq$ (StrictExists $\trianglerighteq$ L6) $\trianglerighteq$ L4 $\gg$	$\quad x \sim y\text{'}\,s\text{'}\,t$	;
L08:	StrictEquivKY $\trianglerighteq$ L7 $\gg$	$\quad \ell\text{'}(y\text{'}\,s\text{'}\,t)$	;
L09:	Block $\gg$	End	;
L10:	Gen2 $\triangleright$ L8 $\gg$	$\ell_2 y$	]*

[The Map proof of ElimInTwo reads

L1:	Hypothesis $\gg$	$x \in_2 y$	;
L2:	Repetition $\triangleright$ L1 $\gg$	$\exists s\colon \exists t\colon x \sim y\text{'}\,s\text{'}\,t$	;
L3:	Define $\gg$	$s \equiv \varepsilon s\colon \exists t\colon x \sim y\text{'}\,s\text{'}\,t$	;
L4:	ElimExists $\trianglerighteq$ L2 $\gg$	$\exists t\colon x \sim y\text{'}\,s\text{'}\,t$	;
L5:	Define $\gg$	$t \equiv \varepsilon t\colon x \sim y\text{'}\,s\text{'}\,t$	;
L6:	ElimExists $\trianglerighteq$ L4 $\gg$	$x \sim y\text{'}\,s\text{'}\,t$	;
L7:	Repetition $\triangleright$ L6 $\gg$	$x \sim y\text{'}\,s_2(x,y)\text{'}\,t_2(x,y)$	]*

[The Map proof of ElimInTwoS reads
```
L1:  Hypothesis $\gg$                      $x \in_2 y$                                  ;
L2:  Repetition $\rhd$ L1 $\gg$            $\exists s\colon \exists t\colon x \sim y\,'\,s\,'\,t$        ;
L3:  ElimExistsEll $\unrhd$ L2 $\gg$       $\ell\,'\,\varepsilon s\colon \exists t\colon x \sim y\,'\,s\,'\,t$   ;
L4:  Repetition $\gg$                      $\ell\,'\,s_2(x,y)$                          ]*
```

[The Map proof of ElimInTwoT reads
```
L1:  Hypothesis $\gg$                      $x \in_2 y$                                  ;
L2:  Repetition $\rhd$ L1 $\gg$            $\exists s\colon \exists t\colon x \sim y\,'\,s\,'\,t$        ;
L3:  Define $\gg$                          $s \equiv \varepsilon s\colon \exists t\colon x \sim y\,'\,s\,'\,t$   ;
L4:  ElimExists $\unrhd$ L2 $\gg$          $\exists t\colon x \sim y\,'\,s\,'\,t$                 ;
L5:  ElimExistsEll $\unrhd$ L4 $\gg$       $\ell\,'\,\varepsilon t\colon x \sim y\,'\,s\,'\,t$            ;
L6:  Repetition $\rhd$ L5 $\gg$            $\ell\,'\,t_2(x,y)$                          ]*
```

[The Map proof of IntroInTwo reads
```
L01: Hypothesis $\gg$                                                  $\ell_2 y$                                 ;
L02: Hypothesis $\gg$                                                  $\ell\,'\,s$                               ;
L03: Hypothesis $\gg$                                                  $\ell\,'\,t$                               ;
L04: Hypothesis $\gg$                                                  $x \sim y\,'\,s\,'\,t$                     ;
L05: StrictEquivKX $\unrhd$ L4 $\gg$                                   $\ell\,'\,x$                               ;
L06: Block $\gg$                                                       Begin                                      ;
L07: Hypothesis $\gg$                                                    $\ell\,'\,v$                             ;
L08: ElimAll2 $\unrhd$ L1 $\unrhd$ L2 $\unrhd$ L7 $\gg$                  $\ell\,'(y\,'\,s\,'\,v)$                 ;
L09: TypeEquivK $\unrhd$ L5 $\unrhd$ L8 $\gg$                            $!x \sim y\,'\,s\,'\,v$                  ;
L10: Block $\gg$                                                       End                                        ;
L11: IntroExists $\unrhd$ (Gen $\rhd$ L9) $\unrhd$ L3 $\unrhd$ L4 $\gg$    $\exists t\colon x \sim y\,'\,s\,'\,t$     ;
L12: Block $\gg$                                                       Begin                                      ;
L13: Hypothesis $\gg$                                                    $\ell\,'\,u$                             ;
L14: Block $\gg$                                                         Begin                                    ;
L15: Hypothesis $\gg$                                                      $\ell\,'\,v$                           ;
L16: ElimAll2 $\unrhd$ L1 $\unrhd$ L13 $\unrhd$ L15 $\gg$                  $\ell\,'(y\,'\,u\,'\,v)$               ;
L17: TypeEquivK $\unrhd$ L5 $\unrhd$ L16 $\gg$                             $!x \sim y\,'\,u\,'\,v$                ;
L18: Block $\gg$                                                         End                                      ;
L19: TypeExists $\unrhd$ (Gen $\rhd$ L17) $\gg$                          $!\exists v\colon x \sim y\,'\,u\,'\,v$  ;
L20: Block $\gg$                                                       End                                        ;
L21: IntroExists $\unrhd$ (Gen $\rhd$ L17) $\unrhd$ L2 $\unrhd$ L11 $\gg$  $\exists s\colon \exists t\colon x \sim y\,'\,s\,'\,t$  ;
L22: Repetition $\rhd$ L21 $\gg$                                       $x \in_2 y$                                ]*
```

[The Map proof of IntroInTwo’ reads
```
L1:  Hypothesis $\gg$                                              $\ell_2 y$                          ;
L2:  Hypothesis $\gg$                                              $\ell\,'\,s$                        ;
L3:  Hypothesis $\gg$                                              $\ell\,'\,t$                        ;
L4:  ElimAll2 $\unrhd$ L1 $\unrhd$ L2 $\unrhd$ L3 $\gg$            $\ell\,'(y\,'\,s\,'\,t)$            ;
L5:  RefEquivK $\unrhd$ L4 $\gg$                                   $y\,'\,s\,'\,t \sim y\,'\,s\,'\,t$  ;
L6:  IntroInTwo $\unrhd$ L1 $\unrhd$ L2 $\unrhd$ L3 $\unrhd$ L5 $\gg$   $y\,'\,s\,'\,t \in_2 y$        ]*
```

[The Map proof of IntroInTwoK reads

L1:	Hypothesis $\gg$	$\ell\text{'}x$	;
L2:	TransEquivK $\trianglerighteq$ L1 $\gg$	$x \sim x$	;
L3:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L2 $\gg$	$x \sim \mathsf{K}\text{'}x\text{'}x$	;
L4:	TypeK $\gg$	$\ell_2\mathsf{K}$	;
L5:	IntroInTwo $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$x \in_2 \mathsf{K}$	]*

[The Map proof of IntroInTwoT reads

L1:	TypeT $\gg$	$\ell\text{'}\mathsf{T}$	;
L2:	SubtypeLL2 $\trianglerighteq$ L1 $\gg$	$\ell_2\mathsf{T}$	;
L3:	Reduction $\gg$	$\mathsf{T} \sim \mathsf{T}\text{'}\mathsf{T}\text{'}\mathsf{T}$	;
L4:	IntroInTwo $\trianglerighteq$ L2 $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$\mathsf{T} \in_2 \mathsf{T}$	]*

[The Map proof of ElimEquiv' reads

L01:	Hypothesis $\gg$	$\ell\text{'}x$	;
L02:	Hypothesis $\gg$	$\ell\text{'}y$	;
L03:	Hypothesis $\gg$	$x =_p y$	;
L04:	Hypothesis $\gg$	$z \in_2 p$	;
L05:	Define $\gg$	$s \equiv s_2(z,p)$	;
L06:	Define $\gg$	$t \equiv t_2(z,p)$	;
L07:	ElimInTwo $\trianglerighteq$ L4 $\gg$	$z \sim p\text{'}s\text{'}t$	;
L08:	ElimInTwoS $\trianglerighteq$ L4 $\gg$	$\ell\text{'}s$	;
L09:	ElimInTwoT $\trianglerighteq$ L4 $\gg$	$\ell\text{'}t$	;
L10:	ElimEquiv $\trianglerighteq$ L8 $\trianglerighteq$ L9 $\trianglerighteq$ L3 $\gg$	$x\text{'}(p\text{'}s\text{'}t) =_p y\text{'}(p\text{'}s\text{'}t)$	;
L11:	Block $\gg$	Begin	;
L12:	Hypothesis $\gg$	$\quad\ell\text{'}u$	;
L13:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L12 $\gg$	$\quad\ell\text{'}(x\text{'}u)$	;
L14:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L12 $\gg$	$\quad\ell\text{'}(y\text{'}u)$	;
L15:	StrictInTwoY $\trianglerighteq$ L4 $\gg$	$\quad\ell_2 p$	;
L16:	TypeEquiv $\trianglerighteq$ L15 $\trianglerighteq$ L13 $\trianglerighteq$ L14 $\gg$	$\quad !x\text{'}u =_p y\text{'}u$	;
L17:	Block $\gg$	End	;
L18:	ExtBool" $\trianglerighteq$ (Gen $\triangleright$ L16) $\trianglerighteq$ L7 $\trianglerighteq$ L10 $\gg$	$x\text{'}z =_p y\text{'}z$	]*

[The Map proof of IntroEquiv' reads

L01:	Premise $\gg$	$\ell_2 p$	;
L02:	Premise $\gg$	$\neg x$	;
L03:	Premise $\gg$	$\neg y$	;
L04:	Premise $\gg$	$z \in_2 p \rightarrow x\text{'}z =_p y\text{'}z$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	$\quad\ell\text{'}u$	;
L07:	Hypothesis $\gg$	$\quad\ell\text{'}v$	;
L08:	IntroInTwo' $\trianglerighteq$ L1 $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$\quad p\text{'}u\text{'}v \in_2 p$	;
L09:	L4 $\trianglerighteq$ L8 $\gg$	$\quad x\text{'}(p\text{'}u\text{'}v) =_p y\text{'}(p\text{'}u\text{'}v)$	;
L10:	Block $\gg$	End	;
L11:	IntroEquiv $\triangleright$ L1 $\triangleright$ L2 $\triangleright$ L9 $\gg$	$x =_p y$	]*

[The Map proof of ElimEquivP reads

L01:	Hypothesis $\gg$	$x \sim_p y$	;
L02:	Hypothesis $\gg$	$z \in_2 p$	;
L03:	StrictEquivPP $\trianglerighteq$ L1 $\gg$	$\ell\text{'}p$	;
L04:	StrictEquivPX $\trianglerighteq$ L1 $\gg$	$\ell\text{'}x$	;
L05:	StrictEquivPY $\trianglerighteq$ L1 $\gg$	$\ell\text{'}y$	;
L06:	StrictInTwoX $\trianglerighteq$ L2 $\gg$	$\ell\text{'}z$	;
L07:	Logic $\trianglerighteq$ L1 $\gg$	$x =_p y$	;
L08:	ElimEquiv' $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L7 $\trianglerighteq$ L2 $\gg$	$x\text{'}z =_p y\text{'}z$	;
L09:	TypeApply $\trianglerighteq$ L4 $\trianglerighteq$ L6 $\gg$	$\ell\text{'}(x\text{'}z)$	;
L10:	TypeApply $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$\ell\text{'}(y\text{'}z)$	;
L11:	Logic $\trianglerighteq$ L3 $\trianglerighteq$ L8 $\trianglerighteq$ L9 $\trianglerighteq$ L10 $\gg$	$x\text{'}z \sim_p y\text{'}z$	]*

A.40 Lemmas about $[\, x \subseteq_2 y \,]$

[The Map proof of TypeSub reads

L1:	Hypothesis $\gg$	$\ell_2 x$	;
L2:	Hypothesis $\gg$	$\ell_2 y$	;
L3:	Block $\gg$	Begin	;
L4:	Hypothesis $\gg$	$\ell\text{'}z$	;
L5:	TypeInTwo $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\gg$	$!z \in_2 x$	;
L6:	TypeInTwo $\trianglerighteq$ L4 $\trianglerighteq$ L2 $\gg$	$!z \in_2 y$	;
L7:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$!(z \in_2 x \Rightarrow z \in_2 y)$	;
L8:	Block $\gg$	End	;
L9:	TypeAll $\trianglerighteq$ (Gen $\triangleright$ L7) $\gg$	$!x \subseteq_2 y$	]*

[The Map proof of StrictSubX reads

L1:	Hypothesis $\gg$	$!x \subseteq_2 y$	;
L2:	StrictAll $\trianglerighteq$ L1 $\gg$	$\forall z\colon !(z \in_2 x \Rightarrow z \in_2 y)$	;
L3:	TypeT $\gg$	$\ell\text{'}\mathsf{T}$	;
L4:	ElimAll $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$!(\mathsf{T} \in_2 x \Rightarrow \mathsf{T} \in_2 y)$	;
L5:	Logic $\trianglerighteq$ L4 $\gg$	$!\mathsf{T} \in_2 x$	;
L6:	StrictInTwoY $\trianglerighteq$ L5 $\gg$	$\ell_2 x$	]*

[The Map proof of StrictSubY reads

L1:	Hypothesis $\gg$	$!x \subseteq_2 y$	;
L2:	StrictAll $\trianglerighteq$ L1 $\gg$	$\forall z\colon !(z \in_2 x \Rightarrow z \in_2 y)$	;
L3:	TypeT $\gg$	$\ell\text{'}\mathsf{T}$	;
L4:	ElimAll $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$!(\mathsf{T} \in_2 x \Rightarrow \mathsf{T} \in_2 y)$	;
L5:	Logic $\trianglerighteq$ L4 $\gg$	$!\mathsf{T} \in_2 y$	;
L6:	StrictInTwoY $\trianglerighteq$ L5 $\gg$	$\ell_2 y$	]*

[The Map proof of ElimSub reads

L1:	Hypothesis $\gg$	$p \subseteq_2 q$	;
L2:	Hypothesis $\gg$	$x \in_2 p$	;
L3:	StrictInTwoX $\trianglerighteq$ L2 $\gg$	$\ell\text{'}x$	;
L4:	ElimAll $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$x \in_2 p \Rightarrow x \in_2 q$	;
L5:	Logic $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\gg$	$x \in_2 q$	]*

[The Map proof of IntroSub reads

L01:	Premise $\gg$	$\ell_2 p$	;
L02:	Premise $\gg$	$x \in_2 p \rightarrow x \in_2 q$	;
L03:	TypeT $\gg$	$\ell\,\text{'}\,\mathsf{T}$	;
L04:	IntroInTwo' $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\trianglerighteq$ L3 $\gg$	$p\,\text{'}\,\mathsf{T}\,\text{'}\,\mathsf{T} \in_2 p$	;
L05:	L2 $\trianglerighteq$ L4 $\gg$	$p\,\text{'}\,\mathsf{T}\,\text{'}\,\mathsf{T} \in_2 q$	;
L06:	StrictInTwoY $\trianglerighteq$ L5 $\gg$	$\ell_2 q$	;
L07:	Block $\gg$	Begin	;
L08:	Hypothesis $\gg$	$\quad \ell\,\text{'}\,x$	;
L09:	TypeInTwo $\trianglerighteq$ L1 $\trianglerighteq$ L8 $\gg$	$\quad !x \in_2 p$	;
L10:	TypeInTwo $\trianglerighteq$ L6 $\trianglerighteq$ L8 $\gg$	$\quad !x \in_2 q$	;
L11:	CDeduction $\triangleright$ L9 $\triangleright$ L10 $\triangleright$ L2 $\gg$	$\quad x \in_2 p \Rightarrow x \in_2 q$	;
L12:	Block $\gg$	End	;
L13:	Gen $\triangleright$ L11 $\gg$	$p \subseteq_2 q$	]*

[The Map proof of TransSub reads

L1:	Hypothesis $\gg$	$p \subseteq_2 q$	;
L2:	Hypothesis $\gg$	$q \subseteq_2 r$	;
L3:	StrictSubX $\trianglerighteq$ L1 $\gg$	$\ell_2 p$	;
L4:	Block $\gg$	Begin	;
L5:	Hypothesis $\gg$	$\quad x \in_2 p$	;
L6:	ElimSub $\trianglerighteq$ L1 $\trianglerighteq$ L5 $\gg$	$\quad x \in_2 q$	;
L7:	ElimSub $\trianglerighteq$ L2 $\trianglerighteq$ L6 $\gg$	$\quad x \in_2 r$	;
L8:	Block $\gg$	End	;
L9:	IntroSub $\triangleright$ L3 $\triangleright$ L9 $\gg$	$p \subseteq_2 r$	]*

[The Map proof of IntroSubK reads

L1:	Hypothesis $\gg$	$\ell_2 p$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad x \in_2 p$	;
L4:	StrictInTwoX $\trianglerighteq$ L3 $\gg$	$\quad \ell\,\text{'}\,x$	;
L5:	IntroInTwoK $\trianglerighteq$ L4 $\gg$	$\quad x \in_2 \mathsf{K}$	;
L6:	Block $\gg$	End	;
L7:	IntroSub $\trianglerighteq$ L1 $\trianglerighteq$ L5 $\gg$	$p \subseteq_2 \mathsf{K}$	]*

[The Map proof of IntroKSub reads

L01:	Hypothesis $\gg$	$\ell\,{}'\,x$	;
L02:	Hypothesis $\gg$	$\ell\,{}'\,y$	;
L03:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\,{}'(x\,{}'\,y)$	;
L04:	TypeK $\gg$	$\ell_2\mathsf{K}$	;
L05:	ElimAll $\trianglerighteq$ L4 $\trianglerighteq$ L3 $\gg$	$\ell_1\mathsf{K}\,{}'(x\,{}'\,y)$	;
L06:	SubtypeL1L2 $\trianglerighteq$ L5 $\gg$	$\ell_2\mathsf{K}\,{}'(x\,{}'\,y)$	;
L07:	Block $\gg$	Begin	;
L08:	Hypothesis $\gg$	$\quad z \in_2 \mathsf{K}\,{}'(x\,{}'\,y)$	;
L09:	Define $\gg$	$\quad s \equiv s_2(z, \mathsf{K}\,{}'(x\,{}'\,y))$	;
L10:	Define $\gg$	$\quad t \equiv t_2(z, \mathsf{K}\,{}'(x\,{}'\,y))$	;
L11:	ElimInTwo $\trianglerighteq$ L8 $\gg$	$\quad z \sim \mathsf{K}\,{}'(x\,{}'\,y)\,{}'\,s\,{}'\,t$	;
L12:	ElimInTwoT $\trianglerighteq$ L8 $\gg$	$\quad \ell\,{}'\,t$	;
L13:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L11 $\gg$	$\quad z \sim x\,{}'\,y\,{}'\,t$	;
L14:	SubtypeLL2 $\trianglerighteq$ L1 $\gg$	$\quad \ell_2 x$	;
L15:	IntroInTwo $\triangleright$ L14 $\triangleright$ L2 $\triangleright$ L12 $\triangleright$ L13 $\gg$	$\quad z \in_2 x$	;
L16:	Block $\gg$	End	;
L17:	IntroSub $\triangleright$ L4 $\triangleright$ L15 $\gg$	$\mathsf{K}\,{}'(x\,{}'\,y) \subseteq_2 x$	]*

[Map lemma
SubEquiv'$_{171}$: $p \subseteq_2 q \to \neg u \to \ell\,{}'\,u \to$
$\forall w\colon \forall v\colon (u\,{}'\,w =_q v \Rightarrow u\,{}'\,w =_p v) \to$
$\forall v\colon (u =_q v \Rightarrow u =_p v)$]*

[The Map proof of SubEquiv' reads

L01:	Hypothesis $\gg$	$p \subseteq_2 q$	;
L02:	Hypothesis $\gg$	$\neg u$	;
L03:	Hypothesis $\gg$	$\ell\,'\,u$	;
L04:	Hypothesis $\gg$	$\forall w\colon \forall v\colon (u\,'\,w =_q v \Rightarrow u\,'\,w =_p v)$	;
L05:	StrictSubX $\trianglerighteq$ L1 $\gg$	$\ell_2 p$	;
L06:	StrictSubY $\trianglerighteq$ L1 $\gg$	$\ell_2 q$	;
L07:	Block $\gg$	Begin	;
L08:	Hypothesis $\gg$	$\ell\,'\,v$	;
L09:	TypeEquiv $\trianglerighteq$ L6 $\trianglerighteq$ L3 $\trianglerighteq$ L8 $\gg$	$!u =_q v$	;
L10:	TypeEquiv $\trianglerighteq$ L5 $\trianglerighteq$ L3 $\trianglerighteq$ L8 $\gg$	$!u =_p v$	;
L11:	Block $\gg$	Begin	;
L12:	Hypothesis $\gg$	$u =_q v$	;
L13:	Logic $\trianglerighteq$ L2 $\trianglerighteq$ L12 $\gg$	$\neg v$	;
L14:	Block $\gg$	Begin	;
L15:	Hypothesis $\gg$	$z \in_2 p$	;
L16:	ElimSub $\trianglerighteq$ L1 $\trianglerighteq$ L15 $\gg$	$z \in_2 q$	;
L17:	ElimEquiv' $\trianglerighteq$ L3 $\trianglerighteq$ L8 $\trianglerighteq$ L12 $\trianglerighteq$ L16 $\gg$	$u\,'\,z =_q v\,'\,z$	;
L18:	StrictInTwoX $\trianglerighteq$ L15 $\gg$	$\ell\,'\,z$	;
L19:	TypeApply $\trianglerighteq$ L8 $\trianglerighteq$ L18 $\gg$	$\ell\,'(v\,'\,z)$	;
L20:	ElimAll2 $\trianglerighteq$ L4 $\trianglerighteq$ L18 $\trianglerighteq$ L19 $\gg$	$u\,'\,z =_q v\,'\,z \Rightarrow u\,'\,z =_p v\,'\,z$	;
L21:	Logic $\trianglerighteq$ L17 $\trianglerighteq$ L20 $\gg$	$u\,'\,z =_p v\,'\,z$	;
L22:	Block $\gg$	End	;
L23:	IntroEquiv' $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\trianglerighteq$ L13 $\trianglerighteq$ L21 $\gg$	$u =_p v$	;
L24:	Block $\gg$	End	;
L25:	CDeduction $\triangleright$ L9 $\triangleright$ L10 $\triangleright$ L23 $\gg$	$u =_q v \Rightarrow u =_p v$	;
L26:	Block $\gg$	End	;
L27:	Gen $\triangleright$ L25 $\gg$	$\forall v\colon (u =_q v \Rightarrow u =_p v)$	]*

[The Map proof of SubEquiv reads

L01:	Hypothesis $\gg$	ℓ'x	;
L02:	Hypothesis $\gg$	ℓ'y	;
L03:	Hypothesis $\gg$	$p \subseteq_2 q$	;
L04:	Hypothesis $\gg$	$x =_q y$	;
L05:	StrictSubX $\trianglerighteq$ L3 $\gg$	$\ell_2 p$	;
L06:	StrictSubY $\trianglerighteq$ L3 $\gg$	$\ell_2 q$	;
L07:	Block $\gg$	Begin	;
L08:	Hypothesis $\gg$	ℓ'v	;
L09:	TypeT $\gg$	ℓ'T	;
L10:	TypeEquiv $\trianglerighteq$ L6 $\trianglerighteq$ L9 $\trianglerighteq$ L8 $\gg$	$!\mathsf{T} =_q v$	;
L11:	TypeEquiv $\trianglerighteq$ L5 $\trianglerighteq$ L9 $\trianglerighteq$ L8 $\gg$	$!\mathsf{T} =_p v$	;
L12:	Logic $\gg$	$\mathsf{T} =_q v \rightarrow \mathsf{T} =_p v$	;
L13:	CDeduction $\triangleright$ L10 $\triangleright$ L11 $\triangleright$ L12 $\gg$	$\mathsf{T} =_q v \Rightarrow \mathsf{T} =_p v$	;
L14:	Block $\gg$	End	;
L15:	Gen $\triangleright$ L13 $\gg$	$\forall v\colon (\mathsf{T} =_q v \Rightarrow \mathsf{T} =_p v)$	;
L16:	Transfinite'$\triangleright$L15$\triangleright$(SubEquiv'$\trianglerighteq$L3)$\gg$	$\forall u\colon \forall v\colon u =_p v$	;
L17:	ElimAll2 $\trianglerighteq$ L16 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$(x =_q y \Rightarrow x =_p y)$	;
L18:	Logic $\trianglerighteq$ L17 $\trianglerighteq$ L4 $\gg$	$x =_p y$	]*

[The Map proof of SubEquivP reads

L1:	Hypothesis $\gg$	ℓ'p	;
L2:	Hypothesis $\gg$	$p \subseteq_2 q$	;
L3:	Hypothesis $\gg$	$x \sim_q y$	;
L4:	Logic $\trianglerighteq$ L3 $\gg$	$!x \sim_q y$	;
L5:	StrictEquivPX $\trianglerighteq$ L4 $\gg$	ℓ'x	;
L6:	StrictEquivPY $\trianglerighteq$ L4 $\gg$	ℓ'y	;
L7:	Logic $\trianglerighteq$ L3 $\gg$	$x =_q y$	;
L8:	SubEquiv $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\trianglerighteq$ L2 $\trianglerighteq$ L7 $\gg$	$x =_p y$	;
L9:	Logic $\trianglerighteq$ L1 $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\trianglerighteq$ L8 $\gg$	$x \sim_p y$	]*

[The Map proof of SubEquivK reads

L1:	Hypothesis $\gg$	ℓ'p	;
L2:	Hypothesis $\gg$	$x \sim y$	;
L3:	Logic $\trianglerighteq$ L2 $\gg$	ℓ'x	;
L4:	Logic $\trianglerighteq$ L2 $\gg$	ℓ'y	;
L5:	Logic $\trianglerighteq$ L2 $\gg$	$x =_{\mathsf{K}} y$	;
L6:	SubtypeLL2 $\trianglerighteq$ L1 $\gg$	$\ell_2 p$	;
L7:	IntroSubK $\trianglerighteq$ L6 $\gg$	$p \subseteq_2 \mathsf{K}$	;
L8:	SubEquiv $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\trianglerighteq$ L7 $\trianglerighteq$ L5 $\gg$	$x =_p y$	;
L9:	Logic $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\trianglerighteq$ L8 $\gg$	$x \sim_p y$	]*

[The Map proof of SubInEll reads

L1:	Hypothesis $\gg$	$\ell\,'\,q$	;
L2:	Hypothesis $\gg$	$p \subseteq_2 q$	;
L3:	Hypothesis $\gg$	$x \in_\ell p$	;
L4:	Block $\gg$	Begin	;
L5:	Hypothesis $\gg$	$u \sim_q v$	;
L6:	SubEquivP $\trianglerighteq$ L2 $\trianglerighteq$ L5 $\gg$	$u \sim_p v$	;
L7:	ElimInEll $\trianglerighteq$ L3 $\trianglerighteq$ L6 $\gg$	$x\,'\,u \sim x\,'\,v$	;
L8:	Block $\gg$	End	;
L9:	IntroInEll $\triangleright$ L1 $\triangleright$ L7 $\gg$	$x \in_\ell q$	]*

A.41 Extensionality lemmas

[The Map proof of ExtT reads

L1:	RefEquivK $\trianglerighteq$ TypeT $\gg$	$\mathsf{T} \sim \mathsf{T}$	]*

[The Map proof of ExtKApplyP reads

L1:	Hypothesis $\gg$	$a \in_\ell p$	;
L2:	Hypothesis $\gg$	$a \sim \underline{a}$	;
L3:	Hypothesis $\gg$	$x \sim_p \underline{x}$	;
L4:	ElimInEll $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$a\,'\,x \sim a\,'\,\underline{x}$	;
L5:	StrictEquivPY $\trianglerighteq$ L3 $\gg$	$\ell\,'\,\underline{x}$	;
L6:	ElimEquivK $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\gg$	$a\,'\,\underline{x} \sim \underline{a}\,'\,\underline{x}$	;
L7:	TransEquivK $\trianglerighteq$ L4 $\trianglerighteq$ L6 $\gg$	$a\,'\,x \sim \underline{a}\,'\,\underline{x}$	]*

[The Map proof of ExtPApplyK reads

L01:	Hypothesis $\gg$	$a \in_2 p$	;
L02:	Hypothesis $\gg$	$x \sim_p \underline{x}$	;
L03:	Hypothesis $\gg$	$a \sim \underline{a}$	;
L04:	ElimEquivP $\trianglerighteq$ L2 $\trianglerighteq$ L1 $\gg$	$x\,'\,a \sim_p \underline{x}\,'\,a$	;
L05:	StrictEquivPY $\trianglerighteq$ L2 $\gg$	$\ell\,'\,\underline{x}$	;
L06:	SubtypeLL1 $\trianglerighteq$ L5 $\gg$	$\ell_1\underline{x}$	;
L07:	Ext $\trianglerighteq$ L6 $\trianglerighteq$ L3 $\gg$	$\underline{x}\,'\,a \sim \underline{x}\,'\,\underline{a}$	;
L08:	StrictEquivPP $\trianglerighteq$ L2 $\gg$	$\ell\,'\,p$	;
L09:	SubEquivK $\trianglerighteq$ L8 $\trianglerighteq$ L7 $\gg$	$\underline{x}\,'\,a \sim_p \underline{x}\,'\,\underline{a}$	;
L10:	TransEquivP $\trianglerighteq$ L4 $\trianglerighteq$ L9 $\gg$	$x\,'\,a \sim_p \underline{x}\,'\,\underline{a}$	]*

[The Map proof of ExtKApplyK reads

L1:	Hypothesis $\gg$	$a \sim \underline{a}$	;
L2:	Hypothesis $\gg$	$b \sim \underline{b}$	;
L3:	StrictEquivKX $\trianglerighteq$ L1 $\gg$	$\ell\,'\,a$	;
L4:	ElimEll $\trianglerighteq$ L3 $\gg$	$a \in_\ell \mathcal{I}a$	;
L5:	StrictInEllP $\trianglerighteq$ L4 $\gg$	$\ell\,'\,\mathcal{I}a$	;
L6:	SubEquivK $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\gg$	$b \sim_{\mathcal{I}a} \underline{b}$	;
L7:	ExtKApplyP $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\trianglerighteq$ L6 $\gg$	$a\,'\,b \sim \underline{a}\,'\,\underline{b}$	]*

[The Map proof of ExtIfBKK reads

L01:	Hypothesis $\gg$	$x \Leftrightarrow \underline{x}$	;
L02:	Hypothesis $\gg$	$b \sim \underline{b}$	;
L03:	Hypothesis $\gg$	$c \sim \underline{c}$	;
L04:	Logic $\trianglerighteq$ L1 $\gg$	$!x$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	x	;
L07:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L1 $\gg$	$\underline{x}$	;
L08:	Logic $\trianglerighteq$ L6 $\gg$	$\mathrm{if}(x, b, c) \equiv b$	;
L09:	Logic $\trianglerighteq$ L7 $\gg$	$\mathrm{if}(\underline{x}, \underline{b}, \underline{c}) \equiv \underline{b}$	;
L10:	Replace' $\trianglerighteq$ L8 $\trianglerighteq$ L2 $\gg$	$\mathrm{if}(x, b, c) \sim \underline{b}$	;
L11:	Replace' $\trianglerighteq$ L9 $\trianglerighteq$ L10 $\gg$	$\mathrm{if}(x, b, c) \sim \mathrm{if}(\underline{x}, \underline{b}, \underline{c})$	;
L12:	Block $\gg$	End	;
L13:	Block $\gg$	Begin	;
L14:	Hypothesis $\gg$	$\neg x$	;
L15:	Logic $\trianglerighteq$ L14 $\trianglerighteq$ L1 $\gg$	$\neg \underline{x}$	;
L16:	Logic $\trianglerighteq$ L14 $\gg$	$\mathrm{if}(x, b, c) \equiv c$	;
L17:	Logic $\trianglerighteq$ L15 $\gg$	$\mathrm{if}(\underline{x}, \underline{b}, \underline{c}) \equiv \underline{c}$	;
L18:	Replace' $\trianglerighteq$ L16 $\trianglerighteq$ L3 $\gg$	$\mathrm{if}(x, b, c) \sim \underline{c}$	;
L19:	Replace' $\trianglerighteq$ L17 $\trianglerighteq$ L18 $\gg$	$\mathrm{if}(x, b, c) \sim \mathrm{if}(\underline{x}, \underline{b}, \underline{c})$	;
L20:	Block $\gg$	End	;
L21:	TND $\triangleright$ L4 $\triangleright$ L11 $\triangleright$ L19 $\gg$	$\mathrm{if}(x, b, c) \sim \mathrm{if}(\underline{x}, \underline{b}, \underline{c})$	]*

[The Map proof of ExtIfPKK reads

L1:	Hypothesis $\gg$	$x \sim_p \underline{x}$	;
L2:	Hypothesis $\gg$	$b \sim \underline{b}$	;
L3:	Hypothesis $\gg$	$c \sim \underline{c}$	;
L4:	Logic $\trianglerighteq$ L1 $\gg$	$x \Leftrightarrow \underline{x}$	;
L5:	ExtIfBKK $\trianglerighteq$ L4 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$\mathrm{if}(x, b, c) \sim \mathrm{if}(\underline{x}, \underline{b}, \underline{c})$	]*

[The Map proof of ExtIfKKK reads

L1:	Hypothesis $\gg$	$a \sim \underline{a}$	;
L2:	Hypothesis $\gg$	$b \sim \underline{b}$	;
L3:	Hypothesis $\gg$	$c \sim \underline{c}$	;
L4:	SubEquivK $\trianglerighteq$ TypeT $\gg$	$a \sim_{\mathsf{T}} \underline{a}$	;
L5:	ExtIfPKK $\trianglerighteq$ L4 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$\mathrm{if}(a, b, c) \sim \mathrm{if}(\underline{a}, \underline{b}, \underline{c})$	]*

[The Map proof of ExtLambda reads

L01:	Premise $\gg$	$\ell\text{'}p$	;
L02:	Premise $\gg$	$x \sim_p \underline{x} \to a \sim \underline{a}$	;
L03:	Define $\gg$	$h \equiv \lambda x.a$	;
L04:	Define $\gg$	$\underline{h} \equiv \lambda \underline{x}.\underline{a}$	;
L05:	Reduction $\gg$	$h\text{'}x \equiv a$	;
L06:	Reduction $\gg$	$\underline{h}\text{'}\underline{x} \equiv \underline{a}$	;
L07:	Replace' $\rhd$ L5 $\rhd$ L2 $\gg$	$x \sim_p \underline{x} \to h\text{'}x \sim \underline{a}$	;
L08:	Replace' $\rhd$ L6 $\rhd$ L7 $\gg$	$x \sim_p \underline{x} \to h\text{'}x \sim \underline{h}\text{'}\underline{x}$	;
L09:	Reduction $\gg$	$\neg h$	;
L10:	Reduction $\gg$	$\neg \underline{h}$	;
L11:	Block $\gg$	Begin	;
L12:	Hypothesis $\gg$	$x \sim_p \underline{x}$	;
L13:	L8 $\trianglerighteq$ L12 $\gg$	$h\text{'}x \sim \underline{h}\text{'}\underline{x}$	;
L14:	StrictEquivPY $\trianglerighteq$ L12 $\gg$	$\ell\text{'}\underline{x}$	;
L15:	RefEquivP $\trianglerighteq$ L1 $\trianglerighteq$ L14 $\gg$	$\underline{x} \sim_p \underline{x}$	;
L16:	L8 $\trianglerighteq$ L15 $\gg$	$h\text{'}\underline{x} \sim \underline{h}\text{'}\underline{x}$	;
L17:	ComEquivP $\trianglerighteq$ L16 $\gg$	$\underline{h}\text{'}\underline{x} \sim h\text{'}\underline{x}$	;
L18:	TransEquivP $\trianglerighteq$ L13 $\trianglerighteq$ L17 $\gg$	$h\text{'}x \sim h\text{'}\underline{x}$	;
L19:	Block $\gg$	End	;
L20:	IntroInEll $\rhd$ L1 $\rhd$ L18 $\gg$	$h \in_\ell p$	;
L21:	IntroEll $\trianglerighteq$ L20 $\gg$	$\ell\text{'}h$	;
L22:	Block $\gg$	Begin	;
L23:	Hypothesis $\gg$	$\ell\text{'}x$	;
L24:	RefEquivP $\trianglerighteq$ L1 $\trianglerighteq$ L23 $\gg$	$x \sim_p x$	;
L25:	L8 $\trianglerighteq$ L24 $\gg$	$h\text{'}x \sim \underline{h}\text{'}x$	;
L26:	Block $\gg$	End	;
L27:	Gen $\rhd$ L25 $\gg$	$\forall x\colon h\text{'}x \sim \underline{h}\text{'}x$	;
L28:	IntroEquivK $\trianglerighteq$ L9 $\trianglerighteq$ L10 $\trianglerighteq$ L21 $\trianglerighteq$ L27 $\gg$	$h \sim \underline{h}$	;
L29:	Repetition $\rhd$ L28 $\gg$	$\lambda x.a \sim \lambda \underline{x}.\underline{a}$	]*

A.42 Elementary type lemmas

[The Map proof of ExtF reads

L1:	TypeT $\gg$	$\ell\text{'}\mathsf{T}$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$x \sim_{\mathsf{T}} \underline{x}$	;
L4:	ExtT $\gg$	$\mathsf{T} \sim \mathsf{T}$	;
L5:	Block $\gg$	End	;
L6:	ExtLambda $\rhd$ L1 $\rhd$ L4 $\gg$	$\lambda x.\mathsf{T} \sim \lambda \underline{x}.\mathsf{T}$	;
L7:	Rename $\rhd$ L6 $\gg$	$\mathsf{F} \sim \mathsf{F}$	]*

[The Map proof of TypeF reads

L1:	ExtF $\gg$	$\mathsf{F} \sim \mathsf{F}$	;
L2:	StrictEquivKX $\trianglerighteq$ L1 $\gg$	$\ell\text{'}\mathsf{F}$	]*

[The Map proof of TypeKa reads

L1:	Hypothesis $\gg$	$\ell\,'\,a$	;
L2:	TypeT $\gg$	$\ell\,'\,\mathsf{T}$	;
L3:	Block $\gg$	Begin	;
L4:	Hypothesis $\gg$	$\quad x \sim_{\mathsf{T}} \underline{x}$	;
L5:	RefEquivK $\trianglerighteq$ L1 $\gg$	$\quad a \sim a$	;
L6:	Block $\gg$	End	;
L7:	ExtLambda $\rhd$ L2 $\rhd$ L5 $\gg$	$\lambda x.a \sim \lambda \underline{x}.a$	;
L8:	StrictEquivKX $\trianglerighteq$ L7 $\gg$	$\ell\,'\,\lambda x.a$	;
L9:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L8 $\gg$	$\ell\,'(\mathsf{K}\,'\,a)$	]*

A.43 First union operator

To develop the theory of classicality further, it is necessary to prove a lemma that resembles the union set axiom of ZFC. To do so, a sequence of four union operators [u_1], [u_2], [u_3], and [u_4] of increasing power have to be introduced. After that, it is necessary to prove a lemma that resembles the existence of transitive closures in ZFC. That requires two more union operators [u_5] and [u_6]. The first union operator essentially allows to form the set [$a \cup \{\mathsf{T}, \mathsf{F}\}$] given a set [a].

$$\left[\ \boxed{u_1(a)} \doteq \lambda x.\mathrm{if}(x\,'\,\mathsf{T}, \mathsf{T} :: \mathsf{F}, a\,'(x\,'\,\mathsf{F}))\ \right]^*$$

[Map lemma

TypeU1L2$_{178}$:	$\ell a{:}\, \ell_2 u_1(a)$	;
IntroU1T$_{178}$:	$\ell a{:}\, \mathsf{T} \in_2 u_1(a)$	;
IntroU1F$_{178}$:	$\ell a{:}\, \mathsf{F} \in_2 u_1(a)$	;
IntroU1a$_{179}$:	$\ell a{:}\, a \subseteq_2 u_1(a)$	;
IntroU1U1$_{179}$:	$\ell\,'\,u_1(b) \rightarrow a \in_\ell b \rightarrow u_1(a) \in_\ell u_1(b)$	;
TypeU1$_{180}$:	$\ell a{:}\, \ell\,'\,u_1(a)$	;
ElimU1$_{181}$:	$\ell x, a{:}\, x \in_2 u_1(a) \rightarrow x \sim \mathsf{T} \vee x \sim \mathsf{F} \vee x \in_2 a$	]*

[The Map proof of TypeU1L2 reads

L01:	Hypothesis $\gg$	$\ell \text{'} a$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\ell \text{'} x$	;
L04:	TypeT $\gg$	$\ell \text{'} \mathsf{T}$	;
L05:	TypeF $\gg$	$\ell \text{'} \mathsf{F}$	;
L06:	TypeApply $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$\ell \text{'}(x \text{'} \mathsf{T})$	;
L07:	SubtypeLB $\trianglerighteq$ L6 $\gg$	$!x \text{'} \mathsf{T}$	;
L08:	Block $\gg$	Begin	;
L09:	Hypothesis $\gg$	$x \text{'} \mathsf{T}$	;
L10:	Logic $\trianglerighteq$ L9 $\gg$	$u_1(a) \text{'} x \equiv \mathsf{T} :: \mathsf{F}$	;
L11:	TypePair $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\gg$	$\ell \text{'}(\mathsf{T} :: \mathsf{F})$	;
L12:	Replace' $\rhd$ L10 $\rhd$ L11 $\gg$	$\ell \text{'}(u_1(a) \text{'} x)$	;
L13:	Block $\gg$	End	;
L14:	Block $\gg$	Begin	;
L15:	Hypothesis $\gg$	$\neg(x \text{'} \mathsf{T})$	;
L16:	Logic $\trianglerighteq$ L15 $\gg$	$u_1(a) \text{'} x \equiv a \text{'}(x \text{'} \mathsf{F})$	;
L17:	TypeApply $\trianglerighteq$ L3 $\trianglerighteq$ L5 $\gg$	$\ell \text{'}(x \text{'} \mathsf{F})$	;
L18:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L17 $\gg$	$\ell \text{'}(a \text{'}(x \text{'} \mathsf{F}))$	;
L19:	Replace' $\rhd$ L16 $\rhd$ L18 $\gg$	$\ell \text{'}(u_1(a) \text{'} x)$	;
L20:	Block $\gg$	End	;
L21:	TND $\rhd$ L7 $\rhd$ L12 $\rhd$ L19 $\gg$	$\ell \text{'}(u_1(a) \text{'} x)$	;
L22:	Block $\gg$	End	;
L23:	Gen $\rhd$ L21 $\gg$	$\ell_1 u_1(a)$	;
L24:	SubtypeL1L2 $\trianglerighteq$ L23 $\gg$	$\ell_2 u_1(a)$	]*

[The Map proof of IntroU1T reads

L1:	Hypothesis $\gg$	$\ell \text{'} a$	;
L2:	TypeU1L2 $\trianglerighteq$ L1 $\gg$	$\ell_2 u_1(a)$	;
L3:	TypeT $\gg$	$\ell \text{'} \mathsf{T}$	;
L4:	IntroInTwo' $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\trianglerighteq$ L3 $\gg$	$u_1(a) \text{'} \mathsf{T} \text{'} \mathsf{T} \in_2 u_1(a)$	;
L5:	Reduction $\gg$	$u_1(a) \text{'} \mathsf{T} \text{'} \mathsf{T} \equiv \mathsf{T}$	;
L6:	Replace' $\rhd$ L5 $\rhd$ L4 $\gg$	$\mathsf{T} \in_2 u_1(a)$	]*

[The Map proof of IntroU1F reads

L1:	Hypothesis $\gg$	$\ell \text{'} a$	;
L2:	TypeU1L2 $\trianglerighteq$ L1 $\gg$	$\ell_2 u_1(a)$	;
L3:	TypeT $\gg$	$\ell \text{'} \mathsf{T}$	;
L4:	TypeF $\gg$	$\ell \text{'} \mathsf{F}$	;
L5:	IntroInTwo' $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$u_1(a) \text{'} \mathsf{T} \text{'} \mathsf{F} \in_2 u_1(a)$	;
L6:	Reduction $\gg$	$u_1(a) \text{'} \mathsf{T} \text{'} \mathsf{F} \equiv \mathsf{F}$	;
L7:	Replace' $\rhd$ L6 $\rhd$ L5 $\gg$	$\mathsf{F} \in_2 u_1(a)$	]*

[The Map proof of IntroU1a reads

L01:	Hypothesis $\gg$	$\ell\text{'}a$	;
L02:	SubtypeLL2 $\trianglerighteq$ L1 $\gg$	$\ell_2 a$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$x \in_2 a$	;
L05:	Define $\gg$	$s \equiv s_2(x,a)$	;
L06:	Define $\gg$	$t \equiv t_2(x,a)$	;
L07:	ElimInTwo $\trianglerighteq$ L4 $\gg$	$x \sim a\text{'}s\text{'}t$	;
L08:	ElimInTwoS $\trianglerighteq$ L4 $\gg$	$\ell\text{'}s$	;
L09:	ElimInTwoT $\trianglerighteq$ L4 $\gg$	$\ell\text{'}t$	;
L10:	TypeF $\gg$	$\ell\text{'}\mathsf{F}$	;
L11:	TypePair $\trianglerighteq$ L10 $\trianglerighteq$ L8 $\gg$	$\ell\text{'}(\mathsf{F} :: s)$	;
L12:	Reduction $\gg$	$u_1(a)\text{'}(\mathsf{F} :: s)\text{'}t \equiv a\text{'}s\text{'}t$	;
L13:	Replace' $\triangleright$ L12 $\triangleright$ L7 $\gg$	$x \sim u_1(a)\text{'}(\mathsf{F} :: s)\text{'}t$	;
L14:	TypeU1L2 $\trianglerighteq$ L1 $\gg$	$\ell_2 u_1(a)$	;
L15:	IntroInTwo$\trianglerighteq$L14$\trianglerighteq$L11$\trianglerighteq$L9$\trianglerighteq$L13$\gg$	$x \in_2 u_1(a)$	;
L16:	Block $\gg$	End	;
L17:	IntroSub $\triangleright$ L2 $\triangleright$ L15 $\gg$	$a \subseteq_2 u_1(a)$	]*

[The Map proof of IntroU1U1 reads

L01:	Hypothesis $\gg$	$\ell\text{'}u_1(b)$	;
L02:	Hypothesis $\gg$	$a \in_\ell b$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$x \sim_{u_1(b)} \underline{x}$	;
L05:	StrictInEllP $\trianglerighteq$ L2 $\gg$	$\ell\text{'}b$	;
L06:	IntroU1T $\trianglerighteq$ L5 $\gg$	$\mathsf{T} \in_2 u_1(b)$	;
L07:	IntroU1F $\trianglerighteq$ L5 $\gg$	$\mathsf{F} \in_2 u_1(b)$	;
L08:	IntroU1a $\trianglerighteq$ L5 $\gg$	$b \subseteq_2 u_1(b)$	;
L09:	ElimEquivP $\trianglerighteq$ L4 $\trianglerighteq$ L6 $\gg$	$x\text{'}\mathsf{T} \sim_{u_1(b)} \underline{x}\text{'}\mathsf{T}$	;
L10:	ExtT $\gg$	$\mathsf{T} \sim \mathsf{T}$	;
L11:	ExtF $\gg$	$\mathsf{F} \sim \mathsf{F}$	;
L12:	ExtKApplyK $\trianglerighteq$ L10 $\trianglerighteq$ L11 $\gg$	$\mathsf{T} :: \mathsf{F} \sim \mathsf{T} :: \mathsf{F}$	;
L13:	ElimEquivP $\trianglerighteq$ L4 $\trianglerighteq$ L7 $\gg$	$x\text{'}\mathsf{F} \sim_{u_1(b)} \underline{x}\text{'}\mathsf{F}$	;
L14:	SubEquivP $\trianglerighteq$ L1 $\trianglerighteq$ L8 $\trianglerighteq$ L4 $\gg$	$x\text{'}\mathsf{F} \sim_b \underline{x}\text{'}\mathsf{F}$	;
L15:	ElimInEll $\trianglerighteq$ L2 $\trianglerighteq$ L14 $\gg$	$a\text{'}(x\text{'}\mathsf{F}) \sim a\text{'}(\underline{x}\text{'}\mathsf{F})$	;
L16:	ExtIfPKK $\trianglerighteq$ L9 $\trianglerighteq$ L12 $\trianglerighteq$ L15 $\gg$	$\mathrm{if}(x, \mathsf{T} :: \mathsf{F}, a\text{'}(x\text{'}\mathsf{F})) \sim$	
		$\mathrm{if}(\underline{x}, \mathsf{T} :: \mathsf{F}, a\text{'}(\underline{x}\text{'}\mathsf{F}))$	;
L17:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L16 $\gg$	$u_1(a)\text{'}x \sim u_1(a)\text{'}\underline{x}$	;
L18:	Block $\gg$	End	;
L19:	IntroInEll $\triangleright$ L1 $\triangleright$ L17 $\gg$	$u_1(a) \in_\ell u_1(b)$	]*

[The Map proof of TypeU1 reads

L01:	TypeT $\gg$	$\ell\,'\,\mathsf{T}$	;
L02:	TypeF $\gg$	$\ell\,'\,\mathsf{F}$	;
L03:	TypePair $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\,'(\mathsf{T} :: \mathsf{F})$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\quad x \sim_{\mathsf{T}} \underline{x}$	;
L06:	IntroInTwoT $\gg$	$\quad \mathsf{T} \in_2 \mathsf{T}$	;
L07:	ExtT $\gg$	$\quad \mathsf{T} \sim \mathsf{T}$	;
L08:	ExtPApplyK$\trianglerighteq$L6$\trianglerighteq$L5$\trianglerighteq$L7$\gg$	$\quad x\,'\,\mathsf{T} \sim_{\mathsf{T}} \underline{x}\,'\,\mathsf{T}$	;
L09:	RefEquivK $\trianglerighteq$ L3 $\gg$	$\quad \mathsf{T} :: \mathsf{F} \sim \mathsf{T} :: \mathsf{F}$	;
L10:	ExtIfPKK$\trianglerighteq$L8$\trianglerighteq$L9$\trianglerighteq$L7$\gg$	$\quad \mathrm{if}(x'\mathsf{T}, \mathsf{T}::\mathsf{F}, \mathsf{T}) \sim \mathrm{if}(\underline{x}'\mathsf{T}, \mathsf{T}::\mathsf{F}, \mathsf{T})$	;
L11:	Block $\gg$	End	;
L12:	ExtLambda $\triangleright$ L1 $\triangleright$ L10 $\gg$	$\ell\,'\,\lambda x.\mathrm{if}(x\,'\,\mathsf{T}, \mathsf{T} :: \mathsf{F}, \mathsf{T})$	;
L13:	Reduction $\gg$	$u_1(\mathsf{T}) \equiv \lambda x.\mathrm{if}(x\,'\,\mathsf{T}, \mathsf{T} :: \mathsf{F}, \mathsf{T})$	;
L14:	Replace $\triangleright$ L13 $\triangleright$ L12 $\gg$	$\ell\,'\,u_1(\mathsf{T})$	;
L15:	Block $\gg$	Begin	;
L16:	Hypothesis $\gg$	$\quad \neg a$	;
L17:	Hypothesis $\gg$	$\quad \ell\,'\,a$	;
L18:	Hypothesis $\gg$	$\quad \forall b\colon \ell\,'\,u_1(a\,'\,b)$	;
L19:	Hypothesis $\gg$	$\quad \mathsf{E} b\colon \ell\,'\,b \wedge \ell\,'\,u_1(b) \wedge a \in_l b$	;
L20:	Block $\gg$	$\quad$ Begin	;
L21:	Hypothesis $\gg$	$\qquad \ell\,'\,b \wedge \ell\,'\,u_1(b) \wedge a \in_l b$	;
L22:	Logic $\trianglerighteq$ L21 $\gg$	$\qquad \ell\,'\,b$	;
L23:	Logic $\trianglerighteq$ L21 $\gg$	$\qquad \ell\,'\,u_1(b)$	;
L24:	Logic $\trianglerighteq$ L21 $\gg$	$\qquad a \in_l b$	;
L25:	SubtypeLL1 $\trianglerighteq$ L17 $\gg$	$\qquad \ell_1 a$	;
L26:	Logic $\trianglerighteq$ L22 $\trianglerighteq$ L24 $\trianglerighteq$ L25 $\gg$	$\qquad a \in_\ell b$	;
L27:	IntroU1U1 $\trianglerighteq$ L23 $\trianglerighteq$ L26 $\gg$	$\qquad u_1(a) \in_\ell u_1(b)$	;
L28:	IntroEll $\trianglerighteq$ L27 $\gg$	$\qquad \ell\,'\,u_1(a)$	;
L29:	Block $\gg$	$\quad$ End	;
L30:	ElimPure $\triangleright$ L19 $\triangleright$ L28 $\gg$	$\quad \ell\,'\,u_1(a)$	;
L31:	Block $\gg$	End	;
L32:	Transfinite" $\triangleright$ L14 $\triangleright$ L30 $\gg$	$\forall a\colon \ell\,'\,u_1(a)$	;
L33:	ElimAll $\trianglerighteq$ L32 $\gg$	$\ell a\colon \ell\,'\,u_1(a)$	]*

[The Map proof of ElimU1 reads

L01:	Hypothesis $\gg$	$\ell\,'\,x$	;
L02:	Hypothesis $\gg$	$\ell\,'\,a$	;
L03:	Hypothesis $\gg$	$x \in_2 u_1(a)$	;
L04:	Define $\gg$	$s \equiv s_2(x, u_1(a))$	;
L05:	Define $\gg$	$t \equiv t_2(x, u_1(a))$	;
L06:	ElimInTwoS $\trianglerighteq$ L3 $\gg$	$\ell\,'\,s$	;
L07:	ElimInTwoT $\trianglerighteq$ L3 $\gg$	$\ell\,'\,t$	;
L08:	ElimInTwo $\trianglerighteq$ L3 $\gg$	$x \sim u_1(a)\,'\,s\,'\,t$	;
L09:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L8 $\gg$	$x \sim \text{if}(s'\mathsf{T}, \mathsf{T}{:}{:}\mathsf{F}, a'(s'\mathsf{F}))'t$	;
L10:	TypeT $\gg$	$\ell\,'\,\mathsf{T}$	;
L11:	TypeF $\gg$	$\ell\,'\,\mathsf{F}$	;
L12:	TypeEquivK $\trianglerighteq$ L1 $\trianglerighteq$ L10 $\gg$	$!x \sim \mathsf{T}$	;
L13:	TypeEquivK $\trianglerighteq$ L1 $\trianglerighteq$ L11 $\gg$	$!x \sim \mathsf{F}$	;
L14:	SubtypeLL2 $\trianglerighteq$ L2 $\gg$	$\ell_2 a$	;
L15:	TypeInTwo $\trianglerighteq$ L1 $\trianglerighteq$ L14 $\gg$	$!x \in_2 a$	;
L16:	TypeApply $\trianglerighteq$ L6 $\trianglerighteq$ L10 $\gg$	$\ell\,'(s\,'\,\mathsf{T})$	;
L17:	SubtypeLB $\trianglerighteq$ L16 $\gg$	$!s\,'\,\mathsf{T}$	;
L18:	TypeApply $\trianglerighteq$ L6 $\trianglerighteq$ L11 $\gg$	$\ell\,'(s\,'\,\mathsf{F})$	;
L19:	SubtypeLB $\trianglerighteq$ L7 $\gg$	$!t$	;
L20:	Block $\gg$	Begin	;
L21:	Hypothesis $\gg$	$s\,'\,\mathsf{T}$	;
L22:	Block $\gg$	Begin	;
L23:	Hypothesis $\gg$	t	;
L24:	Logic $\trianglerighteq$ L21 $\trianglerighteq$ L23 $\gg$	$\text{if}(s'\mathsf{T}, \mathsf{T}{:}{:}\mathsf{F}, a'(s'\mathsf{F}))'t \equiv \mathsf{T}$	;
L25:	Replace' $\trianglerighteq$ L24 $\trianglerighteq$ L9 $\gg$	$x \sim \mathsf{T}$	;
L26:	Logic $\trianglerighteq$ L13 $\trianglerighteq$ L15 $\trianglerighteq$ L25 $\gg$	$x \sim \mathsf{T} \vee x \sim \mathsf{F} \vee x \in_2 a$	;
L27:	Block $\gg$	End	;
L28:	Block $\gg$	Begin	;
L29:	Hypothesis $\gg$	$\neg t$	;
L30:	Logic $\trianglerighteq$ L21 $\trianglerighteq$ L29 $\gg$	$\text{if}(s'\mathsf{T}, \mathsf{T}{:}{:}\mathsf{F}, a'(s'\mathsf{F}))'t \equiv \mathsf{F}$	;
L31:	Replace' $\trianglerighteq$ L30 $\trianglerighteq$ L9 $\gg$	$x \sim \mathsf{F}$	;
L32:	Logic $\trianglerighteq$ L12 $\trianglerighteq$ L15 $\trianglerighteq$ L31 $\gg$	$x \sim \mathsf{T} \vee x \sim \mathsf{F} \vee x \in_2 a$	;
L33:	Block $\gg$	End	;
L34:	TND $\triangleright$ L19 $\triangleright$ L26 $\triangleright$ L32 $\gg$	$x \sim \mathsf{T} \vee x \sim \mathsf{F} \vee x \in_2 a$	;
L35:	Block $\gg$	End	;
L36:	Block $\gg$	Begin	;
L37:	Hypothesis $\gg$	$\neg s\,'\,\mathsf{T}$	;
L38:	Logic $\trianglerighteq$ L37 $\gg$	$\text{if}(s'\mathsf{T}, \mathsf{T}{:}{:}\mathsf{F}, a'(s'\mathsf{F}))'t \equiv$ $a\,'(s\,'\,\mathsf{F})\,'\,\mathsf{T}$	;
L39:	Replace' $\trianglerighteq$ L38 $\trianglerighteq$ L9 $\gg$	$x \sim a\,'(s\,'\,\mathsf{F})\,'\,\mathsf{T}$	;
L40:	IntroInTwo $\trianglerighteq$ L14 $\triangleright$ L1 $\trianglerighteq$ L18 $\trianglerighteq$ L7 $\trianglerighteq$ L39 $\gg$	$x \in_2 a$	;
L41:	Logic $\trianglerighteq$ L12 $\trianglerighteq$ L13 $\trianglerighteq$ L40 $\gg$	$x \sim \mathsf{T} \vee x \sim \mathsf{F} \vee x \in_2 a$	;
L42:	Block $\gg$	End	;
L43:	TND $\triangleright$ L17 $\triangleright$ L34 $\triangleright$ L41 $\gg$	$x \sim \mathsf{T} \vee x \sim \mathsf{F} \vee x \in_2 a$	]*

A.44 Second union operator

The second union operator essentially allows to form the set [$a \cup b$] given the sets [a] and [b].

$$\left[\ \boxed{u_2(a,b)} \doteq \lambda x.\mathrm{if}(x \text{'} \mathsf{T}, a \text{'}(x \text{'} \mathsf{F}), b \text{'}(x \text{'} \mathsf{F}))\ \right]^*$$

[Map lemma

TypeU2L2$_{182}$:	$\ell a,b{:}\, \ell_2 u_2(a,b)$	;
IntroU2a$_{183}$:	$\ell a,b{:}\, a \subseteq_2 u_2(a,b)$	;
IntroU2b$_{183}$:	$\ell a,b{:}\, b \subseteq_2 u_2(a,b)$	;
IntroU2U2$_{184}$:	$\ell \text{'} u_2(c,d) \rightarrow a \in_\ell c \rightarrow b \in_\ell d \rightarrow u_2(a,b) \in_\ell u_1(u_2(c,d))$	;
TypeU2T$_{185}$:	$\forall b{:}\, \ell \text{'} u_2(\mathsf{T},b)$	;
TypeU2$_{186}$:	$\ell a,b{:}\, \ell \text{'} u_2(a,b)$	;
ElimU2$_{187}$:	$\ell a,b{:}\, x \in_2 u_2(a,b) \rightarrow x \in_2 a \vee x \in_2 b$	]*

[The Map proof of TypeU2L2 reads

L01:	Hypothesis ≫	$\ell \text{'} a$	;
L02:	Hypothesis ≫	$\ell \text{'} b$	;
L03:	Block ≫	Begin	;
L04:	Hypothesis ≫	$\quad \ell \text{'} x$	;
L05:	TypeT ≫	$\quad \ell \text{'} \mathsf{T}$	;
L06:	TypeF ≫	$\quad \ell \text{'} \mathsf{F}$	;
L07:	TypeApply ⊵ L4 ⊵ L5 ≫	$\quad \ell \text{'}(x \text{'} \mathsf{T})$	;
L08:	TypeApply ⊵ L4 ⊵ L6 ≫	$\quad \ell \text{'}(x \text{'} \mathsf{F})$	;
L09:	SubtypeLB ⊵ L7 ≫	$\quad !x \text{'} \mathsf{T}$	;
L10:	Block ≫	$\quad$ Begin	;
L11:	Hypothesis ≫	$\quad\quad x \text{'} \mathsf{T}$	;
L12:	Logic ⊵ L11 ≫	$\quad\quad u_2(a,b) \text{'} x \equiv a \text{'}(x \text{'} \mathsf{F})$	;
L13:	TypeApply ⊵ L1 ⊵ L8 ≫	$\quad\quad \ell \text{'}(a \text{'}(x \text{'} \mathsf{F}))$	;
L14:	Replace' ▷ L12 ▷ L13 ≫	$\quad\quad \ell \text{'}(u_2(a,b) \text{'} x)$	;
L15:	Block ≫	$\quad$ End	;
L16:	Block ≫	$\quad$ Begin	;
L17:	Hypothesis ≫	$\quad\quad \neg(x \text{'} \mathsf{T})$	;
L18:	Logic ⊵ L17 ≫	$\quad\quad u_2(a,b) \text{'} x \equiv b \text{'}(x \text{'} \mathsf{F})$	;
L19:	TypeApply ⊵ L2 ⊵ L8 ≫	$\quad\quad \ell \text{'}(b \text{'}(x \text{'} \mathsf{F}))$	;
L20:	Replace' ▷ L18 ▷ L19 ≫	$\quad\quad \ell \text{'}(u_2(a,b) \text{'} x)$	;
L21:	Block ≫	$\quad$ End	;
L22:	TND ▷ L9 ▷ L14 ▷ L20 ≫	$\quad \ell \text{'}(u_2(a,b) \text{'} x)$	;
L23:	Block ≫	End	;
L24:	Gen ▷ L22 ≫	$\ell_1 u_2(a,b)$	;
L25:	SubtypeL1L2 ⊵ L24 ≫	$\ell_2 u_2(a,b)$	]*

[The Map proof of IntroU2a reads

L01:	Hypothesis $\gg$	$\ell\text{'}\,a$	;
L02:	Hypothesis $\gg$	$\ell\text{'}\,b$	;
L03:	SubtypeLL2 $\trianglerighteq$ L1 $\gg$	$\ell_2 a$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$x \in_2 a$	;
L06:	Define $\gg$	$s \equiv s_2(x,a)$	;
L07:	Define $\gg$	$t \equiv t_2(x,a)$	;
L08:	ElimInTwo $\trianglerighteq$ L5 $\gg$	$x \sim a\text{'}\,s\text{'}\,t$	;
L09:	ElimInTwoS $\trianglerighteq$ L5 $\gg$	$\ell\text{'}\,s$	;
L10:	ElimInTwoT $\trianglerighteq$ L5 $\gg$	$\ell\text{'}\,t$	;
L11:	TypeT $\gg$	$\ell\text{'}\,\mathsf{T}$	;
L12:	TypePair $\trianglerighteq$ L11 $\trianglerighteq$ L9 $\gg$	$\ell\text{'}(\mathsf{T}{::}s)$	;
L13:	Reduction $\gg$	$u_2(a,b)\text{'}(\mathsf{T}{::}s)\text{'}\,t \equiv a\text{'}\,s\text{'}\,t$	;
L14:	Replace' $\triangleright$ L13 $\triangleright$ L8 $\gg$	$x \sim u_2(a,b)\text{'}(\mathsf{T}{::}s)\text{'}\,t$	;
L15:	TypeU2L2 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell_2 u_2(a,b)$	;
L16:	IntroInTwo$\trianglerighteq$L15$\trianglerighteq$L12$\trianglerighteq$L10$\trianglerighteq$L14$\gg$	$x \in_2 u_2(a,b)$	;
L17:	Block $\gg$	End	;
L18:	IntroSub $\triangleright$ L3 $\triangleright$ L16 $\gg$	$a \subseteq_2 u_2(a,b)$	]*

[The Map proof of IntroU2b reads

L01:	Hypothesis $\gg$	$\ell\text{'}\,a$	;
L02:	Hypothesis $\gg$	$\ell\text{'}\,b$	;
L03:	SubtypeLL2 $\trianglerighteq$ L1 $\gg$	$\ell_2 b$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$x \in_2 b$	;
L06:	Define $\gg$	$s \equiv s_2(x,b)$	;
L07:	Define $\gg$	$t \equiv t_2(x,b)$	;
L08:	ElimInTwo $\trianglerighteq$ L5 $\gg$	$x \sim b\text{'}\,s\text{'}\,t$	;
L09:	ElimInTwoS $\trianglerighteq$ L5 $\gg$	$\ell\text{'}\,s$	;
L10:	ElimInTwoT $\trianglerighteq$ L5 $\gg$	$\ell\text{'}\,t$	;
L11:	TypeF $\gg$	$\ell\text{'}\,\mathsf{F}$	;
L12:	TypeApply $\trianglerighteq$ L11 $\trianglerighteq$ L9 $\gg$	$\ell\text{'}(\mathsf{F}{::}s)$	;
L13:	Reduction $\gg$	$u_2(a,b)\text{'}(\mathsf{F}{::}s)\text{'}\,t \equiv b\text{'}\,s\text{'}\,t$	;
L14:	Replace' $\triangleright$ L13 $\triangleright$ L8 $\gg$	$x \sim u_2(a,b)\text{'}(\mathsf{F}{::}s)\text{'}\,t$	;
L15:	TypeU2L2 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell_2 u_2(a,b)$	;
L16:	IntroInTwo$\trianglerighteq$L15$\trianglerighteq$L12$\trianglerighteq$L10$\trianglerighteq$L14$\gg$	$x \in_2 u_2(a,b)$	;
L17:	Block $\gg$	End	;
L18:	IntroSub $\triangleright$ L3 $\triangleright$ L16 $\gg$	$b \subseteq_2 u_2(a,b)$	]*

[The Map proof of IntroU2U2 reads

L01:	Hypothesis $\gg$	$\ell \,' u_2(c,d)$	;
L02:	Hypothesis $\gg$	$a \in_\ell c$	;
L03:	Hypothesis $\gg$	$b \in_\ell d$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$x \sim_{u_1(u_2(c,d))} \underline{x}$	;
L06:	IntroU1T $\trianglerighteq$ L1 $\gg$	$\mathsf{T} \in_2 u_1(u_2(c,d))$	;
L07:	IntroU1F $\trianglerighteq$ L1 $\gg$	$\mathsf{F} \in_2 u_1(u_2(c,d))$	;
L08:	IntroU1a $\trianglerighteq$ L1 $\gg$	$u_2(c,d) \subseteq_2 u_1(u_2(c,d))$	;
L09:	StrictInEllP $\trianglerighteq$ L2 $\gg$	$\ell \,' c$	;
L10:	StrictInEllP $\trianglerighteq$ L3 $\gg$	$\ell \,' d$	;
L11:	IntroU2a $\trianglerighteq$ L9 $\trianglerighteq$ L10 $\gg$	$c \subseteq_2 u_2(c,d)$	;
L12:	IntroU2b $\trianglerighteq$ L9 $\trianglerighteq$ L10 $\gg$	$d \subseteq_2 u_2(c,d)$	;
L13:	TransSub $\trianglerighteq$ L11 $\trianglerighteq$ L8 $\gg$	$c \subseteq_2 u_1(u_2(c,d))$	;
L14:	TransSub $\trianglerighteq$ L12 $\trianglerighteq$ L8 $\gg$	$d \subseteq_2 u_1(u_2(c,d))$	;
L15:	ElimEquivP $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$x \,' \mathsf{T} \sim_{u_1(u_2(c,d))} \underline{x} \,' \mathsf{T}$	;
L16:	ElimEquivP $\trianglerighteq$ L5 $\trianglerighteq$ L7 $\gg$	$x \,' \mathsf{F} \sim_{u_1(u_2(c,d))} \underline{x} \,' \mathsf{F}$	;
L17:	SubEquivP $\trianglerighteq$ L13 $\trianglerighteq$ L16 $\gg$	$x \,' \mathsf{F} \sim_c \underline{x} \,' \mathsf{F}$	;
L18:	SubEquivP $\trianglerighteq$ L14 $\trianglerighteq$ L16 $\gg$	$x \,' \mathsf{F} \sim_d \underline{x} \,' \mathsf{F}$	;
L19:	ElimInEll $\trianglerighteq$ L2 $\trianglerighteq$ L17 $\gg$	$a \,'(x \,' \mathsf{F}) \sim a \,'(\underline{x} \,' \mathsf{F})$	;
L20:	ElimInEll $\trianglerighteq$ L3 $\trianglerighteq$ L18 $\gg$	$b \,'(x \,' \mathsf{F}) \sim b \,'(\underline{x} \,' \mathsf{F})$	;
L21:	ExtIfPKK$\trianglerighteq$L15$\trianglerighteq$L19$\trianglerighteq$L20$\gg$	$\mathsf{if}(x \,' \mathsf{T}, a \,'(x \,' \mathsf{F}), b \,'(x \,' \mathsf{F})) \sim$	
		$\mathsf{if}(\underline{x} \,' \mathsf{T}, a \,'(\underline{x} \,' \mathsf{F}), b \,'(\underline{x} \,' \mathsf{F}))$	;
L22:	Trans $\triangleright \bar{\mathcal{R}} \triangleright$ L21 $\gg$	$u_2(a,b) \,' x \sim u_2(a,b) \,' \underline{x}$	;
L23:	Block $\gg$	End	;
L24:	TypeU1 $\trianglerighteq$ L1 $\gg$	$\ell \,' u_1(u_2(c,d))$	;
L25:	IntroInEll $\trianglerighteq$ L24 $\trianglerighteq$ L22 $\gg$	$u_2(a,b) \in_\ell u_1(u_2(c,d))$	]*

[The Map proof of TypeU2T reads

L01:	Block $\gg$	Begin	;
L02:	Hypothesis $\gg$	$\ell\text{'}\,b$	;
L03:	ElimEll $\trianglerighteq$ L2 $\gg$	$b \in_\ell \mathcal{I}b$	;
L04:	StrictInEllP $\trianglerighteq$ L3 $\gg$	$\ell\text{'}\,\mathcal{I}b$	;
L05:	TypeU1 $\trianglerighteq$ L4 $\gg$	$\ell\text{'}\,u_1(\mathcal{I}b)$	;
L06:	IntroU1T $\trianglerighteq$ L4 $\gg$	$\mathsf{T} \in_2 u_1(\mathcal{I}b)$	;
L07:	IntroU1F $\trianglerighteq$ L4 $\gg$	$\mathsf{F} \in_2 u_1(\mathcal{I}b)$	;
L08:	IntroU1a $\trianglerighteq$ L4 $\gg$	$\mathcal{I}b \subseteq_2 u_1(\mathcal{I}b)$	;
L09:	Block $\gg$	Begin	;
L10:	Hypothesis $\gg$	$x \sim_{u_1(\mathcal{I}b)} \underline{x}$	;
L11:	ElimEquivP $\trianglerighteq$ L10 $\trianglerighteq$ L6 $\gg$	$x\text{'}\,\mathsf{T} \sim_{u_1(\mathcal{I}b)} \underline{x}\text{'}\,\mathsf{T}$	;
L12:	ExtT $\gg$	$\mathsf{T} \sim \mathsf{T}$	;
L13:	ElimEquivP $\trianglerighteq$ L10 $\trianglerighteq$ L7 $\gg$	$x\text{'}\,\mathsf{F} \sim_{u_1(\mathcal{I}b)} \underline{x}\text{'}\,\mathsf{F}$	;
L14:	SubEquivP $\trianglerighteq$ L8 $\trianglerighteq$ L13 $\gg$	$x\text{'}\,\mathsf{F} \sim_{\mathcal{I}b} \underline{x}\text{'}\,\mathsf{F}$	;
L15:	ElimInEll $\trianglerighteq$ L3 $\trianglerighteq$ L14 $\gg$	$b\text{'}(x\text{'}\,\mathsf{F}) \sim b\text{'}(\underline{x}\text{'}\,\mathsf{F})$	;
L16:	ExtIfPKK $\trianglerighteq$ L11 $\trianglerighteq$ L12 $\trianglerighteq$ L $\gg$	$\text{if}(x\text{'}\,\mathsf{T}, \mathsf{T}, b\text{'}(x\text{'}\,\mathsf{F})) \sim$	
		$\text{if}(\underline{x}\text{'}\,\mathsf{T}, \mathsf{T}, b\text{'}(\underline{x}\text{'}\,\mathsf{F}))$	;
L17:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L16 $\gg$	$u_2(\mathsf{T}, b)\text{'}\,x \sim u_2(\mathsf{T}, b)\text{'}\,\underline{x}$	;
L18:	Block $\gg$	End	;
L19:	IntroInEll $\triangleright$ L5 $\triangleright$ L17 $\gg$	$u_2(\mathsf{T}, b) \in_\ell u_1(\mathcal{I}b)$	;
L20:	IntroEll $\trianglerighteq$ L19 $\gg$	$\ell\text{'}\,u_2(\mathsf{T}, b)$	;
L21:	Block $\gg$	End	;
L22:	Gen $\triangleright$ L20 $\gg$	$\forall b\colon \ell\text{'}\,u_2(\mathsf{T}, b)$	]*

[The Map proof of TypeU2 reads
```
L01: TypeU2T $\gg$                                 $\forall b\colon \ell\,'\,u_2(\mathsf{T}, b)$          ;
L02: Block $\gg$                                   Begin                                                 ;
L03: Hypothesis $\gg$                                $\neg a$                                            ;
L04: Hypothesis $\gg$                                $\ell\,'\,a$                                        ;
L05: Hypothesis $\gg$                                $\forall c\colon \forall b\colon \ell\,'\,u_2(a\,'\,c, b)$  ;
L06: Hypothesis $\gg$                                $\mathsf{E}c\colon \ell\,'\,c \wedge \forall b\colon \ell\,'\,u_2(c, b) \wedge a \in_l c$  ;
L07: Block $\gg$                                     Begin                                               ;
L08: Hypothesis $\gg$                                  $\ell\,'\,c \wedge \forall b\colon \ell\,'\,u_2(c, b) \wedge a \in_l c$  ;
L09: Logic $\trianglerighteq$ L8 $\gg$                 $\ell\,'\,c$                                      ;
L10: Logic $\trianglerighteq$ L8 $\gg$                 $\forall b\colon \ell\,'\,u_2(c, b)$              ;
L11: Logic $\trianglerighteq$ L8 $\gg$                 $a \in_l c$                                       ;
L12: SubtypeLL1 $\trianglerighteq$ L4 $\gg$            $\ell_1 a$                                        ;
L13: Logic $\trianglerighteq$ L9 $\trianglerighteq$ L11 $\trianglerighteq$ L12 $\gg$   $a \in_\ell c$        ;
L14: Block $\gg$                                       Begin                                             ;
L15: Hypothesis $\gg$                                    $\ell\,'\,b$                                    ;
L16: ElimEll $\trianglerighteq$ L15 $\gg$                $b \in_\ell \mathcal{I}b$                       ;
L17: StrictInEllP $\trianglerighteq$ L16 $\gg$           $\ell\,'\,\mathcal{I}b$                         ;
L18: ElimAll $\trianglerighteq$ L10 $\trianglerighteq$ L17 $\gg$   $\ell\,'\,u_2(c, \mathcal{I}b)$       ;
L19: IntroU2U2$\trianglerighteq$L18$\trianglerighteq$L13$\trianglerighteq$L16$\gg$   $u_2(a, b) \in_\ell u_1(u_2(c, \mathcal{I}b))$  ;
L20: IntroEll $\trianglerighteq$ L19 $\gg$               $\ell\,'\,u_2(a, b)$                            ;
L21: Block $\gg$                                       End                                               ;
L22: Gen $\triangleright$ L20 $\gg$                    $\forall b\colon \ell\,'\,u_2(a, b)$              ;
L23: Block $\gg$                                     End                                                 ;
L24: ElimPure $\triangleright$ L6 $\triangleright$ L22 $\gg$   $\forall b\colon \ell\,'\,u_2(a, b)$      ;
L25: Block $\gg$                                   End                                                   ;
L26: Transfinite" $\triangleright$ L1 $\triangleright$ L24 $\gg$   $\forall a\colon \forall b\colon \ell\,'\,u_2(a, b)$   ;
L27: ElimAll2 $\trianglerighteq$ L26 $\gg$         $\ell a, b\colon \ell\,'\,u_2(a, b)$                  ]*
```

[The Map proof of ElimU2 reads

L01:	Hypothesis $\gg$	$\ell\text{'}a$	;
L02:	Hypothesis $\gg$	$\ell\text{'}b$	;
L03:	Hypothesis $\gg$	$x \in_2 u_2(a,b)$	;
L04:	StrictInTwoX $\unrhd$ L3 $\gg$	$\ell\text{'}x$	;
L05:	Define $\gg$	$s \equiv s_2(x, u_2(a,b))$	;
L06:	Define $\gg$	$t \equiv t_2(x, u_2(a,b))$	;
L07:	ElimInTwoS $\unrhd$ L3 $\gg$	$\ell\text{'}s$	;
L08:	ElimInTwoT $\unrhd$ L3 $\gg$	$\ell\text{'}t$	;
L09:	ElimInTwo $\unrhd$ L3 $\gg$	$x \sim u_2(a,b)\text{'}s\text{'}t$	;
L10:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L9 $\gg$	$x \sim \mathrm{if}(s\text{'}\mathsf{T}, a\text{'}(s\text{'}\mathsf{F}), b\text{'}(s\text{'}\mathsf{F}))\text{'}t$	;
L11:	TypeT $\gg$	$\ell\text{'}\mathsf{T}$	;
L12:	TypeF $\gg$	$\ell\text{'}\mathsf{F}$	;
L13:	SubtypeLL2 $\unrhd$ L1 $\gg$	$\ell_2 a$	;
L14:	TypeInTwo $\unrhd$ L4 $\unrhd$ L13 $\gg$	$!x \in_2 a$	;
L15:	SubtypeLL2 $\unrhd$ L2 $\gg$	$\ell_2 b$	;
L16:	TypeInTwo $\unrhd$ L4 $\unrhd$ L15 $\gg$	$!x \in_2 b$	;
L17:	TypeApply $\unrhd$ L7 $\unrhd$ L11 $\gg$	$\ell\text{'}(s\text{'}\mathsf{T})$	;
L18:	SubtypeLB $\unrhd$ L17 $\gg$	$!s\text{'}\mathsf{T}$	;
L19:	TypeApply $\unrhd$ L7 $\unrhd$ L12 $\gg$	$\ell\text{'}(s\text{'}\mathsf{F})$	;
L20:	Block $\gg$	Begin	;
L21:	Hypothesis $\gg$	$s\text{'}\mathsf{T}$	;
L22:	Logic $\unrhd$ L21 $\gg$	$\mathrm{if}(s\text{'}\mathsf{T}, a\text{'}(s\text{'}\mathsf{F}), b\text{'}(s\text{'}\mathsf{F}))\text{'}t \equiv$ $a\text{'}(s\text{'}\mathsf{F})\text{'}t$	;
L23:	Replace' $\rhd$ L22 $\rhd$ L10 $\gg$	$x \sim a\text{'}(s\text{'}\mathsf{F})\text{'}t$	;
L24:	IntroInTwo $\unrhd$ L13 $\unrhd$ L4 $\unrhd$ L19 $\unrhd$ L8 $\unrhd$ L23 $\gg$	$x \in_2 a$	;
L25:	Logic $\unrhd$ L16 $\unrhd$ L24 $\gg$	$x \in_2 a \vee x \in_2 b$	;
L26:	Block $\gg$	End	;
L27:	Block $\gg$	Begin	;
L28:	Hypothesis $\gg$	$\neg s\text{'}\mathsf{T}$	;
L29:	Logic $\unrhd$ L28 $\gg$	$\mathrm{if}(s\text{'}\mathsf{T}, a\text{'}(s\text{'}\mathsf{F}), b\text{'}(s\text{'}\mathsf{F}))\text{'}t \equiv$ $b\text{'}(s\text{'}\mathsf{F})\text{'}t$	;
L30:	Replace' $\rhd$ L29 $\rhd$ L10 $\gg$	$x \sim b\text{'}(s\text{'}\mathsf{F})\text{'}t$	;
L31:	IntroInTwo $\unrhd$ L15 $\unrhd$ L1 $\unrhd$ L19 $\unrhd$ L8 $\unrhd$ L30 $\gg$	$x \in_2 b$	;
L32:	Logic $\unrhd$ L14 $\unrhd$ L31 $\gg$	$x \in_2 a \vee x \in_2 b$	;
L33:	Block $\gg$	End	;
L34:	TND $\rhd$ L18 $\rhd$ L25 $\rhd$ L32 $\gg$	$x \in_2 a \vee x \in_2 b$	]*

A.45 Third union operator

The third union operator is a very technical construct needed for proving properties about the fourth union operator which is introduced in the next section.

$$\Big[\ \boxed{u_3} \doteq \lambda c.\varepsilon b\colon c \in_\ell \mathsf{K}\text{'}b\ \Big]^*$$

[Map lemma

TypeU3$_{188}$:	$\ell_1 u_3$	;
TypeU3a$_{188}$:	$\ell a{:}\,\ell\,\text{'}(u_3 \circ a)$	;
IntroU3Hyp$_{188}$:	$\ell a, c{:}\,\forall c{:}\,\exists b{:}\, a\,\text{'}\,c \in_\ell \mathsf{K}\,\text{'}\,b \to a\,\text{'}\,c \in_\ell u_3 \circ a$	]*

[The Map proof of TypeU3 reads

L01:	Block $\gg$	Begin	;
L02:	Hypothesis $\gg$	$\ell\,\text{'}\,c$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\ell\,\text{'}\,b$	;
L05:	TypeKa $\trianglerighteq$ L4 $\gg$	$\ell\,\text{'}(\mathsf{K}\,\text{'}\,b)$	;
L06:	SubtypeLL1 $\trianglerighteq$ L2 $\gg$	$\ell_1 c$	;
L07:	TypeInEll $\trianglerighteq$ L6 $\trianglerighteq$ L5 $\gg$	$!c \in_\ell \mathsf{K}\,\text{'}\,b$	;
L08:	Block $\gg$	End	;
L09:	TypeChoice $\trianglerighteq$ (Gen $\triangleright$ L7) $\gg$	$\ell\,\text{'}\,\varepsilon b{:}\, c \in_\ell \mathsf{K}\,\text{'}\,b$	;
L10:	Block $\gg$	End	;
L11:	Gen $\triangleright$ L9 $\gg$	$\ell_1 u_3$	]*

[The Map proof of TypeU3a reads

L1:	Hypothesis $\gg$	$\ell\,\text{'}\,a$	;
L2:	TypeU3 $\gg$	$\ell_1 u_3$	;
L3:	TypeCompose $\trianglerighteq$ L2 $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}(u_3 \circ a)$	]*

[The Map proof of IntroU3Hyp reads

L1:	Hypothesis $\gg$	$\ell\,\text{'}\,a$	;
L2:	Hypothesis $\gg$	$\ell\,\text{'}\,c$	;
L3:	Hypothesis $\gg$	$\forall c{:}\,\exists b{:}\, a\,\text{'}\,c \in_\ell \mathsf{K}\,\text{'}\,b$	;
L4:	ElimAll $\trianglerighteq$ L3 $\trianglerighteq$ L2 $\gg$	$\exists b{:}\, a\,\text{'}\,c \in_\ell \mathsf{K}\,\text{'}\,b$	;
L5:	ElimExists $\trianglerighteq$ L4 $\gg$	$a\,\text{'}\,c \in_\ell \mathsf{K}\,\text{'}\,\varepsilon b{:}\, a\,\text{'}\,c \in_\ell \mathsf{K}\,\text{'}\,b$	;
L6:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L5 $\gg$	$a\,\text{'}\,c \in_\ell \mathsf{K}\,\text{'}((u_3 \circ a)\,\text{'}\,c)$	;
L7:	TypeU3a $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}(u_3 \circ a)$	;
L8:	IntroKSub $\trianglerighteq$ L7 $\trianglerighteq$ L2 $\gg$	$\mathsf{K}\,\text{'}((u_3 \circ a)\,\text{'}\,c) \subseteq_2 u_3 \circ a$	;
L9:	SubInEll $\trianglerighteq$ L7 $\trianglerighteq$ L8 $\trianglerighteq$ L6 $\gg$	$a\,\text{'}\,c \in_\ell u_3 \circ a$	]*

A.46 Fourth union operator

The fourth union operator essentially allows to form the union set [$\cup a$] given a set [a].

$$\Big[\ \boxed{u_4(a)} \doteq \lambda x.a\,\text{'}(x\,\text{'}\,\mathsf{T})\,\text{'}(x\,\text{'}\,\mathsf{F})\ \Big]^*$$

[Map lemma

$\mathrm{IntroU4'}_{189}$:	$\ell a\colon a \subseteq_2 \mathsf{K}\text{'}\,u_4(a)$	;
$\mathrm{TypeU4Hyp}_{190}$:	$\ell a\colon \forall c\colon \exists b\colon a\text{'}\,c \in_\ell \mathsf{K}\text{'}\,b \to \ell\text{'}\,u_4(a)$	;
$\mathrm{ExistsUnion1}_{191}$:	$\exists b\colon \mathsf{T} \subseteq_2 \mathsf{K}\text{'}\,b$	;
$\mathrm{ExistsUnion2}_{191}$:	$\exists b\colon \mathsf{T} \in_\ell \mathsf{K}\text{'}\,b$	;
$\mathrm{ExistsUnion3}_{192}$:	$\ell a\colon \forall c\colon \exists b\colon a\text{'}\,c \in_\ell \mathsf{K}\text{'}\,b \to \exists b\colon a \subseteq_2 \mathsf{K}\text{'}\,b$	;
$\mathrm{ExistsUnion}_{193}$:	$\forall a\colon (\exists b\colon a \subseteq_2 \mathsf{K}\text{'}\,b) \wedge (\exists b\colon a \in_\ell \mathsf{K}\text{'}\,b)$	;
$\mathrm{UnionHyp}_{194}$:	$\ell a\colon \forall c\colon \exists b\colon a\text{'}\,c \in_\ell \mathsf{K}\text{'}\,b$	;
$\mathrm{TypeU4}_{194}$:	$\ell a\colon \ell\text{'}\,u_4(a)$	;
$\mathrm{IntroU3}_{194}$:	$\ell a, c\colon a\text{'}\,c \in_\ell u_3 \circ a$	;
$\mathrm{ElimU4}_{194}$:	$\ell x, y\colon x \in_2 u_4(y) \to \exists z\colon x \in_2 y\text{'}\,z$	;
$\mathrm{IntroU4}_{195}$:	$\ell y, z\colon x \in_2 y\text{'}\,z \to x \in_2 u_4(y)$	;
$\mathrm{IntroSubU4}_{195}$:	$\ell x, y\colon x\text{'}\,y \subseteq_2 u_4(x)$	;
$\mathrm{ElimU4Sub}_{195}$:	$\ell x, y, z\colon u_4(x) \subseteq_2 y \to x\text{'}\,z \subseteq_2 y$	]*

[The Map proof of IntroU4' reads

L01:	Hypothesis $\gg$	$\ell\text{'}\,a$	;
L02:	SubtypeLL2 $\trianglerighteq$ L1 $\gg$	$\ell_2 a$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$x \in_2 a$	;
L05:	Define $\gg$	$s \equiv s_2(x,a)$	;
L06:	Define $\gg$	$t \equiv t_2(x,a)$	;
L07:	ElimInTwo $\trianglerighteq$ L4 $\gg$	$x \sim a\text{'}\,s\text{'}\,t$	;
L08:	ElimInTwoS $\trianglerighteq$ L4 $\gg$	$\ell\text{'}\,s$	;
L09:	ElimInTwoT $\trianglerighteq$ L4 $\gg$	$\ell\text{'}\,t$	;
L10:	TypePair $\trianglerighteq$ L8 $\trianglerighteq$ L9 $\gg$	$\ell\text{'}(s :: t)$	;
L11:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L7 $\gg$	$x \sim \mathsf{K}\text{'}\,u_4(a)\text{'}\,s\text{'}(s :: t)$	;
L12:	Block $\gg$	Begin	;
L13:	Hypothesis $\gg$	$\ell\text{'}\,u$	;
L14:	Hypothesis $\gg$	$\ell\text{'}\,v$	;
L15:	TypeT $\gg$	$\ell\text{'}\,\mathsf{T}$	;
L16:	TypeApply $\trianglerighteq$ L14 $\trianglerighteq$ L15 $\gg$	$\ell\text{'}(v\text{'}\,\mathsf{T})$	;
L17:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L16 $\gg$	$\ell\text{'}(a\text{'}(v\text{'}\,\mathsf{T}))$	;
L18:	TypeF $\gg$	$\ell\text{'}\,\mathsf{F}$	;
L19:	TypeApply $\trianglerighteq$ L14 $\trianglerighteq$ L18 $\gg$	$\ell\text{'}(v\text{'}\,\mathsf{F})$	;
L20:	TypeApply $\trianglerighteq$ L17 $\trianglerighteq$ L19 $\gg$	$\ell(\text{'}\,a\text{'}(v\text{'}\,\mathsf{T})\text{'}(v\text{'}\,\mathsf{F}))$	;
L21:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L20 $\gg$	$\ell\text{'}(\mathsf{K}\text{'}\,u_4(a)\text{'}\,u\text{'}\,v)$	;
L22:	Block $\gg$	End	;
L23:	Gen2 $\triangleright$ L21 $\gg$	$\ell_2 \mathsf{K}\text{'}\,u_4(a)$	;
L24:	IntroInTwo$\trianglerighteq$L23$\trianglerighteq$L8$\trianglerighteq$L10$\trianglerighteq$L11$\gg$	$x \in_2 \mathsf{K}\text{'}\,u_4(a)$	;
L25:	Block $\gg$	End	;
L26:	IntroSub $\triangleright$ L2 $\triangleright$ L24 $\gg$	$a \subseteq_2 \mathsf{K}\text{'}\,u_4(a)$	]*

[The Map proof of TypeU4Hyp reads

$\mathsf{L01}$:	Hypothesis $\gg$	$\ell\,'\,a$	;
$\mathsf{L02}$:	Hypothesis $\gg$	$\forall c\colon \exists b\colon a\,'\,c \in_\ell \mathsf{K}\,'\,b$	;
$\mathsf{L03}$:	TypeU3a $\unrhd$ $\mathsf{L1}$ $\gg$	$\ell\,'(u_3 \circ a)$	;
$\mathsf{L04}$:	ElimEll $\unrhd$ $\mathsf{L1}$ $\gg$	$a \in_\ell \mathcal{I}a$	;
$\mathsf{L05}$:	StrictInEllP $\unrhd$ $\mathsf{L4}$ $\gg$	$\ell\,'\,\mathcal{I}a$	;
$\mathsf{L06}$:	Define $\gg$	$j \equiv u_2(\mathcal{I}a, u_3 \circ a)$	;
$\mathsf{L07}$:	TypeU2 $\unrhd$ $\mathsf{L5}$ $\unrhd$ $\mathsf{L3}$ $\gg$	$\ell\,'\,j$	;
$\mathsf{L08}$:	IntroU2a $\unrhd$ $\mathsf{L5}$ $\unrhd$ $\mathsf{L3}$ $\gg$	$\mathcal{I}a \subseteq_2 j$	;
$\mathsf{L09}$:	IntroU2b $\unrhd$ $\mathsf{L5}$ $\unrhd$ $\mathsf{L3}$ $\gg$	$u_3 \circ a \subseteq_2 j$	;
$\mathsf{L10}$:	Define $\gg$	$k \equiv u_1(j)$	;
$\mathsf{L11}$:	TypeU1 $\unrhd$ $\mathsf{L7}$ $\gg$	$\ell\,'\,k$	;
$\mathsf{L12}$:	IntroU1T $\unrhd$ $\mathsf{L7}$ $\gg$	$\mathsf{T} \in_2 k$	;
$\mathsf{L13}$:	IntroU1F $\unrhd$ $\mathsf{L7}$ $\gg$	$\mathsf{F} \in_2 k$	;
$\mathsf{L14}$:	IntroU1a $\unrhd$ $\mathsf{L7}$ $\gg$	$j \subseteq_2 k$	;
$\mathsf{L15}$:	TransSub $\unrhd$ $\mathsf{L8}$ $\unrhd$ $\mathsf{L14}$ $\gg$	$\mathcal{I}a \subseteq_2 k$	;
$\mathsf{L16}$:	TransSub $\unrhd$ $\mathsf{L9}$ $\unrhd$ $\mathsf{L14}$ $\gg$	$u_3 \circ a \subseteq_2 k$	;
$\mathsf{L17}$:	SubInEll $\unrhd$ $\mathsf{L11}$ $\unrhd$ $\mathsf{L15}$ $\unrhd$ $\mathsf{L4}$ $\gg$	$a \in_\ell k$	;
$\mathsf{L18}$:	Block $\gg$	Begin	;
$\mathsf{L19}$:	Hypothesis $\gg$	$x \sim_k \underline{x}$	;
$\mathsf{L20}$:	ExtT $\gg$	$\mathsf{T} \sim \mathsf{T}$	;
$\mathsf{L21}$:	ExtPApplyK $\unrhd$ $\mathsf{L12}$ $\unrhd$ $\mathsf{L19}$ $\unrhd$ $\mathsf{L20}$ $\gg$	$x\,'\,\mathsf{T} \sim_k \underline{x}\,'\,\mathsf{T}$	;
$\mathsf{L22}$:	ExtF $\gg$	$\mathsf{F} \sim \mathsf{F}$	;
$\mathsf{L23}$:	ExtPApplyK $\unrhd$ $\mathsf{L13}$ $\unrhd$ $\mathsf{L19}$ $\unrhd$ $\mathsf{L22}$ $\gg$	$x\,'\,\mathsf{F} \sim_k \underline{x}\,'\,\mathsf{F}$	;
$\mathsf{L24}$:	ElimInEll $\unrhd$ $\mathsf{L17}$ $\unrhd$ $\mathsf{L21}$ $\gg$	$a\,'(x\,'\,\mathsf{T}) \sim a\,'(\underline{x}\,'\,\mathsf{T})$	;
$\mathsf{L25}$:	StrictEquivPX $\unrhd$ $\mathsf{L19}$ $\gg$	$\ell\,'\,x$	;
$\mathsf{L26}$:	TypeT $\gg$	$\ell\,'\,\mathsf{T}$	;
$\mathsf{L27}$:	TypeApply $\unrhd$ $\mathsf{L25}$ $\unrhd$ $\mathsf{L26}$ $\gg$	$\ell\,'(x\,'\,\mathsf{T})$	;
$\mathsf{L28}$:	IntroU3Hyp $\unrhd$ $\mathsf{L1}$ $\unrhd$ $\mathsf{L27}$ $\unrhd$ $\mathsf{L2}$ $\gg$	$a\,'(x\,'\,\mathsf{T}) \in_\ell u_3 \circ a$	;
$\mathsf{L29}$:	SubInEll $\unrhd$ $\mathsf{L11}$ $\unrhd$ $\mathsf{L16}$ $\unrhd$ $\mathsf{L28}$ $\gg$	$a\,'(x\,'\,\mathsf{T}) \in_\ell k$	;
$\mathsf{L30}$:	ExtKApplyP $\unrhd$ $\mathsf{L29}$ $\unrhd$ $\mathsf{L24}$ $\unrhd$ $\mathsf{L23}$ $\gg$	$a\,'(x\,'\,\mathsf{T})\,'(x\,'\,\mathsf{F}) \sim$ $a\,'(\underline{x}\,'\,\mathsf{T})\,'(\underline{x}\,'\,\mathsf{F})$	;
$\mathsf{L31}$:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ $\mathsf{L30}$ $\gg$	$u_4(a)\,'\,x \sim u_4(a)\,'\,\underline{x}$	;
$\mathsf{L32}$:	Block $\gg$	End	;
$\mathsf{L33}$:	IntroEll $\unrhd$ (IntroInEll $\triangleright$ $\mathsf{L11}$ $\triangleright$ $\mathsf{L31}$) $\gg$	$\ell\,'\,u_4(a)$	]*

[The Map proof of ExistsUnion1 reads

L01:	TypeT $\gg$	$\ell\text{'}\mathsf{T}$	;
L02:	SubtypeLL2 $\trianglerighteq$ L1 $\gg$	$\ell_2\mathsf{T}$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\ell\text{'}b$	;
L05:	TypeKa $\trianglerighteq$ L4 $\gg$	$!\mathsf{K}\text{'}b$	;
L06:	SubtypeLL2 $\trianglerighteq$ L5 $\gg$	$\ell_2\mathsf{K}\text{'}b$	;
L07:	TypeSub $\trianglerighteq$ L2 $\trianglerighteq$ L6 $\gg$	$!\mathsf{T}\subseteq_2\mathsf{K}\text{'}b$	;
L08:	Block $\gg$	End	;
L09:	TypeKa $\trianglerighteq$ L1 $\gg$	$\ell\text{'}(\mathsf{K}\text{'}\mathsf{T})$	;
L10:	SubtypeLL2 $\trianglerighteq$ L9 $\gg$	$\ell_2\mathsf{K}\text{'}\mathsf{T}$	;
L11:	Block $\gg$	Begin	;
L12:	Hypothesis $\gg$	$x\in_2\mathsf{T}$	;
L13:	ElimInTwo $\trianglerighteq$ L12 $\gg$	$x\sim\mathsf{T}\text{'}s_2(x,\mathsf{T})\text{'}t_2(x,\mathsf{T})$	;
L14:	Logic $\trianglerighteq$ L13 $\gg$	$x\sim\mathsf{K}\text{'}\mathsf{T}\text{'}\mathsf{T}\text{'}\mathsf{T}$	;
L15:	IntroInTwo$\trianglerighteq$L10$\trianglerighteq$L1$\trianglerighteq$L1$\trianglerighteq$L14$\gg$	$x\in_2\mathsf{K}\text{'}\mathsf{T}$	;
L16:	Block $\gg$	End	;
L17:	IntroSub $\triangleright$ L10 $\triangleright$ L15 $\gg$	$\mathsf{T}\subseteq_2\mathsf{K}\text{'}\mathsf{T}$	;
L18:	IntroExists$\trianglerighteq$(Gen$\triangleright$L7)$\trianglerighteq$L1$\trianglerighteq$L17$\gg$	$\exists b\colon\mathsf{T}\subseteq_2\mathsf{K}\text{'}b$	]*

[The Map proof of ExistsUnion2 reads

L01:	TypeT $\gg$	$\ell\text{'}\mathsf{T}$	;
L02:	TypeKa $\trianglerighteq$ L1 $\gg$	$\ell\text{'}(\mathsf{K}\text{'}\mathsf{T})$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\ell\text{'}b$	;
L05:	SubtypeLL1 $\trianglerighteq$ L1 $\gg$	$\ell_1\mathsf{T}$	;
L06:	TypeKa $\trianglerighteq$ L4 $\gg$	$\ell\text{'}(\mathsf{K}\text{'}b)$	;
L07:	TypeInEll $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$!(\mathsf{T}\in_\ell\mathsf{K}\text{'}b)$	;
L08:	Block $\gg$	End	;
L09:	Block $\gg$	Begin	;
L10:	Hypothesis $\gg$	$x\sim_{\mathsf{K}\text{'}\mathsf{T}}\underline{x}$	;
L11:	Reduction $\gg$	$\mathsf{T}\text{'}(\mathsf{K}\text{'}\mathsf{T}\text{'}x)\sim\mathsf{T}\text{'}(\mathsf{K}\text{'}\mathsf{T}\text{'}\underline{x})$	;
L12:	Block $\gg$	End	;
L13:	IntroInEll $\triangleright$ L2 $\triangleright$ L11 $\gg$	$\mathsf{T}\in_\ell\mathsf{K}\text{'}\mathsf{T}$	;
L14:	IntroExists$\trianglerighteq$(Gen$\triangleright$L7)$\trianglerighteq$L1$\trianglerighteq$L13$\gg$	$\exists b\colon\mathsf{T}\in_\ell\mathsf{K}\text{'}b$	]*

[The Map proof of ExistsUnion3 reads

L01:	Hypothesis $\gg$	$\ell\,'\,a$	;
L02:	Hypothesis $\gg$	$\forall c\colon \exists b\colon a\,'\,c \in_\ell \mathsf{K}\,'\,b$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\ell\,'\,b$	;
L05:	TypeKa $\unrhd$ L4 $\gg$	$\ell\,'(\mathsf{K}\,'\,b)$	;
L06:	SubtypeLL2 $\unrhd$ L5 $\gg$	$\ell_2 \mathsf{K}\,'\,b$	;
L07:	SubtypeLL2 $\unrhd$ L1 $\gg$	$\ell_2 a$	;
L08:	TypeSub $\unrhd$ L7 $\unrhd$ L6 $\gg$	$!a \subseteq_2 \mathsf{K}\,'\,b$	;
L09:	Block $\gg$	End	;
L10:	TypeU4Hyp $\unrhd$ L1 $\unrhd$ L2 $\gg$	$\ell\,'\,u_4(a)$	;
L11:	IntroU4' $\unrhd$ L1 $\gg$	$a \subseteq_2 \mathsf{K}\,'\,u_4(a)$	;
L12:	IntroExists $\unrhd$ (Gen $\rhd$ L8) $\unrhd$ L10 $\unrhd$ L11 $\gg$	$\exists b\colon a \subseteq_2 \mathsf{K}\,'\,b$	]*

```
[ The Map proof of ExistsUnion reads
 L01:  ExistsUnion1 $\gg$                          $\exists b\colon \mathsf{T} \subseteq_2 \mathsf{K}\,'\,b$                              ;
 L02:  ExistsUnion2 $\gg$                          $\exists b\colon \mathsf{T} \in_\ell \mathsf{K}\,'\,b$                              ;
 L03:  Logic $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$                   $(\exists b\colon \mathsf{T} \subseteq_2 \mathsf{K}\,'\,b) \wedge$
                                               $\exists b\colon \mathsf{T} \in_\ell \mathsf{K}\,'\,b$                              ;
 L04:  Block $\gg$                                 Begin                                    ;
 L05:  Hypothesis $\gg$                              $\neg a$                                   ;
 L06:  Hypothesis $\gg$                              $\ell\,'\,a$                                   ;
 L07:  Hypothesis $\gg$                              $\forall c\colon (\exists b\colon a\,'\,c \subseteq_2 \mathsf{K}\,'\,b) \wedge$
                                                 $(\exists b\colon a\,'\,c \in_\ell \mathsf{K}\,'\,b)$                    ;
 L08:  Hypothesis $\gg$                              $\mathsf{E}c\colon \ell\,'\,c \wedge ((\exists b\colon c \subseteq_2 \mathsf{K}\,'\,b) \wedge$
                                                 $(\exists b\colon c \in_\ell \mathsf{K}\,'\,b)) \wedge a \in_l c$        ;
 L09:  Block $\gg$                                     Begin                                ;
 L10:  Hypothesis $\gg$                                  $\ell\,'\,c$                               ;
 L11:  ElimAll $\trianglerighteq$ L7 $\trianglerighteq$ L10 $\gg$                       $(\exists b\colon a\,'\,c \subseteq_2 \mathsf{K}\,'\,b) \wedge$
                                                     $(\exists b\colon a\,'\,c \in_\ell \mathsf{K}\,'\,b)$                ;
 L12:  Logic $\trianglerighteq$ L11 $\gg$                              $\exists b\colon a\,'\,c \in_\ell \mathsf{K}\,'\,b$                  ;
 L13:  Block $\gg$                                     End                                  ;
 L14:  ExistsUnion3$\trianglerighteq$L6$\trianglerighteq$(Gen$\triangleright$L12)$\gg$            $\exists b\colon a \subseteq_2 \mathsf{K}\,'\,b$                         ;
 L15:  Block $\gg$                                     Begin                                ;
 L16:  Hypothesis $\gg$                                  $\ell\,'\,c \wedge ((\exists b\colon c \subseteq_2 \mathsf{K}\,'\,b) \wedge$
                                                     $(\exists b\colon c \in_\ell \mathsf{K}\,'\,b)) \wedge a \in_l c$    ;
 L17:  Logic $\trianglerighteq$ L16 $\gg$                                $\ell\,'\,c$                               ;
 L18:  Logic $\trianglerighteq$ L16 $\gg$                                $\exists b\colon c \subseteq_2 \mathsf{K}\,'\,b$                     ;
 L19:  Logic $\trianglerighteq$ L16 $\gg$                                $a \in_l c$                                ;
 L20:  SubtypeLL1 $\trianglerighteq$ L6 $\gg$                           $\ell_1 a$                                 ;
 L21:  Logic $\trianglerighteq$ L17 $\trianglerighteq$ L19 $\trianglerighteq$ L20 $\gg$                  $a \in_\ell c$                              ;
 L22:  Block $\gg$                                         Begin                            ;
 L23:  Hypothesis $\gg$                                      $\ell\,'\,b$                           ;
 L24:  TypeKa $\trianglerighteq$ L23 $\gg$                                 $\ell\,'(\mathsf{K}\,'\,b)$                       ;
 L26:  TypeInEll $\trianglerighteq$ L20 $\trianglerighteq$ L24 $\gg$                      $!a \in_\ell \mathsf{K}\,'\,b$                       ;
 L27:  Block $\gg$                                         End                              ;
 L28:  Define $\gg$                                        $b \equiv \varepsilon b\colon c \subseteq_2 \mathsf{K}\,'\,b$                ;
 L29:  ElimExists $\trianglerighteq$ L18 $\gg$                           $c \subseteq_2 \mathsf{K}\,'\,b$                          ;
 L30:  ElimExistsEll $\trianglerighteq$ L18 $\gg$                        $\ell\,'\,b$                               ;
 L31:  TypeKa $\trianglerighteq$ L30 $\gg$                               $\ell\,'(\mathsf{K}\,'\,b)$                           ;
 L32:  SubInEll $\trianglerighteq$ L31 $\trianglerighteq$ L29 $\trianglerighteq$ L21 $\gg$               $a \in_\ell \mathsf{K}\,'\,b$                           ;
 L33:  IntroExists $\trianglerighteq$
       (Gen $\triangleright$ L26) $\trianglerighteq$ L30 $\trianglerighteq$ L32 $\gg$                 $\exists b\colon a \in_\ell \mathsf{K}\,'\,b$                    ;
 L34:  Block $\gg$                                     End                                  ;
 L35:  ElimPure $\triangleright$ L8 $\triangleright$ L33 $\gg$                      $\exists b\colon a \in_\ell \mathsf{K}\,'\,b$                        ;
 L36:  Logic $\trianglerighteq$ L14 $\trianglerighteq$ L35 $\gg$                      $(\exists b\colon a \subseteq_2 \mathsf{K}\,'\,b) \wedge$
                                                 $(\exists b\colon a \in_\ell \mathsf{K}\,'\,b)$                      ;
 L37:  Block $\gg$                                 End                                      ;
 L38:  Transfinite'' $\triangleright$ L3 $\triangleright$ L36 $\gg$                 $\forall a\colon (\exists b\colon a \subseteq_2 \mathsf{K}\,'\,b) \wedge$
                                               $(\exists b\colon a \in_\ell \mathsf{K}\,'\,b)$                        ]*
```

```
[ The Map proof of UnionHyp reads
 L1:  Hypothesis $\gg$                      $\ell ' a$                                        ;
 L2:  ExistsUnion $\gg$                     $\forall a\colon (\exists b\colon a \subseteq_2 \mathsf{K} ' b) \wedge$
                                            $(\exists b\colon a \in_\ell \mathsf{K} ' b)$                          ;
 L3:  Block $\gg$                           Begin                                             ;
 L4:  Hypothesis $\gg$                        $\ell ' c$                                      ;
 L5:  TypeApply $\unrhd$ L1 $\unrhd$ L4 $\gg$             $\ell '(a ' c)$                                 ;
 L6:  ElimAll $\unrhd$ L2 $\unrhd$ L5 $\gg$               $(\exists b\colon a ' c \subseteq_2 \mathsf{K} ' b) \wedge$
                                              $(\exists b\colon a ' c \in_\ell \mathsf{K} ' b)$                ;
 L7:  Logic $\unrhd$ L6 $\gg$                       $\exists b\colon a ' c \in_\ell \mathsf{K} ' b$                   ;
 L8:  Block $\gg$                           End                                               ;
 L9:  Gen $\rhd$ L7 $\gg$                         $\forall c\colon \exists b\colon a ' c \in_\ell \mathsf{K} ' b$                      ]*

[ The Map proof of TypeU4 reads
 L1:  Hypothesis $\gg$                      $\ell ' a$                                  ;
 L2:  UnionHyp $\unrhd$ L1 $\gg$                  $\forall c\colon \exists b\colon a ' c \in_\ell \mathsf{K} ' b$                ;
 L3:  TypeU4Hyp $\unrhd$ L1 $\unrhd$ L2 $\gg$             $\ell ' u_4(a)$                             ]*

[ The Map proof of IntroU3 reads
 L1:  Hypothesis $\gg$                           $\ell ' a$                                  ;
 L2:  Hypothesis $\gg$                           $\ell ' c$                                  ;
 L3:  UnionHyp $\unrhd$ L1 $\gg$                       $\forall c\colon \exists b\colon a ' c \in_\ell \mathsf{K} ' b$                ;
 L4:  IntroU3Hyp $\unrhd$ L1 $\unrhd$ L2 $\unrhd$ L3 $\gg$         $a ' c \in_\ell u_3 \circ a$                      ]*

[ The Map proof of ElimU4 reads
 L01:  Hypothesis $\gg$                                   $\ell ' x$                                       ;
 L02:  Hypothesis $\gg$                                   $\ell ' y$                                       ;
 L03:  Hypothesis $\gg$                                   $x \in_2 u_4(y)$                                 ;
 L04:  Define $\gg$                                       $s \equiv s_2(x, u_4(y))$                          ;
 L05:  Define $\gg$                                       $t \equiv t_2(x, u_4(y))$                          ;
 L06:  ElimInTwoS $\unrhd$ L3 $\gg$                             $\ell ' s$                                       ;
 L07:  ElimInTwoT $\unrhd$ L3 $\gg$                             $\ell ' t$                                       ;
 L08:  ElimInTwo $\unrhd$ L3 $\gg$                              $x \sim u_4(y) ' s ' t$                            ;
 L09:  Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L8 $\gg$                           $x \sim y '(s ' \mathsf{T}) '(s ' \mathsf{F}) ' t$                   ;
 L10:  TypeT $\gg$                                        $\ell ' \mathsf{T}$                                       ;
 L11:  TypeApply $\unrhd$ L6 $\unrhd$ L10 $\gg$                      $\ell '(s ' \mathsf{T})$                                  ;
 L12:  TypeF $\gg$                                        $\ell ' \mathsf{F}$                                       ;
 L13:  TypeApply $\unrhd$ L6 $\unrhd$ L12 $\gg$                      $\ell '(s ' \mathsf{F})$                                  ;
 L14:  TypeApply $\unrhd$ L2 $\unrhd$ L11 $\gg$                      $\ell '(y '(s ' \mathsf{T}))$                              ;
 L15:  SubtypeLL2 $\unrhd$ L14 $\gg$                            $\ell_2 y '(s ' \mathsf{T})$                                ;
 L16:  IntroInTwo $\unrhd$ L15 $\unrhd$ L13 $\unrhd$ L7 $\unrhd$ L9 $\gg$        $x \in_2 y '(s ' \mathsf{T})$                             ;
 L17:  Block $\gg$                                        Begin                                          ;
 L18:  Hypothesis $\gg$                                     $\ell ' z$                                     ;
 L19:  TypeApply $\unrhd$ L2 $\unrhd$ L18 $\gg$                        $\ell '(y ' z)$                                ;
 L20:  SubtypeLL2 $\unrhd$ L19 $\gg$                              $\ell_2 y ' z$                                 ;
 L21:  TypeInTwo $\unrhd$ L1 $\unrhd$ L20 $\gg$                        $!x \in_2 y ' z$                               ;
 L22:  Block $\gg$                                        End                                            ;
 L23:  IntroExists $\rhd$ L21 $\rhd$ L11 $\unrhd$ L16 $\gg$              $\exists z\colon x \in_2 y ' z$                           ]*
```

[The Map proof of IntroU4 reads

L01:	Hypothesis $\gg$	$\ell\text{'}y$	;
L02:	Hypothesis $\gg$	$\ell\text{'}z$	;
L03:	Hypothesis $\gg$	$x \in_2 y\text{'}z$	;
L04:	Define $\gg$	$s \equiv s_2(x, y\text{'}z)$	;
L05:	Define $\gg$	$t \equiv t_2(x, y\text{'}z)$	;
L06:	ElimInTwoS $\trianglerighteq$ L3 $\gg$	$\ell\text{'}s$	;
L07:	ElimInTwoT $\trianglerighteq$ L3 $\gg$	$\ell\text{'}t$	;
L08:	ElimInTwo $\trianglerighteq$ L3 $\gg$	$x \sim y\text{'}z\text{'}s\text{'}t$	;
L09:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L8 $\gg$	$x \sim u_4(y)\text{'}(z :: s)\text{'}t$	;
L10:	TypePair $\trianglerighteq$ L2 $\trianglerighteq$ L6 $\gg$	$\ell\text{'}(z :: s)$	;
L11:	TypeU4 $\trianglerighteq$ L1 $\gg$	$\ell\text{'}u_4(y)$	;
L12:	SubtypeLL2 $\trianglerighteq$ L11 $\gg$	$\ell_2 u_4(y)$	;
L13:	IntroInTwo $\trianglerighteq$ L12 $\trianglerighteq$ L10 $\trianglerighteq$ L7 $\trianglerighteq$ L9 $\gg$	$x \in_2 u_4(y)$	]*

[The Map proof of IntroSubU4 reads

L1:	Hypothesis $\gg$	$\ell\text{'}x$	;
L2:	Hypothesis $\gg$	$\ell\text{'}y$	;
L3:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\text{'}(x\text{'}y)$	;
L4:	SubtypeLL2 $\trianglerighteq$ L3 $\gg$	$\ell_2 x\text{'}y$	;
L5:	IntroU4 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$u \in_2 x\text{'}y \to u \in_2 u_4(x)$	;
L6:	IntroSub $\triangleright$ L4 $\triangleright$ L5 $\gg$	$x\text{'}y \subseteq_2 u_4(x)$	]*

[The Map proof of ElimU4Sub reads

L01:	Hypothesis $\gg$	$\ell\text{'}x$	;
L02:	Hypothesis $\gg$	$\ell\text{'}y$	;
L03:	Hypothesis $\gg$	$\ell\text{'}z$	;
L04:	Hypothesis $\gg$	$u_4(x) \subseteq_2 y$	;
L05:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$\ell\text{'}(x\text{'}z)$	;
L06:	SubtypeLL2 $\trianglerighteq$ L5 $\gg$	$\ell_2 x\text{'}z$	;
L07:	Block $\gg$	Begin	;
L08:	Hypothesis $\gg$	$u \in_2 x\text{'}z$	;
L09:	IntroU4 $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\trianglerighteq$ L8 $\gg$	$u \in_2 u_4(x)$	;
L10:	ElimSub $\trianglerighteq$ L4 $\trianglerighteq$ L9 $\gg$	$u \in_2 y$	;
L11:	Block $\gg$	End	;
L12:	IntroSub $\triangleright$ L6 $\triangleright$ L10 $\gg$	$x\text{'}z \subseteq_2 y$	]*

A.47 Transitive maps

[Map lemma

TypeTr'$_{196}$:	$\ell p, x\colon !(x \in_\ell p \wedge \forall y\colon x\text{'}y \in_2 p)$	;
IntroTr$_{197}$:	$\langle x, y{\in}\mathcal{V}; p{\in}\mathcal{T}\rangle \mathrm{notfree}(y; p) \Vdash \ell\text{'}p \vdash$ $\ell x, y\colon x \in_2 p \to x \in_\ell p \vdash$ $\ell x, y\colon x \in_2 p \to x\text{'}y \in_2 p \vdash \mathrm{Tr}(p)$	;
IntroTrT$_{198}$:	$\mathrm{Tr}(\mathsf{T})$	]*

[The Map proof of TypeTr' reads

L01:	Hypothesis $\gg$	$\ell\,'p$	;
L02:	Hypothesis $\gg$	$\ell\,'x$	;
L03:	SubtypeLL1 $\trianglerighteq$ L2 $\gg$	$!_1 x$	;
L04:	TypeInEll $\trianglerighteq$ L3 $\trianglerighteq$ L1 $\gg$	$!x \in_\ell p$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	$\quad \ell\,'y$	;
L07:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L6 $\gg$	$\quad \ell\,'(x\,'y)$	;
L08:	SubtypeLL2 $\trianglerighteq$ L1 $\gg$	$\quad \ell_2 p$	;
L09:	TypeInTwo $\trianglerighteq$ L7 $\trianglerighteq$ L8 $\gg$	$\quad !x\,'y \in_2 p$	;
L10:	Block $\gg$	End	;
L11:	TypeAll $\trianglerighteq$ (Gen $\triangleright$ L9) $\gg$	$!\forall y\colon x\,'y \in_2 p$	;
L12:	Logic $\trianglerighteq$ L4 $\trianglerighteq$ L11 $\gg$	$!(x \in_\ell p \wedge \forall y\colon x\,'y \in_2 p)$	]*

[The Map proof of TypeTr reads

L1:	Hypothesis $\gg$	$\ell\,'p$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad \ell\,'x$	;
L4:	SubtypeLL2 $\trianglerighteq$ L1 $\gg$	$\quad \ell_2 p$	;
L5:	TypeInTwo $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$\quad !x \in_2 p$	;
L6:	TypeTr' $\trianglerighteq$ L1 $\trianglerighteq$ L5 $\gg$	$\quad !(x \in_\ell p \wedge \forall y\colon x\,'y \in_2 p)$	;
L7:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$\quad !(x \in_2 p \Rightarrow x \in_\ell p \wedge \forall y\colon x\,'y \in_2 p)$	;
L9:	Block $\gg$	End	;
L9:	TypeAll $\trianglerighteq$ (Gen $\triangleright$ L7) $\gg$	$!\mathrm{Tr}(p)$	]*

[The Map proof of StrictTr reads

L01:	Hypothesis $\gg$	$!\mathrm{Tr}(p)$	;
L02:	TypeT $\gg$	$\ell\,'\mathsf{T}$	;
L03:	ElimAll $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\mathsf{T} \in_2 p \Rightarrow \mathsf{T} \in_\ell p \wedge \forall y\colon \mathsf{T}\,'y \in_2 p$	;
L04:	Logic $\trianglerighteq$ L3 $\gg$	$!\mathsf{T} \in_\ell p$	;
L05:	StrictInEllP $\trianglerighteq$ L4 $\gg$	$\ell\,'p$	]*

[The Map proof of ElimTrInEll reads

L1:	Hypothesis $\gg$	$\mathrm{Tr}(p)$	;
L2:	Hypothesis $\gg$	$x \in_2 p$	;
L3:	StrictInTwoX $\trianglerighteq$ L2 $\gg$	$\ell\,'x$	;
L4:	ElimAll $\triangleright$ L1 $\trianglerighteq$ L3 $\gg$	$x \in_2 p \Rightarrow x \in_\ell p \wedge \forall y\colon x\,'y \in_2 p$	;
L5:	Logic $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\gg$	$x \in_\ell p$	]*

[The Map proof of ElimTrInTwo reads

L1:	Hypothesis $\gg$	$\ell\,'y$	;
L2:	Hypothesis $\gg$	$\mathrm{Tr}(p)$	;
L3:	$\gg$	$x \in_2 p$	;
L4:	StrictInTwoX $\trianglerighteq$ L3 $\gg$	$\ell\,'x$	;
L5:	ElimAll $\triangleright$ L2 $\trianglerighteq$ L4 $\gg$	$x \in_2 p \Rightarrow x \in_\ell p \wedge \forall y\colon x\,'y \in_2 p$	;
L6:	Logic $\triangleright$ L3 $\triangleright$ L5 $\gg$	$\forall y\colon x\,'y \in_2 p$	;
L7:	ElimAll $\trianglerighteq$ L6 $\trianglerighteq$ L1 $\gg$	$x\,'y \in_2 p$	]*

[The Map proof of ElimTrHead reads

L1:	Hypothesis $\gg$	$\mathrm{Tr}(p)$	;
L2:	Hypothesis $\gg$	$x :: y \in_2 p$	;
L3:	TypeT $\gg$	$\ell\,\text{'}\,\mathsf{T}$	;
L4:	ElimTrInTwo $\trianglerighteq$ L3 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$(x :: y)\,\text{'}\,\mathsf{T} \in_2 p$	;
L5:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L4 $\gg$	$x \in_2 p$	]*

[The Map proof of ElimTrTail reads

L1:	Hypothesis $\gg$	$\mathrm{Tr}(p)$	;
L2:	Hypothesis $\gg$	$x :: y \in_2 p$	;
L3:	TypeF $\gg$	$\ell\,\text{'}\,\mathsf{F}$	;
L4:	ElimTrInTwo $\trianglerighteq$ L3 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$(x :: y)\,\text{'}\,\mathsf{F} \in_2 p$	;
L5:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L4 $\gg$	$y \in_2 p$	]*

[The Map proof of ElimTrSub reads

L01:	Hypothesis $\gg$	$\mathrm{Tr}(p)$	;
L02:	Hypothesis $\gg$	$x \in_2 p$	;
L03:	StrictInTwoX $\trianglerighteq$ L2 $\gg$	$\ell\,\text{'}\,x$	;
L04:	StrictInTwoY $\trianglerighteq$ L2 $\gg$	$\ell_2 p$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	$\quad y \in_2 x$	;
L07:	Define $\gg$	$\quad s \equiv s_2(y, x)$	;
L08:	Define $\gg$	$\quad t \equiv t_2(y, x)$	;
L09:	ElimInTwoS $\trianglerighteq$ L6 $\gg$	$\quad \ell\,\text{'}\,s$	;
L10:	ElimInTwoT $\trianglerighteq$ L6 $\gg$	$\quad \ell\,\text{'}\,t$	;
L11:	ElimInTwo $\trianglerighteq$ L6 $\gg$	$\quad y \sim x\,\text{'}\,s\,\text{'}\,t$	;
L12:	ElimTrInTwo $\trianglerighteq$ L9 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\quad x\,\text{'}\,s \in_2 p$	;
L13:	ElimTrInTwo $\trianglerighteq$ L10 $\trianglerighteq$ L1 $\trianglerighteq$ L12 $\gg$	$\quad x\,\text{'}\,s\,\text{'}\,t \in_2 p$	;
L14:	Block $\gg$	$\quad$ Begin	;
L15:	Hypothesis $\gg$	$\quad\quad \ell\,\text{'}\,w$	;
L16:	TypeInTwo $\trianglerighteq$ L15 $\trianglerighteq$ L4 $\gg$	$\quad\quad !w \in_2 p$	;
L17:	Block $\gg$	$\quad$ End	;
L18:	ExtBool" $\trianglerighteq$ (Gen $\triangleright$ L16) $\trianglerighteq$ L11 $\trianglerighteq$ L13 $\gg$	$\quad y \in_2 p$	;
L19:	Block $\gg$	End	;
L20:	IntroSub $\triangleright$ (SubtypeLL2 $\trianglerighteq$ L3) $\triangleright$ L18 $\gg$	$x \subseteq_2 p$	]*

[The Map proof of IntroTr reads

L01:	Premise $\gg$	$\ell\,'\,p$	;
L02:	Premise $\gg$	$\ell x, y{:}\, x \in_2 p \to x \in_\ell p$	;
L03:	Premise $\gg$	$\ell x, y{:}\, x \in_2 p \to x\,'\,y \in_2 p$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\quad \ell\,'\,x$	;
L06:	SubtypeLL2 $\trianglerighteq$ L1 $\gg$	$\quad \ell_2 p$	;
L07:	TypeInTwo $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$\quad !x \in_2 p$	;
L08:	TypeTr' $\trianglerighteq$ L1 $\trianglerighteq$ L5 $\gg$	$\quad !(x \in_\ell p \wedge \forall y{:}\, x\,'\,y \in_2 p)$	;
L09:	Block $\gg$	$\quad$ Begin	;
L10:	Hypothesis $\gg$	$\quad\quad x \in_2 p$	;
L11:	L2 $\trianglerighteq$ L5 $\trianglerighteq$ L5 $\trianglerighteq$ L10 $\gg$	$\quad\quad x \in_\ell p$	;
L12:	Block $\gg$	$\quad\quad$ Begin	;
L13:	Hypothesis $\gg$	$\quad\quad\quad \ell\,'\,y$	;
L14:	L3 $\trianglerighteq$ L5 $\trianglerighteq$ L13 $\trianglerighteq$ L10 $\gg$	$\quad\quad\quad x\,'\,y \in_2 p$	;
L15:	Block $\gg$	$\quad\quad$ End	;
L16:	Gen $\triangleright$ L14 $\gg$	$\quad\quad \forall y{:}\, x\,'\,y \in_2 p$	;
L17:	Logic $\trianglerighteq$ L11 $\trianglerighteq$ L16 $\gg$	$\quad\quad x \in_\ell p \wedge \forall y{:}\, x\,'\,y \in_2 p$	;
L18:	Block $\gg$	$\quad$ End	;
L19:	CDeduction$\triangleright$L7$\triangleright$L8$\triangleright$L17$\gg$	$\quad x \in_2 p \Rightarrow x \in_\ell p \wedge \forall y{:}\, x\,'\,y \in_2 p$	;
L20:	Block $\gg$	End	;
L21:	Gen $\triangleright$ L19 $\gg$	$\mathrm{Tr}(p)$	]*

[The Map proof of IntroTrT reads

L01:	TypeT $\gg$	$\ell\,'\,\mathsf{T}$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\quad \ell\,'\,x$	;
L04:	Hypothesis $\gg$	$\quad \ell\,'\,y$	;
L05:	Hypothesis $\gg$	$\quad x \in_2 \mathsf{T}$	;
L06:	ElimInTwo $\trianglerighteq$ L5 $\gg$	$\quad x \sim \mathsf{T}\,'\,s_2(x, \mathsf{T})\,'\,t_2(x, \mathsf{T})$	;
L07:	Logic $\trianglerighteq$ L6 $\gg$	$\quad x$	;
L08:	IntroInEllT $\trianglerighteq$ L1 $\gg$	$\quad \mathsf{T} \in_\ell \mathsf{T}$	;
L09:	Replace' $\triangleright$ L7 $\triangleright$ L8 $\gg$	$\quad x \in_\ell \mathsf{T}$	;
L10:	Logic $\trianglerighteq$ L7 $\gg$	$\quad x\,'\,y$	;
L11:	IntroInTwoT $\gg$	$\quad \mathsf{T} \in_2 \mathsf{T}$	;
L12:	Replace' $\trianglerighteq$ L10 $\trianglerighteq$ L11 $\gg$	$\quad x\,'\,y \in_2 \mathsf{T}$	;
L13:	Block $\gg$	End	;
L14:	IntroTr $\triangleright$ L1 $\triangleright$ L9 $\triangleright$ L12 $\gg$	$\mathrm{Tr}(\mathsf{T})$	]*

A.48 Fifth union operator

[$u_5\,'\,z$] denotes a transitive closure of [z] if such a one exists (the existence is proved later).

$$[\ \boxed{u_5} \quad \doteq \quad \lambda z.\varepsilon a{:}\, z \in_2 a \wedge \mathrm{Tr}(a) \quad]^*$$

[Map lemma

$\mathrm{TypeU5}_{199}$:	$\ell_1 u_5$	;
$\mathrm{TypeU5Apply}_{199}$:	$\ell x{:}\, \ell\,\text{'}(u_5\,\text{'}\,x)$	;
$\mathrm{TypeU5Compose}_{199}$:	$\ell x{:}\, \ell\,\text{'}(u_5 \circ x)$	;
$\mathrm{IntroTrU5}_{199}$:	$\exists a{:}\, y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow \mathrm{Tr}(u_5\,\text{'}\,y)$	;
$\mathrm{IntroInU5}_{199}$:	$\exists a{:}\, y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow y \in_2 u_5\,\text{'}\,y$	]*

[The Map proof of TypeU5 reads

L01:	Block $\gg$	Begin	;
L02:	Hypothesis $\gg$	$\ell\,\text{'}\,z$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\ell\,\text{'}\,a$	;
L05:	SubtypeLL2 $\trianglerighteq$ L4 $\gg$	$\ell_2 a$	;
L06:	TypeInTwo $\trianglerighteq$ L2 $\trianglerighteq$ L5 $\gg$	$!z \in_2 a$	;
L07:	TypeTr $\trianglerighteq$ L4 $\gg$	$!\mathrm{Tr}(a)$	;
L08:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$!(z \in_2 a \wedge \mathrm{Tr}(a))$	;
L09:	Block $\gg$	End	;
L10:	TypeChoice $\trianglerighteq$ (Gen $\triangleright$ L8) $\gg$	$\ell\,\text{'}\,\varepsilon a{:}\, z \in_2 a \wedge \mathrm{Tr}(a)$	;
L11:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L10 $\gg$	$\ell\,\text{'}(u_5\,\text{'}\,z)$	;
L12:	Block $\gg$	End	;
L13:	Gen $\triangleright$ L11 $\gg$	$\ell_1 u_5$	]*

[The Map proof of TypeU5Apply reads

L1:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L2:	TypeU5 $\gg$	$\ell_1 u_5$	;
L3:	ElimAll $\trianglerighteq$ L2 $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}(u_5\,\text{'}\,x)$	]*

[The Map proof of TypeU5Compose reads

L1:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L2:	TypeU5 $\gg$	$\ell_1 u_5$	;
L3:	TypeCompose $\trianglerighteq$ L2 $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}(u_5 \circ x)$	]*

[The Map proof of IntroTrU5 reads

L1:	Hypothesis $\gg$	$\exists a{:}\, y \in_2 a \wedge \mathrm{Tr}(a)$	;
L2:	Define $\gg$	$a \equiv \varepsilon a{:}\, y \in_2 a \wedge \mathrm{Tr}(a)$	;
L3:	ElimExists $\trianglerighteq$ L1 $\gg$	$y \in_2 a \wedge \mathrm{Tr}(a)$	;
L4:	Logic $\trianglerighteq$ L3 $\gg$	$\mathrm{Tr}(a)$	;
L5:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L4 $\gg$	$\mathrm{Tr}(u_5\,\text{'}\,y)$	]*

[The Map proof of IntroInU5 reads

L1:	Hypothesis $\gg$	$\exists a{:}\, y \in_2 a \wedge \mathrm{Tr}(a)$	;
L2:	Define $\gg$	$a \equiv \varepsilon a{:}\, y \in_2 a \wedge \mathrm{Tr}(a)$	;
L3:	ElimExists $\trianglerighteq$ L1 $\gg$	$y \in_2 a \wedge \mathrm{Tr}(a)$	;
L4:	Logic $\trianglerighteq$ L3 $\gg$	$y \in_2 a$	;
L5:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L4 $\gg$	$y \in_2 u_5\,\text{'}\,y$	]*

A.49 Sixth union operator

The existence of transitive closures will be by transfinite induction. The sixth union operator allows to combine transitive closures into new transitive closures in a way that will be needed in the induction step of that proof.

$$[\ \boxed{u_6(x,y)}\ \doteq\ u_2(u_2(\mathsf{K}\,'(\mathsf{K}\,'x)),u_5\,'y),u_4(u_5\circ x))\quad]^*$$

[Map lemma

Alternate$_{200}$:	$\langle a,b,c{\in}\mathcal{T}\rangle a\to c\vdash b\to c\vdash a\vee b\to c$	;
Alternate3$_{200}$:	$\langle a,b,c,d{\in}\mathcal{T}\rangle a\to d\vdash b\to d\vdash c\to d\vdash a\vee b\vee c\to d$	;
AlternateA$_{201}$:	$\langle a,b,c{\in}\mathcal{T}\rangle a\to \underline{a}\vdash a\vee b\to \underline{a}\vee b$	;
AlternateB$_{201}$:	$\langle a,b,c{\in}\mathcal{T}\rangle b\to \underline{b}\vdash a\vee b\to a\vee \underline{b}$	;
TypeKK$_{201}$:	$\ell x{:}\,\ell\,'(\mathsf{K}\,'(\mathsf{K}\,'x))$	;
ElimKK$_{201}$:	$x\in_2\mathsf{K}\,'(\mathsf{K}\,'y)\to x\sim y$	;
IntroKK$_{201}$:	$\ell x{:}\,x\in_2\mathsf{K}\,'(\mathsf{K}\,'x)$	;
TypeU6'$_{201}$:	$\ell x,y{:}\,\ell\,'u_2(\mathsf{K}\,'(\mathsf{K}\,'x)),u_5\,'y)$	;
ElimU6'$_{201}$:	$\ell x,y{:}\,z\in_2 u_2(\mathsf{K}\,'(\mathsf{K}\,'x)),u_5\,'y)\to z\sim x\vee z\in_2 u_5\,'y$	;
IntroU6'X$_{202}$:	$\ell x,y{:}\,x\in_2 u_2(\mathsf{K}\,'(\mathsf{K}\,'x)),u_5\,'y)$	;
IntroU6'Y$_{202}$:	$\ell x,y{:}\,u_5\,'y\subseteq_2 u_2(\mathsf{K}\,'(\mathsf{K}\,'x)),u_5\,'y)$	;
TypeU6$_{202}$:	$\ell x,y{:}\,\ell\,'u_6(x,y)$	;
ElimU6$_{202}$:	$\ell x,y{:}\,z\in_2 u_6(x,y)\to z{\sim}x\vee z{\in_2}u_5{'}y\vee\exists u{:}\,z{\in_2}u_5{'}(x{'}u)$	;
IntroU6X$_{203}$:	$\ell x,y{:}\,x\in_2 u_6(x,y)$	;
IntroU6Y$_{203}$:	$\ell x,y{:}\,u_5\,'y\subseteq_2 u_6(x,y)$	;
IntroU6XU$_{203}$:	$\ell x,y,u{:}\,u_5\,'(x\,'u)\subseteq_2 u_6(x,y)$	$]^*$

[The Map proof of Alternate reads

L01:	Premise $\gg$	$a\to c$	;
L02:	Premise $\gg$	$b\to c$	;
L03:	Hypothesis $\gg$	$a\vee b$	;
L04:	Logic $\trianglerighteq$ L3 $\gg$	$!a$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	a	;
L08:	L1 $\trianglerighteq$ L6 $\gg$	c	;
L09:	Block $\gg$	End	;
L10:	Block $\gg$	Begin	;
L11:	Hypothesis $\gg$	$\neg a$	;
L12:	Logic $\trianglerighteq$ L3 $\trianglerighteq$ L11 $\gg$	b	;
L13:	L2 $\trianglerighteq$ L12 $\gg$	c	;
L14:	Block $\gg$	End	;
L15:	TND $\rhd$ L4 $\rhd$ L8 $\rhd$ L13 $\gg$	c	$]^*$

[The Map proof of Alternate3 reads

L1:	Premise $\gg$	$a\to d$	;
L2:	Premise $\gg$	$b\to d$	;
L3:	Premise $\gg$	$c\to d$	;
L4:	Alternate $\rhd$ L1 $\rhd$ L2 $\gg$	$a\vee b\to d$	;
L5:	Alternate $\rhd$ L4 $\rhd$ L3 $\gg$	$a\vee b\vee c\to d$	$]^*$

[The Map proof of AlternateA reads
L1: Premise $\gg$ $a \to \underline{a}$;
L2: Hypothesis $\gg$ $a \vee b$;
L3: Logic $\gg$ $b \to b$;
L4: Alternate $\triangleright$ L1 $\triangleright$ L3 $\trianglerighteq$ L2 $\gg$ $\underline{a} \vee b$]*

[The Map proof of AlternateB reads
L1: Premise $\gg$ $b \to \underline{b}$;
L2: Hypothesis $\gg$ $a \vee b$;
L3: Logic $\gg$ $a \to a$;
L4: Alternate $\triangleright$ L3 $\triangleright$ L1 $\trianglerighteq$ L2$a \vee \underline{b}$]*

[The Map proof of TypeKK reads
L1: Hypothesis $\gg$ $\ell\text{'}x$;
L2: TypeKa $\trianglerighteq$ L1 $\gg$ $\ell\text{'}(\mathsf{K}\text{'}x)$;
L3: TypeKa $\trianglerighteq$ L2 $\gg$ $\ell\text{'}(\mathsf{K}\text{'}(\mathsf{K}\text{'}x))$]*

[The Map proof of ElimKK reads
L1: Hypothesis $\gg$ $x \in_2 \mathsf{K}\text{'}(\mathsf{K}\text{'}y)$;
L2: ElimInTwo $\trianglerighteq$ L1 $\gg$ $x \sim \mathsf{K}\text{'}(\mathsf{K}\text{'}y)\text{'}s_2(x, \mathsf{K}\text{'}(\mathsf{K}\text{'}y))\text{'}t_2(x, \mathsf{K}\text{'}(\mathsf{K}\text{'}y))$;
L3: Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L2 $\gg$ $x \sim y$]*

[The Map proof of IntroKK reads
L1: Hypothesis $\gg$ $\ell\text{'}x$;
L2: RefEquivK $\trianglerighteq$ L1 $\gg$ $x \sim x$;
L3: Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L2 $\gg$ $x \sim \mathsf{K}\text{'}(\mathsf{K}\text{'}x)\text{'}x\text{'}x$;
L4: TypeKK $\trianglerighteq$ L1 $\gg$ $\ell\text{'}(\mathsf{K}\text{'}(\mathsf{K}\text{'}x))$;
L5: SubtypeLL2 $\trianglerighteq$ L4 $\gg$ $\ell_2\mathsf{K}\text{'}(\mathsf{K}\text{'}x)$;
L6: IntroInTwo $\trianglerighteq$ L5 $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$ $x \in_2 \mathsf{K}\text{'}(\mathsf{K}\text{'}x)$]*

[The Map proof of TypeU6' reads
L1: Hypothesis $\gg$ $\ell\text{'}x$;
L2: Hypothesis $\gg$ $\ell\text{'}y$;
L3: TypeKK $\trianglerighteq$ L1 $\gg$ $\ell\text{'}(\mathsf{K}\text{'}(\mathsf{K}\text{'}x))$;
L4: TypeU5Apply $\trianglerighteq$ L2 $\gg$ $\ell\text{'}(u_5\text{'}y)$;
L5: TypeU2 $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$ $\ell\text{'}u_2(\mathsf{K}\text{'}(\mathsf{K}\text{'}x)), u_5\text{'}y)$]*

[The Map proof of ElimU6' reads
L1: Hypothesis $\gg$ $\ell\text{'}x$;
L2: Hypothesis $\gg$ $\ell\text{'}y$;
L3: Hypothesis $\gg$ $z \in_2 u_2(\mathsf{K}\text{'}(\mathsf{K}\text{'}x)), u_5\text{'}y)$;
L4: TypeKK $\trianglerighteq$ L1 $\gg$ $\ell\text{'}(\mathsf{K}\text{'}(\mathsf{K}\text{'}x))$;
L5: TypeU5Apply $\trianglerighteq$ L2 $\gg$ $\ell\text{'}(u_5\text{'}y)$;
L6: ElimU2 $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L3 $\gg$ $z \in_2 \mathsf{K}\text{'}(\mathsf{K}\text{'}x)) \vee z \in_2 u_5\text{'}y$;
L7: AlternateA $\triangleright$ ElimKK $\trianglerighteq$ L6 $\gg$ $z \sim x \vee z \in_2 u_5\text{'}y$]*

[The Map proof of IntroU6'X reads

$\mathsf{L1}$:	Hypothesis $\gg$	$\ell\text{'}x$	;
$\mathsf{L2}$:	Hypothesis $\gg$	$\ell\text{'}y$	;
$\mathsf{L3}$:	TypeKK $\trianglerighteq \mathsf{L1} \gg$	$\ell\text{'}(\mathsf{K}\text{'}(\mathsf{K}\text{'}x))$	;
$\mathsf{L4}$:	TypeU5Apply $\trianglerighteq \mathsf{L2} \gg$	$\ell\text{'}(u_5\text{'}y)$	;
$\mathsf{L5}$:	IntroU2a $\trianglerighteq \mathsf{L3} \trianglerighteq \mathsf{L4} \gg$	$\mathsf{K}\text{'}(\mathsf{K}\text{'}x) \subseteq_2 u_2(\mathsf{K}\text{'}(\mathsf{K}\text{'}x)), u_5\text{'}y)$	;
$\mathsf{L6}$:	IntroKK $\trianglerighteq \mathsf{L1} \gg$	$x \in_2 \mathsf{K}\text{'}(\mathsf{K}\text{'}x)$	;
$\mathsf{L7}$:	ElimSub $\trianglerighteq \mathsf{L5} \trianglerighteq \mathsf{L6} \gg$	$x \in_2 u_2(\mathsf{K}\text{'}(\mathsf{K}\text{'}x)), u_5\text{'}y)$	]*

[The Map proof of IntroU6'Y reads

$\mathsf{L1}$:	Hypothesis $\gg$	$\ell\text{'}x$	;
$\mathsf{L2}$:	Hypothesis $\gg$	$\ell\text{'}y$	;
$\mathsf{L3}$:	TypeKK $\trianglerighteq \mathsf{L1} \gg$	$\ell\text{'}(\mathsf{K}\text{'}(\mathsf{K}\text{'}x))$	;
$\mathsf{L4}$:	TypeU5Apply $\trianglerighteq \mathsf{L2} \gg$	$\ell\text{'}(u_5\text{'}y)$	;
$\mathsf{L5}$:	IntroU2b $\trianglerighteq \mathsf{L3} \trianglerighteq \mathsf{L4} \gg$	$u_5\text{'}y \subseteq_2 u_2(\mathsf{K}\text{'}(\mathsf{K}\text{'}x)), u_5\text{'}y)$	]*

[The Map proof of TypeU6 reads

$\mathsf{L1}$:	Hypothesis $\gg$	$\ell\text{'}x$	;
$\mathsf{L2}$:	Hypothesis $\gg$	$\ell\text{'}y$	;
$\mathsf{L3}$:	TypeU6' $\trianglerighteq \mathsf{L1} \trianglerighteq \mathsf{L2} \gg$	$\ell\text{'}u_2(\mathsf{K}\text{'}(\mathsf{K}\text{'}x), u_5\text{'}y)$	;
$\mathsf{L4}$:	TypeCompose $\trianglerighteq \mathsf{L5} \trianglerighteq \mathsf{L1} \gg$	$\ell\text{'}(u_5 \circ x)$	;
$\mathsf{L5}$:	TypeU4 $\trianglerighteq \mathsf{L4} \gg$	$\ell\text{'}u_4(u_5 \circ x)$	;
$\mathsf{L6}$:	TypeU2 $\trianglerighteq \mathsf{L3} \trianglerighteq \mathsf{L5} \gg$	$\ell\text{'}u_6(x,y)$	]*

[The Map proof of ElimU6 reads

$\mathsf{L01}$:	Hypothesis $\gg$	$\ell\text{'}x$	;
$\mathsf{L02}$:	Hypothesis $\gg$	$\ell\text{'}y$	;
$\mathsf{L03}$:	Hypothesis $\gg$	$z \in_2 u_6(x,y)$	;
$\mathsf{L04}$:	TypeU6' $\trianglerighteq \mathsf{L1} \trianglerighteq \mathsf{L2} \gg$	$\ell\text{'}u_2(\mathsf{K}\text{'}(\mathsf{K}\text{'}x)), u_5\text{'}y)$	;
$\mathsf{L05}$:	TypeU5Compose $\trianglerighteq \mathsf{L1} \gg$	$\ell\text{'}(u_5 \circ x)$	;
$\mathsf{L06}$:	TypeU4 $\trianglerighteq \mathsf{L5} \gg$	$\ell\text{'}u_4(u_5 \circ x)$	;
$\mathsf{L07}$:	ElimU2 $\trianglerighteq \mathsf{L4} \trianglerighteq \mathsf{L6} \trianglerighteq \mathsf{L3} \gg$	$z \in_2 u_2(\mathsf{K}\text{'}(\mathsf{K}\text{'}x)), u_5\text{'}y) \vee$ $z \in_2 u_4(u_5 \circ x)$	;
$\mathsf{L08}$:	AlternateA $\triangleright$ ElimU6' $\trianglerighteq \mathsf{L7} \gg$	$z \sim x \vee z \in_2 u_5\text{'}y \vee$ $z \in_2 u_4(u_5 \circ x)$	;
$\mathsf{L09}$:	AlternateB $\triangleright$ ElimU4 $\trianglerighteq \mathsf{L8} \gg$	$z \sim x \vee z \in_2 u_5\text{'}y \vee$ $\exists u\colon z \in_2 (u_5 \circ x)\text{'}u$	;
$\mathsf{L10}$:	Trans $\triangleright \bar{\mathcal{R}} \triangleright \mathsf{L9} \gg$	$z \sim x \vee z \in_2 u_5\text{'}y \vee$ $\exists u\colon z \in_2 u_5\text{'}(x\text{'}u)$	]*

[The Map proof of IntroU6X reads

L1:	Hypothesis $\gg$	$\ell\,'\,x$	;
L2:	Hypothesis $\gg$	$\ell\,'\,y$	;
L3:	TypeU6' $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\,'\,u_2(\mathsf{K}\,'(\mathsf{K}\,'\,x)), u_5\,'\,y)$	;
L4:	TypeU5Compose $\trianglerighteq$ L1 $\gg$	$\ell\,'(u_5 \circ x)$	;
L5:	TypeU4 $\trianglerighteq$ L4 $\gg$	$\ell\,'\,u_4(u_5 \circ x)$	;
L6:	IntroU2a $\trianglerighteq$ L3 $\trianglerighteq$ L5 $\gg$	$u_2(\mathsf{K}\,'(\mathsf{K}\,'\,x)), u_5\,'\,y) \subseteq_2 u_6(x,y)$	;
L7:	IntroU6'X $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$x \in_2 \mathsf{K}\,'(\mathsf{K}\,'\,x)$	;
L8:	ElimSub $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$x \in_2 u_6(x,y)$	]*

[The Map proof of IntroU6Y reads

L1:	Hypothesis $\gg$	$\ell\,'\,x$	;
L2:	Hypothesis $\gg$	$\ell\,'\,y$	;
L3:	TypeU6' $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\,'\,u_2(\mathsf{K}\,'(\mathsf{K}\,'\,x)), u_5\,'\,y)$	;
L4:	TypeU5Compose $\trianglerighteq$ L1 $\gg$	$\ell\,'(u_5 \circ x)$	;
L5:	TypeU4 $\trianglerighteq$ L4 $\gg$	$\ell\,'\,u_4(u_5 \circ x)$	;
L6:	IntroU2a $\trianglerighteq$ L3 $\trianglerighteq$ L5 $\gg$	$u_2(\mathsf{K}\,'(\mathsf{K}\,'\,x)), u_5\,'\,y) \subseteq_2 u_6(x,y)$	;
L7:	IntroU6'Y $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$u_5\,'\,y \subseteq_2 u_2(\mathsf{K}\,'(\mathsf{K}\,'\,x)), u_5\,'\,y)$	;
L8:	TransSub $\trianglerighteq$ L7 $\trianglerighteq$ L6 $\gg$	$u_5\,'\,y \subseteq_2 u_6(x,y)$	]*

[The Map proof of IntroU6XU reads

L01:	Hypothesis $\gg$	$\ell\,'\,x$	;
L02:	Hypothesis $\gg$	$\ell\,'\,y$	;
L03:	Hypothesis $\gg$	$\ell\,'\,u$	;
L04:	TypeU6' $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\,'\,u_2(\mathsf{K}\,'(\mathsf{K}\,'\,x)), u_5\,'\,y)$	;
L05:	TypeU5Compose $\trianglerighteq$ L1 $\gg$	$\ell\,'(u_5 \circ x)$	;
L06:	TypeU4 $\trianglerighteq$ L5 $\gg$	$\ell\,'\,u_4(u_5 \circ x)$	;
L07:	IntroU2b $\trianglerighteq$ L4 $\trianglerighteq$ L6 $\gg$	$u_4(u_5 \circ x) \subseteq_2 u_6(x,y)$	;
L08:	TypeU6 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\,'\,u_6(x,y)$	;
L09:	ElimU4Sub $\trianglerighteq$ L5 $\trianglerighteq$ L8 $\trianglerighteq$ L3 $\trianglerighteq$ L7 $\gg$	$(u_5 \circ x)\,'\,u \subseteq_2 u_6(x,y)$	;
L10:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L9 $\gg$	$u_5\,'(x\,'\,u) \subseteq_2 u_6(x,y)$	]*

A.50 Transitivity of $[\,u_6(x,y)\,]$

A proof of the transitivity of $[\,u_6(x,y)\,]$ naturally falls in six parts which may then be combined.

[Map lemma

TrU6a$_{204}$:	$\ell x, y, z\colon \exists a\colon y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow x \in_\ell y \rightarrow z \sim x \rightarrow z \in_\ell u_6(x, y)$	;
TrU6b$_{204}$:	$\ell x, y, z\colon \exists a\colon y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow z \in_2 u_5\text{'}y \rightarrow z \in_\ell u_6(x, y)$	;
TrU6c$_{205}$:	$\ell x, y, z\colon \forall y\colon \exists a\colon x\text{'}y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow \exists u\colon z \in_2 u_5\text{'}(x\text{'}u) \rightarrow z \in_\ell u_6(x, y)$	;
TrU6d$_{205}$:	$\ell x, y, z, v\colon \forall y\colon \exists a\colon x\text{'}y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow z \sim x \rightarrow z\text{'}v \in_2 u_6(x, y)$	;
TrU6e$_{205}$:	$\ell x, y, z, v\colon \exists a\colon y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow z \in_2 u_5\text{'}y \rightarrow z\text{'}v \in_2 u_6(x, y)$	;
TrU6f$_{206}$:	$\ell x, y, z, v\colon \forall y\colon \exists a\colon x\text{'}y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow \exists u\colon z \in_2 u_5\text{'}(x\text{'}u) \rightarrow z\text{'}v \in_2 u_6(x, y)$	;
TrU6$_{206}$:	$\ell x, y\colon \forall y\colon \exists a\colon x\text{'}y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow \exists a\colon y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow x \in_\ell y \rightarrow \mathrm{Tr}(u_6(x, y))$	]*

[The Map proof of TrU6a reads

L01:	Hypothesis ≫	$\ell\text{'}x$	;
L02:	Hypothesis ≫	$\ell\text{'}y$	;
L03:	Hypothesis ≫	$\ell\text{'}z$	;
L04:	Hypothesis ≫	$x \in_\ell y$	;
L05:	Hypothesis ≫	$\exists a\colon y \in_2 a \wedge \mathrm{Tr}(a)$	;
L06:	Hypothesis ≫	$z \sim x$	;
L07:	IntroTrU5 ⊵ L5 ≫	$\mathrm{Tr}(u_5(y))$	;
L08:	IntroInU5 ⊵ L5 ≫	$y \in_2 u_5(y)$	;
L09:	ElimTrSub ⊵ L7 ⊵ L8 ≫	$y \subseteq_2 u_5(y)$	;
L10:	IntroU6Y ⊵ L1 ⊵ L2 ≫	$u_5\text{'}y \subseteq_2 u_6(x, y)$	;
L11:	TransSub ⊵ L9 ⊵ L10 ≫	$y \subseteq_2 u_6(x, y)$	;
L12:	TypeU6 ⊵ L1 ⊵ L2 ≫	$\ell\text{'}u_6(x, y)$	;
L13:	SubInEll ⊵ L12 ⊵ L11 ⊵ L4 ≫	$x \in_\ell u_6(x, y)$	;
L14:	Block ≫	Begin	;
L15:	Hypothesis ≫	$\ell\text{'}w$	;
L16:	SubtypeLL2 ⊵ L12 ≫	$\ell_2 u_6(x, y)$	;
L17:	TypeInEll ⊵ L15 ⊵ L16 ≫	$!w \in_\ell u_6(x, y)$	;
L18:	Block ≫	End	;
L19:	ExtBool” ⊵ (Gen ⊳ L17) ⊵ L6 ⊵ L13 ≫	$z \in_\ell u_6(x, y)$	]*

[The Map proof of TrU6b reads

L01:	Hypothesis ≫	$\ell\text{'}x$	;
L02:	Hypothesis ≫	$\ell\text{'}y$	;
L03:	Hypothesis ≫	$\ell\text{'}z$	;
L04:	Hypothesis ≫	$\exists a\colon y \in_2 a \wedge \mathrm{Tr}(a)$	;
L05:	Hypothesis ≫	$z \in_2 u_5\text{'}y$	;
L06:	IntroTrU5 ⊵ L4 ≫	$\mathrm{Tr}(u_5\text{'}y)$	;
L07:	ElimTrInEll ⊵ L6 ⊵ L5 ≫	$z \in_\ell u_5\text{'}y$	;
L08:	IntroU6Y ⊵ L1 ⊵ L2 ≫	$u_5\text{'}y \subseteq_2 u_6(x, y)$	;
L09:	SubInEll ⊵ L8 ⊵ L7 ≫	$z \in_\ell u_6(x, y)$	]*

[The Map proof of TrU6c reads

L01:	Hypothesis $\gg$	$\ell \text{'} x$	;
L02:	Hypothesis $\gg$	$\ell \text{'} y$	;
L03:	Hypothesis $\gg$	$\ell \text{'} z$	;
L04:	Hypothesis $\gg$	$\forall y\colon \exists a\colon x \text{'} y \in_2 a \wedge \mathrm{Tr}(a)$	;
L05:	Hypothesis $\gg$	$\exists u\colon z \in_2 u_5 \text{'}(x \text{'} u)$	;
L06:	Define $\gg$	$u \equiv \varepsilon u\colon z \in_2 u_5 \text{'}(x \text{'} u)$	;
L07:	ElimExistsEll $\trianglerighteq$ L5 $\gg$	$\ell \text{'} u$	;
L08:	ElimExists $\trianglerighteq$ L5 $\gg$	$z \in_2 u_5 \text{'}(x \text{'} u)$	;
L09:	ElimAll $\trianglerighteq$ L4 $\trianglerighteq$ L7 $\gg$	$\exists a\colon x \text{'} u \in_2 a \wedge \mathrm{Tr}(a)$	;
L10:	IntroTrU5 $\trianglerighteq$ L9 $\gg$	$\mathrm{Tr}(u_5 \text{'}(x \text{'} u))$	;
L11:	ElimTrInEll $\trianglerighteq$ L10 $\trianglerighteq$ L8 $\gg$	$z \in_\ell u_5 \text{'}(x \text{'} u)$	;
L12:	IntroU6XU $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L7 $\gg$	$u_5 \text{'}(x \text{'} u) \subseteq_2 u_6(x,y)$	;
L13:	SubInEll $\trianglerighteq$ L12 $\trianglerighteq$ L11 $\gg$	$z \in_\ell u_6(x,y)$	]*

[The Map proof of TrU6d reads

L01:	Hypothesis $\gg$	$\ell \text{'} x$	;
L02:	Hypothesis $\gg$	$\ell \text{'} y$	;
L03:	Hypothesis $\gg$	$\ell \text{'} z$	;
L04:	Hypothesis $\gg$	$\ell \text{'} v$	;
L05:	Hypothesis $\gg$	$\forall y\colon \exists a\colon x \text{'} y \in_2 a \wedge \mathrm{Tr}(a)$	;
L06:	Hypothesis $\gg$	$z \sim x$	;
L07:	ElimAll $\trianglerighteq$ L5 $\trianglerighteq$ L4 $\gg$	$\exists a\colon x \text{'} v \in_2 a \wedge \mathrm{Tr}(a)$	;
L08:	IntroInU5 $\trianglerighteq$ L4 $\trianglerighteq$ L7 $\gg$	$x \text{'} v \in_2 u_5(x \text{'} v)$	;
L09:	IntroU6XU $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\gg$	$u_5 \text{'}(x \text{'} v) \subseteq_2 u_6(x,y)$	;
L10:	ElimSub $\trianglerighteq$ L9 $\trianglerighteq$ L8 $\gg$	$x \text{'} v \in_2 u_6(x,y)$	;
L11:	Block $\gg$	Begin	;
L12:	Hypothesis $\gg$	$\quad \ell \text{'} w$	;
L13:	TypeApply $\trianglerighteq$ L12 $\trianglerighteq$ L4 $\gg$	$\quad \ell \text{'}(w \text{'} v)$	;
L14:	TypeU6 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\quad \ell \text{'} u_6(x,y)$	;
L15:	SubtypeLL2 $\trianglerighteq$ L14 $\gg$	$\quad \ell_2 \text{'} u_6(x,y)$	;
L16:	TypeInTwo $\trianglerighteq$ L13 $\trianglerighteq$ L15 $\gg$	$\quad !w \text{'} v \in_2 u_6(x,y)$	;
L17:	Block $\gg$	End	;
L18:	ExtBool" $\trianglerighteq$ (Gen $\triangleright$ L16) $\trianglerighteq$ L6 $\trianglerighteq$ L10 $\gg$	$z \text{'} v \in_2 u_6(x,y)$	]*

[The Map proof of TrU6e reads

L01:	Hypothesis $\gg$	$\ell \text{'} x$	;
L02:	Hypothesis $\gg$	$\ell \text{'} y$	;
L03:	Hypothesis $\gg$	$\ell \text{'} z$	;
L04:	Hypothesis $\gg$	$\ell \text{'} v$	;
L05:	Hypothesis $\gg$	$\exists a\colon y \in_2 a \wedge \mathrm{Tr}(a)$	;
L06:	Hypothesis $\gg$	$z \in_2 u_5 \text{'} y$	;
L07:	IntroTrU5 $\trianglerighteq$ L5 $\gg$	$\mathrm{Tr}(u_5 \text{'} y)$	;
L08:	ElimTrInTwo $\trianglerighteq$ L4 $\trianglerighteq$ L7 $\trianglerighteq$ L6 $\gg$	$z \text{'} v \in_2 u_5 \text{'} y$	;
L09:	IntroU6Y $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$u_5 \text{'} y \subseteq_2 u_6(x,y)$	;
L10:	ElimSub $\trianglerighteq$ L9 $\trianglerighteq$ L8 $\gg$	$z \text{'} v \in_2 u_6(x,y)$	]*

[The Map proof of TrU6f reads

L01:	Hypothesis ≫	$\ell\,'x$	;
L02:	Hypothesis ≫	$\ell\,'y$	;
L03:	Hypothesis ≫	$\ell\,'z$	;
L04:	Hypothesis ≫	$\ell\,'v$	;
L05:	Hypothesis ≫	$\forall y\colon \exists a\colon x\,'y \in_2 a \wedge \mathrm{Tr}(a)$	;
L06:	Hypothesis ≫	$\exists u\colon z \in_2 u_5\,'(x\,'u)$	;
L07:	Define ≫	$u \equiv \varepsilon u\colon z \in_2 u_5\,'(x\,'u)$	;
L08:	ElimExistsEll ⊵ L6 ≫	$\ell\,'u$	;
L09:	ElimExists ⊵ L6 ≫	$z \in_2 u_5\,'(x\,'u)$	;
L10:	ElimAll ⊵ L5 ⊵ L8 ≫	$\exists a\colon x\,'u \in_2 a \wedge \mathrm{Tr}(a)$	;
L11:	IntroTrU5 ⊵ L10 ≫	$\mathrm{Tr}(u_5\,'(x\,'u))$	;
L12:	ElimTrInTwo ⊵ L4 ⊵ L11 ⊵ L9 ≫	$z\,'v \in_2 u_5\,'(x\,'u)$	;
L13:	IntroU6XU ⊵ L1 ⊵ L2 ⊵ L8 ≫	$u_5\,'(x\,'u) \subseteq_2 u_6(x,y)$	;
L14:	ElimSub ⊵ L13 ⊵ L12 ≫	$z\,'v \in_2 u_6(x,y)$	]*

[The Map proof of TrU6 reads

L01:	Hypothesis ≫	$\ell\,'x$	;
L02:	Hypothesis ≫	$\ell\,'y$	;
L03:	Hypothesis ≫	$\forall y\colon \exists a\colon x\,'y \in_2 a \wedge \mathrm{Tr}(a)$	;
L04:	Hypothesis ≫	$\exists a\colon y \in_2 a \wedge \mathrm{Tr}(a)$	;
L05:	Hypothesis ≫	$x \in_\ell y$	;
L06:	TypeU6 ⊵ L1 ⊵ L2 ≫	$\ell\,'u_6(x,y)$	;
L07:	Block ≫	Begin	;
L08:	Hypothesis ≫	$\quad\ell\,'z$	;
L09:	Hypothesis ≫	$\quad\ell\,'v$	;
L10:	Hypothesis ≫	$\quad z \in_2 u_6(x,y)$	;
L11:	ElimU6⊵L1⊵L2⊵L10≫	$\quad z \sim x \vee z \in_2 u_5\,'y \vee \exists u\colon z \in_2 u_5\,'(x\,'u)$	;
L12:	TrU6a⊵L1⊵L2⊵L8⊵L4⊵L5≫	$\quad z \sim x \rightarrow z \in_\ell u_6(x,y)$	;
L13:	TrU6b⊵L1⊵L2⊵L8⊵L4≫	$\quad z \in_2 u_5\,'y \rightarrow z \in_\ell u_6(x,y)$	;
L14:	TrU6c⊵L1⊵L2⊵L8⊵L3≫	$\quad \exists u\colon z \in_2 u_5\,'(x\,'u) \rightarrow z \in_\ell u_6(x,y)$	;
L15:	Alternate3▷L12▷L13▷L14⊵L11≫	$\quad z \in_\ell u_6(x,y)$	;
L16:	TrU6d⊵L1⊵L2⊵L8⊵L9⊵L3≫	$\quad z \sim x \rightarrow z\,'v \in_2 u_6(x,y)$	;
L17:	TrU6e⊵L1⊵L2⊵L8⊵L9⊵L4≫	$\quad z{\in_2}u_5'y \rightarrow z'v{\in_2}u_6(x,y)$	;
L18:	TrU6f⊵L1⊵L2⊵L8⊵L9⊵L3≫	$\quad \exists u\colon z \in_2 u_5\,'(x\,'u) \rightarrow z\,'v \in_2 u_6(x,y)$	;
L19:	Alternate3▷L16▷L17▷L18⊵L11≫	$\quad z\,'v \in_2 u_6(x,y)$	;
L20:	Block ≫	End	;
L21:	IntroTr▷L6▷L15▷L19≫	$\mathrm{Tr}(u_6(x,y))$	]*

A.51 Existence of transitive hulls

[Map lemma

ExistsHull'$_{207}$:	$\exists a\colon \mathsf{T} \in_2 a \wedge \mathrm{Tr}(a)$	;
ExistsHull"$_{207}$:	$\ell x, y\colon \forall y\colon \exists a\colon x \text{'} y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow \exists a\colon y \in_2 a \wedge \mathrm{Tr}(a) \rightarrow$ $x \in_\ell y \rightarrow \exists a\colon x \in_2 a \wedge \mathrm{Tr}(a)$	]*

[The Map proof of ExistsHull' reads

L01:	TypeT $\gg$	$\ell \text{'} \mathsf{T}$	;
L02:	IntroInTwoT $\gg$	$\mathsf{T} \in_2 \mathsf{T}$	;
L03:	IntroTrT $\gg$	$\mathrm{Tr}(\mathsf{T})$	;
L04:	Logic $\trianglerighteq \mathsf{L2} \trianglerighteq \mathsf{L3} \gg$	$\mathsf{T} \in_2 \mathsf{T} \wedge \mathrm{Tr}(\mathsf{T})$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	$\ell \text{'} a$	;
L07:	SubtypeLL2 $\trianglerighteq \mathsf{L6} \gg$	$\ell_2 a$	;
L08:	TypeInTwo $\trianglerighteq \mathsf{L1} \trianglerighteq \mathsf{L7} \gg$	$!\mathsf{T} \in_2 a$	;
L09:	TypeTr $\trianglerighteq \mathsf{L6} \gg$	$!\mathrm{Tr}(a)$	;
L10:	Logic $\trianglerighteq \mathsf{L8} \trianglerighteq \mathsf{L9} \gg$	$!(\mathsf{T} \in_2 a \wedge \mathrm{Tr}(a))$	;
L11:	Block $\gg$	End	;
L12:	IntroExists $\trianglerighteq$ (Gen $\triangleright \mathsf{L10}) \trianglerighteq \mathsf{L1} \trianglerighteq \mathsf{L4} \gg$	$\exists a\colon \mathsf{T} \in_2 a \wedge \mathrm{Tr}(a)$	]*

[The Map proof of ExistsHull" reads

L01:	Hypothesis $\gg$	$\ell \text{'} x$	;
L02:	Hypothesis $\gg$	$\ell \text{'} y$	;
L03:	Hypothesis $\gg$	$\forall y\colon \exists a\colon x \text{'} y \in_2 a \wedge \mathrm{Tr}(a)$	;
L04:	Hypothesis $\gg$	$\exists a\colon y \in_2 a \wedge \mathrm{Tr}(a)$	;
L05:	Hypothesis $\gg$	$x \in_\ell y$	;
L06:	Block $\gg$	Begin	;
L07:	Hypothesis $\gg$	$\ell \text{'} a$	;
L08:	SubtypeLL2 $\trianglerighteq \mathsf{L7} \gg$	$\ell_2 a$	;
L09:	TypeInTwo $\trianglerighteq \mathsf{L1} \trianglerighteq \mathsf{L8} \gg$	$!x \in_2 a$	;
L10:	TypeTr $\trianglerighteq \mathsf{L7} \gg$	$!\mathrm{Tr}(a)$	;
L11:	Logic $\trianglerighteq \mathsf{L9} \trianglerighteq \mathsf{L10} \gg$	$!(x \in_2 a \wedge \mathrm{Tr}(a))$	;
L12:	Block $\gg$	End	;
L13:	TypeU6 $\trianglerighteq \mathsf{L1} \trianglerighteq \mathsf{L2} \gg$	$\ell \text{'} u_6(x, y)$	;
L14:	IntroU6X $\trianglerighteq \mathsf{L1} \trianglerighteq \mathsf{L2} \gg$	$x \in_2 u_6(x, y)$	;
L15:	TrU6 $\trianglerighteq \mathsf{L1} \trianglerighteq \mathsf{L2} \trianglerighteq \mathsf{L3} \trianglerighteq \mathsf{L4} \trianglerighteq \mathsf{L5} \gg$	$\mathrm{Tr}(u_6(x, y))$	;
L16:	Logic $\trianglerighteq \mathsf{L14} \trianglerighteq \mathsf{L15} \gg$	$x {\in_2} u_6(x, y) {\wedge} \mathrm{Tr}(u_6(x, y))$	;
L17:	IntroExists$\trianglerighteq$(Gen$\triangleright\mathsf{L11})\triangleright\mathsf{L13}\triangleright\mathsf{L16}\gg$	$\exists a\colon x \in_2 a \wedge \mathrm{Tr}(a)$	]*

[The Map proof of ExistsHull reads

L01:	ExistsHull' $\gg$	$\exists a\colon \mathsf{T} \in_2 a \wedge \mathrm{Tr}(a)$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\quad \neg x$	;
L04:	Hypothesis $\gg$	$\quad \ell \text{'} x$	;
L05:	Hypothesis $\gg$	$\quad \forall y\colon \exists a\colon x \in_2 a \wedge \mathrm{Tr}(a)$	;
L06:	Hypothesis $\gg$	$\quad \mathsf{E} y\colon \ell \text{'} y \wedge (\exists a\colon y {\in_2} a \wedge \mathrm{Tr}(a)) \wedge x \in_l y$	;
L07:	Block $\gg$	$\quad$ Begin	;
L08:	Hypothesis $\gg$	$\qquad \ell \text{'} y \wedge (\exists a\colon y \in_2 a \wedge \mathrm{Tr}(a)) \wedge x \in_l y$	;
L09:	Logic $\trianglerighteq$ L8 $\gg$	$\qquad \ell \text{'} y$	;
L10:	Logic $\trianglerighteq$ L8 $\gg$	$\qquad \exists a\colon y \in_2 a \wedge \mathrm{Tr}(a)$	;
L11:	Logic $\trianglerighteq$ L8 $\gg$	$\qquad x \in_l y$	;
L12:	SubtypeLL1 $\trianglerighteq$ L4 $\gg$	$\qquad \ell_1 x$	;
L13:	Logic$\trianglerighteq$L9$\trianglerighteq$L11$\trianglerighteq$L12$\gg$	$\qquad x \in_\ell y$	;
L14:	ExistsHull" $\trianglerighteq$ L4 $\trianglerighteq$ L9 $\trianglerighteq$ L5 $\trianglerighteq$ L10 $\trianglerighteq$ L13 $\gg$	$\qquad \exists a\colon x \in_2 a \wedge \mathrm{Tr}(a)$	;
L15:	Block $\gg$	$\quad$ End	;
L16:	ElimPure$\triangleright$L6$\triangleright$L14$\gg$	$\quad \exists a\colon x \in_2 a \wedge \mathrm{Tr}(a)$	;
L17:	Block $\gg$	End	;
L18:	Transfinite"$\triangleright$L1$\triangleright$L16$\gg$	$\forall x\colon \exists a\colon x \in_2 a \wedge \mathrm{Tr}(a)$	;
L19:	ElimAll $\trianglerighteq$ L18 $\gg$	$\ell x\colon \exists a\colon x \in_2 a \wedge \mathrm{Tr}(a)$	]*

A.52 Lemmas about $[\, x^{-1} \,]$

[Map lemma

Compact3$_{208}$:	$\ell x, y\colon \neg x \to \neg y \to \neg x \sim y \to \exists z\colon \neg x \text{'} z \sim y \text{'} z$	;
Compact4$_{209}$:	$\ell x, y, z\colon \exists p\colon \neg x \text{'} z \sim_p y \text{'} z \to \exists p\colon \neg x \sim_p y$	;
Compact1$_{210}$:	$\forall y\colon (\neg \mathsf{T} \sim y \stackrel{\sim}{\Rightarrow} \exists p\colon \neg \mathsf{T} \sim_p y)$	;
Compact2$_{211}$:	$\neg x \to \ell \text{'} x \to \forall z\colon \forall y\colon (\neg x\text{'}z{\sim}y \stackrel{\sim}{\Rightarrow} \exists p\colon \neg x\text{'}z{\sim_p}y) \to$ $\forall y\colon (\neg x \sim y \stackrel{\sim}{\Rightarrow} \exists p\colon \neg x \sim_p y)$	;
Disc1$_{213}$:	$\ell x\colon \ell \text{'} \lambda u.\lambda v.\varepsilon p\colon \neg x \text{'} u \sim_p x \text{'} v$	;
Disc2$_{213}$:	$\ell x\colon !_1 \lambda p.\forall u\colon \forall v\colon (x \text{'} u \sim_p x \text{'} v \Rightarrow x \text{'} u \sim x \text{'} v)$	;
Inverse1$_{215}$:	$\ell p, x\colon \ell \text{'} \lambda z.\varepsilon u\colon x \text{'} u \sim_p z$	;
Inverse$_{216}$:	$\ell x\colon \neg x \to \exists y\colon x \circ y \circ x \sim x$	]*

[The Map proof of Compact3 reads

L01:	Hypothesis $\gg$	$\ell ' x$	;
L02:	Hypothesis $\gg$	$\ell ' y$	;
L03:	Hypothesis $\gg$	$\neg x$	;
L04:	Hypothesis $\gg$	$\neg y$	;
L05:	Hypothesis $\gg$	$\neg x \sim y$	;
L06:	Block $\gg$	Begin	;
L07:	Hypothesis $\gg$	$\quad \ell ' z$	;
L08:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L7 $\gg$	$\quad \ell '(x ' z)$	;
L09:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L7 $\gg$	$\quad \ell '(y ' z)$	;
L10:	TypeEquivK $\trianglerighteq$ L8 $\trianglerighteq$ L9 $\gg$	$\quad !x ' z \sim y ' z$	;
L11:	Logic $\trianglerighteq$ L10 $\gg$	$\quad !\neg x ' z \sim y ' z$	;
L12:	Block $\gg$	End	;
L13:	TypeExists $\trianglerighteq$ (Gen $\triangleright$ L11) $\gg$	$!\exists z \colon \neg x ' z \sim y ' z$	;
L14:	Block $\gg$	Begin	;
L15:	Hypothesis $\gg$	$\quad \neg \exists z \colon \neg x ' z \sim y ' z$	;
L16:	Repetition $\triangleright$ L15 $\gg$	$\quad \forall z \colon x ' z \sim y ' z$	;
L17:	IntroEquivK $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\trianglerighteq$ L16 $\gg$	$\quad x \sim y$	;
L18:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L17 $\gg$	$\quad \mathsf{F}$	;
L19:	Block $\gg$	End	;
L20:	Indirect $\triangleright$ L13 $\triangleright$ L18 $\gg$	$\exists z \colon \neg x ' z \sim y ' z$	$]^*$

[The Map proof of Compact4 reads

L01:	Hypothesis $\gg$	$\ell\text{'}x$	;
L02:	Hypothesis $\gg$	$\ell\text{'}y$	;
L03:	Hypothesis $\gg$	$\ell\text{'}z$	;
L04:	Hypothesis $\gg$	$\exists p\colon \neg x\text{'}z \sim_p y\text{'}z$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	$\ell\text{'}p$	;
L07:	TypeEquivP $\unrhd$ L6 $\unrhd$ L1 $\unrhd$ L2 $\gg$	$!x \sim_p y$	;
L08:	Logic $\unrhd$ L7 $\gg$	$!\neg x \sim_p y$	;
L09:	Block $\gg$	End	;
L10:	Define $\gg$	$p \equiv \varepsilon p\colon \neg x\text{'}z \sim_p y\text{'}z$	;
L11:	ElimExistsEll $\unrhd$ L4 $\gg$	$\ell\text{'}p$	;
L12:	ElimExists $\unrhd$ L4 $\gg$	$\neg x\text{'}z \sim_p y\text{'}z$	;
L13:	Define $\gg$	$q \equiv u_2(p, \mathsf{K}\text{'}(\mathsf{K}\text{'}z))$	;
L14:	TypeKa $\unrhd$ L3 $\gg$	$\ell\text{'}(\mathsf{K}\text{'}z)$	;
L15:	TypeKa $\unrhd$ L14 $\gg$	$\ell\text{'}(\mathsf{K}\text{'}(\mathsf{K}\text{'}z))$	;
L16:	TypeU2 $\unrhd$ L11 $\unrhd$ L15 $\gg$	$\ell\text{'}q$	;
L17:	IntroU2a $\unrhd$ L11 $\unrhd$ L15 $\gg$	$p \subseteq_2 q$	;
L18:	IntroU2b $\unrhd$ L11 $\unrhd$ L15 $\gg$	$\mathsf{K}\text{'}(\mathsf{K}\text{'}z) \subseteq_2 q$	;
L19:	SubtypeLL2 $\unrhd$ L15 $\gg$	$\ell_2\mathsf{K}\text{'}(\mathsf{K}\text{'}z)$	;
L20:	IntroInTwo' $\unrhd$ L19 $\unrhd$ L3 $\unrhd$ L3 $\gg$	$(\mathsf{K}\text{'}(\mathsf{K}\text{'}z))\text{'}z\text{'}z \in_2 \mathsf{K}\text{'}(\mathsf{K}\text{'}z)$	;
L21:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L20 $\gg$	$z \in_2 \mathsf{K}\text{'}(\mathsf{K}\text{'}z)$	;
L22:	ElimSub $\unrhd$ L18 $\unrhd$ L21 $\gg$	$z \in_2 q$	;
L23:	Block $\gg$	Begin	;
L24:	Hypothesis $\gg$	$\forall p\colon x \sim_p y$	;
L25:	ElimAll $\unrhd$ L24 $\unrhd$ L16 $\gg$	$x \sim_q y$	;
L26:	ElimEquivP $\unrhd$ L25 $\unrhd$ L22 $\gg$	$x\text{'}z \sim_q y\text{'}z$	;
L27:	SubEquivP $\unrhd$ L17 $\unrhd$ L26 $\gg$	$x\text{'}z \sim_p y\text{'}z$	;
L28:	Logic $\unrhd$ L12 $\unrhd$ L27 $\gg$	F	;
L29:	Block $\gg$	End	;
L30:	TypeExists $\unrhd$ (Gen $\rhd$ L8) $\gg$	$!\exists p\colon \neg x \sim_p y$	;
L31:	Indirect $\rhd$ L30 $\rhd$ L28 $\gg$	$\exists p\colon \neg x \sim_p y$	]*

[The Map proof of Compact1 reads

L01:	Block $\gg$	Begin	;
L02:	Hypothesis $\gg$	$\quad \ell\,\text{'}\,y$	;
L03:	TypeT $\gg$	$\quad \ell\,\text{'}\,\mathsf{T}$	;
L04:	TypeEquivK $\trianglerighteq$ L3 $\trianglerighteq$ L2 $\gg$	$\quad !\mathsf{T} \sim y$	;
L05:	Logic $\trianglerighteq$ L4 $\gg$	$\quad !\neg\mathsf{T} \sim y$	;
L06:	Block $\gg$	$\quad$ Begin	;
L07:	Hypothesis $\gg$	$\quad\quad \neg\mathsf{T} \sim y$	;
L08:	Logic $\trianglerighteq$ L7 $\gg$	$\quad\quad \neg\mathsf{T} \sim_{\mathsf{T}} y$	;
L09:	Block $\gg$	$\quad\quad$ Begin	;
L10:	Hypothesis $\gg$	$\quad\quad\quad \ell\,\text{'}\,p$	;
L11:	TypeEquivP $\trianglerighteq$ L10 $\trianglerighteq$ L3 $\trianglerighteq$ L2 $\gg$	$\quad\quad\quad !\mathsf{T} \sim_p y$	;
L12:	Logic $\trianglerighteq$ L11 $\gg$	$\quad\quad\quad !\neg\mathsf{T} \sim_p y$	;
L13:	Block $\gg$	$\quad\quad$ End	;
L14:	IntroExists$\triangleright$(Gen$\triangleright$L12)$\triangleright$L3$\triangleright$L8$\gg$	$\quad\quad \exists p\colon \neg\mathsf{T} \sim_p y$	;
L15:	Block $\gg$	$\quad$ End	;
L16:	Deduction” $\triangleright$ L5 $\triangleright$ L14 $\gg$	$\quad \neg\mathsf{T} \sim y \mathrel{\dot{\Rightarrow}} \exists p\colon \neg\mathsf{T} \sim_p y$	;
L17:	Block $\gg$	End	;
L18:	Gen $\triangleright$ L16 $\gg$	$\forall y\colon (\neg\mathsf{T} \sim y \Rightarrow \exists p\colon \neg\mathsf{T} \sim_p y)$	]*

[The Map proof of Compact2 reads

```
L01: Hypothesis $\gg$                                   $\neg x$ ;
L02: Hypothesis $\gg$                                   $\ell ' x$ ;
L03: Hypothesis $\gg$                                   $\forall z\colon \forall y\colon (\neg x ' z \sim y \dot{\Rightarrow}$
                                                        $\exists p\colon \neg x ' z \sim_p y)$ ;
L04: Block $\gg$                                        Begin ;
L05: Hypothesis $\gg$                                     $\ell ' y$ ;
L06: TypeEquivK $\trianglerighteq$ L2 $\trianglerighteq$ L5 $\gg$                 $!x \sim y$ ;
L07: Logic $\trianglerighteq$ L6 $\gg$                                $!\neg x \sim y$ ;
L08: Block $\gg$                                          Begin ;
L09: Hypothesis $\gg$                                       $\neg x \sim y$ ;
L10: Block $\gg$                                            Begin ;
L11: Hypothesis $\gg$                                         $\ell ' p$ ;
L12: TypeEquivP $\trianglerighteq$ L11 $\trianglerighteq$ L2 $\trianglerighteq$ L5 $\gg$             $!x \sim_p y$ ;
L13: Logic $\trianglerighteq$ L12 $\gg$                                   $!\neg x \sim_p y$ ;
L14: Block $\gg$                                            End ;
L15: SubtypeLB $\trianglerighteq$ L5 $\gg$                                $!y$ ;
L16: Block $\gg$                                            Begin ;
L17: Hypothesis $\gg$                                         $y$ ;
L18: Logic $\trianglerighteq$ L1 $\trianglerighteq$ L17 $\gg$                         $\neg x \sim_y y$ ;
L19: IntroExists $\trianglerighteq$
     (Gen $\triangleright$ L13) $\trianglerighteq$ L5 $\trianglerighteq$ L18 $\gg$                $\exists p\colon x \sim_p y$ ;
L20: Block $\gg$                                            End ;
L21: Block $\gg$                                            Begin ;
L22: Hypothesis $\gg$                                         $\neg y$ ;
L23: Compact3 $\trianglerighteq$
     L2 $\trianglerighteq$ L5 $\trianglerighteq$ L1 $\trianglerighteq$ L22 $\trianglerighteq$ L9 $\gg$                 $\exists z\colon \neg x ' z \sim y ' z$ ;
L24: Define $\gg$                                             $z \equiv \varepsilon z\colon \neg x ' z \sim y ' z$ ;
L25: ElimExistsEll $\trianglerighteq$ L23 $\gg$                           $\ell ' z$ ;
L26: ElimExists $\trianglerighteq$ L23 $\gg$                              $\neg x ' z \sim y ' z$ ;
L27: TypeApply $\trianglerighteq$ L5 $\trianglerighteq$ L25 $\gg$                     $\ell '(y ' z)$ ;
L28: ElimAll2 $\trianglerighteq$ L3 $\trianglerighteq$ L25 $\trianglerighteq$ L27 $\gg$               $\neg x ' z \sim y ' z \dot{\Rightarrow}$
                                                              $\exists p\colon \neg x ' z \sim_p y ' z$ ;
L29: Logic $\trianglerighteq$ L26 $\trianglerighteq$ L28 $\gg$                        $\exists p\colon \neg x ' z \sim_p y ' z$ ;
L30: Compact4$\trianglerighteq$L2$\trianglerighteq$L5$\trianglerighteq$L25$\trianglerighteq$L29$\gg$              $\exists p\colon \neg x \sim_p y$ ;
L31: Block $\gg$                                            End ;
L32: TND $\triangleright$ L15 $\triangleright$ L19 $\triangleright$ L30 $\gg$                     $\exists p\colon \neg x \sim_p y$ ;
L33: Block $\gg$                                          End ;
L34: Deduction" $\triangleright$ L7 $\triangleright$ L32 $\gg$                     $\neg x \sim y \dot{\Rightarrow} \exists p\colon \neg x \sim_p y$ ;
L35: Block $\gg$                                        End ;
L36: Gen $\triangleright$ L34 $\gg$                                   $\forall y\colon (\neg x \sim y \dot{\Rightarrow} \exists p\colon \neg x \sim_p y)$ ]*
```

[The Map proof of Compactness reads

L1:	Hypothesis $\gg$	$\ell \text{ ' } x$	;
L2:	Hypothesis $\gg$	$\ell \text{ ' } y$	;
L3:	Hypothesis $\gg$	$\neg x \sim y$	;
L4:	Transfinite' $\rhd$		
	Compact1 $\rhd$ Compact2 $\gg$	$\forall x\colon \forall y\colon (\neg x \sim y \dot{\Rightarrow} \exists p\colon \neg x \sim_p y)$	;
L5:	ElimAll2 $\unrhd$ L4 $\unrhd$ L1 $\unrhd$ L2 $\gg$	$\neg x \sim y \dot{\Rightarrow} \exists p\colon \neg x \sim_p y$	;
L6:	Logic $\unrhd$ L3 $\unrhd$ L5 $\gg$	$\exists p\colon \neg x \sim_p y$	]*

[The Map proof of Disc1 reads

L01:	Hypothesis $\gg$	$\ell \text{ ' } x$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\quad \ell \text{ ' } u$	;
L04:	Block $\gg$	$\quad$ Begin	;
L05:	Hypothesis $\gg$	$\quad\quad \ell \text{ ' } v$	;
L06:	Block $\gg$	$\quad\quad$ Begin	;
L07:	Hypothesis $\gg$	$\quad\quad\quad \ell \text{ ' } p$	;
L08:	TypeEquivP $\unrhd$ L7 $\unrhd$ L3 $\unrhd$ L5 $\gg$	$\quad\quad\quad !u \sim_p v$	;
L09:	Logic $\unrhd$ L8 $\gg$	$\quad\quad\quad !\neg u \sim_p v$	;
L10:	Block $\gg$	$\quad\quad$ End	;
L11:	TypeChoice $\unrhd$ (Gen $\rhd$ L9) $\gg$	$\quad\quad \ell \text{ ' } \varepsilon p\colon \neg u \sim_p v$	;
L12:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L11 $\gg$	$\quad\quad \ell \text{ '} ((\lambda v.\varepsilon p\colon \neg u \sim_p v) \text{ ' } v)$	;
L13:	Block $\gg$	$\quad$ End	;
L14:	Gen $\rhd$ L12 $\gg$	$\quad \ell_1 \lambda v.\varepsilon p\colon \neg u \sim_p v$	;
L15:	TypeCompose $\unrhd$ L14 $\unrhd$ L1 $\gg$	$\quad \ell \text{ '} ((\lambda v.\varepsilon p\colon \neg u \sim_p v) \circ x)$	;
L16:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L15 $\gg$	$\quad \ell \text{ '} ((\lambda u.\lambda v.\varepsilon p\colon \neg u \sim_p x \text{ ' } v) \text{ ' } u)$	;
L17:	Block $\gg$	End	;
L18:	Gen $\rhd$ L16 $\gg$	$\ell_1 \lambda u.\lambda v.\varepsilon p\colon \neg u \sim_p x \text{ ' } v$	;
L19:	TypeCompose $\unrhd$ L18 $\unrhd$ L1 $\gg$	$\ell \text{ '} ((\lambda u.\lambda v.\varepsilon p\colon \neg u \sim_p x \text{ ' } v) \circ x)$	;
L20:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L19 $\gg$	$\ell \text{ ' } \lambda u.\lambda v.\varepsilon p\colon \neg x \text{ ' } u \sim_p x \text{ ' } v$	]*

[The Map proof of Disc2 reads

L01:	Hypothesis $\gg$	$\ell\,'\,x$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\ell\,'\,p$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\ell\,'\,u$	;
L06:	Block $\gg$	Begin	;
L07:	Hypothesis $\gg$	$\ell\,'\,v$	;
L07:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L5 $\gg$	$\ell\,'(x\,'\,u)$	;
L08:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L7 $\gg$	$\ell\,'(x\,'\,v)$	;
L09:	TypeEquivP$\trianglerighteq$L3$\trianglerighteq$L7$\trianglerighteq$L8$\gg$	$!x\,'\,u \sim_p x\,'\,v$	;
L10:	TypeEquivK $\trianglerighteq$ L7 $\trianglerighteq$ L8 $\gg$	$!x\,'\,u \sim x\,'\,v$	;
L11:	Logic $\trianglerighteq$ L9 $\trianglerighteq$ L10 $\gg$	$!(x\,'\,u \sim_p x\,'\,v \Rightarrow x\,'\,u \sim x\,'\,v)$	;
L12:	Block $\gg$	End	;
L13:	TypeAll $\trianglerighteq$ (Gen $\triangleright$ L11) $\gg$	$!\forall v\colon (x'u{\sim_p}x'v \Rightarrow x'u{\sim}x'v)$	;
L14:	Block $\gg$	End	;
L15:	TypeAll $\trianglerighteq$ (Gen $\triangleright$ L13) $\gg$	$!\forall u\colon \forall v\colon (x'u{\sim_p}x'v \Rightarrow x'u{\sim}x'v)$	;
L16:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L15 $\gg$	$!(\lambda p.\forall u\colon \forall v\colon (x\,'\,u \sim_p x\,'\,v \Rightarrow x\,'\,u \sim x\,'\,v))\,'\,p$	;
L17:	Block $\gg$	End	;
L18:	Gen $\triangleright$ L16 $\gg$	$!_1\,\lambda p.\forall u\colon \forall v\colon (x'u{\sim_p}x'v \Rightarrow x'u{\sim}x'v)$	]*

[The Map proof of Discriminate reads

```
L01: Hypothesis $\gg$                                      $\ell\,'\,x$                                        ;
L02: Define $\gg$                                          $q \equiv \lambda u.\lambda v.\varepsilon p\colon \neg x\,'\,u \sim_p x\,'\,v$   ;
L03: Disc1 $\trianglerighteq$ L1 $\gg$                     $\ell\,'\,q$                                        ;
L04: Define $\gg$                                          $p \equiv u_4(u_4(q))$                              ;
L05: TypeU4 $\trianglerighteq$ L3 $\gg$                    $\ell\,'\,u_4(q)$                                   ;
L06: TypeU4 $\trianglerighteq$ L5 $\gg$                    $\ell\,'\,p$                                        ;
L07: Block $\gg$                                           Begin                                               ;
L08: Hypothesis $\gg$                                        $\ell\,'\,u$                                      ;
L09: Hypothesis $\gg$                                        $\ell\,'\,v$                                      ;
L10: TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L8 $\gg$   $\ell\,'(x\,'\,u)$                          ;
L11: TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L9 $\gg$   $\ell\,'(x\,'\,v)$                          ;
L12: TypeEquivP$\trianglerighteq$L6$\trianglerighteq$L10$\trianglerighteq$L11$\gg$   $!x\,'\,u \sim_p x\,'\,v$  ;
L13: TypeEquivK $\trianglerighteq$ L10 $\trianglerighteq$ L11 $\gg$   $!x\,'\,u \sim x\,'\,v$                  ;
L14: Block $\gg$                                             Begin                                             ;
L15: Hypothesis $\gg$                                          $x\,'\,u \sim_p x\,'\,v$                        ;
L16: Block $\gg$                                               Begin                                           ;
L17: Hypothesis $\gg$                                            $\neg x\,'\,u \sim x\,'\,v$                   ;
L18: Compactness$\trianglerighteq$L10$\trianglerighteq$L11$\trianglerighteq$L17$\gg$   $\exists r\colon \neg x\,'\,u \sim_r x\,'\,v$   ;
L19: Define $\gg$                                                $r \equiv \varepsilon r\colon \neg x\,'\,u \sim_r x\,'\,v$   ;
L20: ElimExistsEll $\trianglerighteq$ L18 $\gg$                  $\ell\,'\,r$                                  ;
L21: ElimExists $\trianglerighteq$ L18 $\gg$                     $\neg x\,'\,u \sim_r x\,'\,v$                 ;
L22: TypePair $\trianglerighteq$ L8 $\trianglerighteq$ L9 $\gg$  $\ell\,'(u :: v)$                             ;
L23: IntroSubU4 $\trianglerighteq$ L5 $\trianglerighteq$ L16 $\gg$   $u_4(q)\,'(u :: v) \subseteq_2 p$         ;
L24: Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L23 $\gg$   $r \subseteq_2 p$                 ;
L25: SubEquivP $\trianglerighteq$ L24 $\trianglerighteq$ L15 $\gg$   $x\,'\,u \sim_r x\,'\,v$                  ;
L26: Logic $\trianglerighteq$ L21 $\trianglerighteq$ L25 $\gg$   $\mathsf{F}$                                  ;
L27: Block $\gg$                                               End                                             ;
L28: Indirect $\triangleright$ L13 $\triangleright$ L26 $\gg$  $x\,'\,u \sim x\,'\,v$                          ;
L29: Block $\gg$                                             End                                               ;
L30: CDeduction$\triangleright$L12$\triangleright$L13$\triangleright$L28$\gg$   $x'u{\sim_p}x'v \Rightarrow x'u{\sim}x'v$   ;
L31: Block $\gg$                                           End                                                 ;
L32: Gen2 $\triangleright$ L30 $\gg$                       $\forall u\colon \forall v\colon (x'u{\sim_p}x'v \Rightarrow x'u{\sim}x'v)$   ;
L33: IntroExists $\trianglerighteq$
     (Disc2 $\trianglerighteq$ L1) $\trianglerighteq$ L6 $\trianglerighteq$ L32 $\gg$   $\exists p\colon \forall u\colon \forall v\colon (x'u{\sim_p}x'v \Rightarrow x'u{\sim}x'v)$   ]*
```

[The Map proof of Inverse1 reads

L01:	Hypothesis $\gg$	$\ell\,'p$	;
L02:	Hypothesis $\gg$	$\ell\,'x$	;
L03:	Define $\gg$	$y \equiv \lambda z.\varepsilon u\colon x\,'u \sim_p z$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\quad z \sim_p \underline{z}$	;
L06:	Block $\gg$	$\quad$ Begin	;
L07:	Hypothesis $\gg$	$\quad\quad \ell\,'u$	;
L08:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L7 $\gg$	$\quad\quad \ell\,'(x\,'u)$	;
L09:	StrictEquivPX $\trianglerighteq$ L5 $\gg$	$\quad\quad \ell\,'z$	;
L10:	TypeEquivP $\trianglerighteq$ L1 $\trianglerighteq$ L8 $\trianglerighteq$ L9 $\gg$	$\quad\quad !x\,'u \sim_p z$	;
L11:	StrictEquivPY $\trianglerighteq$ L5 $\gg$	$\quad\quad \ell\,'\underline{z}$	;
L12:	TypeEquivP $\trianglerighteq$ L1 $\trianglerighteq$ L8 $\trianglerighteq$ L11 $\gg$	$\quad\quad !x\,'u \sim_p \underline{z}$	;
L13:	Block $\gg$	$\quad\quad$ Begin	;
L14:	Hypothesis $\gg$	$\quad\quad\quad x\,'u \sim_p z$	;
L15:	TransEquivP $\trianglerighteq$ L14 $\trianglerighteq$ L5 $\gg$	$\quad\quad\quad x\,'u \sim_p \underline{z}$	;
L16:	Block $\gg$	$\quad\quad$ End	;
L17:	Block $\gg$	$\quad\quad$ Begin	;
L18:	Hypothesis $\gg$	$\quad\quad\quad x\,'u \sim_p \underline{z}$	;
L19:	ComEquivP $\trianglerighteq$ L5 $\gg$	$\quad\quad\quad \underline{z} \sim_p z$	;
L20:	TransEquivP $\trianglerighteq$ L18 $\trianglerighteq$ L19 $\gg$	$\quad\quad\quad x\,'u \sim_p z$	;
L21:	Block $\gg$	$\quad\quad$ End	;
L22:	CDeduction$\triangleright$L10$\triangleright$L12$\triangleright$L15$\gg$	$\quad\quad x\,'u \sim_p z \Rightarrow x\,'u \sim_p \underline{z}$	;
L23:	CDeduction$\triangleright$L12$\triangleright$L10$\triangleright$L20$\gg$	$\quad\quad x\,'u \sim_p \underline{z} \Rightarrow x\,'u \sim_p z$	;
L24:	Logic $\trianglerighteq$ L22 $\trianglerighteq$ L23 $\gg$	$\quad\quad x\,'u \sim_p z \dot{\Leftrightarrow} x\,'u \sim_p \underline{z}$	;
L25:	Block $\gg$	$\quad$ End	;
L26:	TypeChoice $\trianglerighteq$ (Gen $\triangleright$ L10) $\gg$	$\quad \ell\,'\varepsilon u\colon x\,'u \sim_p z$	;
L27:	Ackermann $\trianglerighteq$ (Gen $\triangleright$ L24) $\gg$	$\quad \varepsilon u\colon x'u{\sim_p}z \equiv \varepsilon u\colon x'u{\sim_p}\underline{z}$	;
L28:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L26 $\gg$	$\quad \ell\,'(y\,'z)$	;
L29:	T4 $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L27 $\triangleright$ $\bar{\mathcal{R}}$ $\gg$	$\quad y\,'z \equiv y\,'\underline{z}$	;
L30:	RefEquivK $\trianglerighteq$ L28 $\gg$	$\quad y\,'z \sim y\,'z$	;
L31:	Replace' $\triangleright$ L29 $\triangleright$ L30 $\gg$	$\quad y\,'z \sim y\,'\underline{z}$	;
L32:	Block $\gg$	End	;
L33:	IntroInEll $\triangleright$ L1 $\triangleright$ L31 $\gg$	$y \in_\ell p$	;
L34:	IntroEll $\trianglerighteq$ L33 $\gg$	$\ell\,'y$	]*

[The Map proof of Inverse reads

L01:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L02:	Hypothesis $\gg$	$\neg x$	;
L03:	Discriminate $\trianglerighteq$ L1 $\gg$	$\exists p{:}\,\forall u{:}\,\forall v{:}\,(x\,\text{'}\,u \sim_p x\,\text{'}\,v \Rightarrow x\,\text{'}\,u \sim x\,\text{'}\,v)$	;
L04:	Define $\gg$	$p \equiv \varepsilon p{:}\,\forall u{:}\,\forall v{:}\,(x\,\text{'}\,u \sim_p x\,\text{'}\,v \Rightarrow x\,\text{'}\,u \sim x\,\text{'}\,v)$	;
L05:	ElimExistsEll $\trianglerighteq$ L3 $\gg$	$\ell\,\text{'}\,p$	;
L06:	ElimExists $\trianglerighteq$ L4 $\gg$	$\forall u{:}\,\forall v{:}\,(x\,\text{'}\,u \sim_p x\,\text{'}\,v \Rightarrow x\,\text{'}\,u \sim x\,\text{'}\,v)$	;
L07:	Block $\gg$	Begin	;
L08:	Hypothesis $\gg$	$\quad\ell\,\text{'}\,y$	;
L09:	SubtypeLL1 $\trianglerighteq$ L1 $\gg$	$\quad\ell_1 x$	;
L10:	TypeCompose $\trianglerighteq$ L9 $\trianglerighteq$ L8 $\gg$	$\quad\ell\,\text{'}(x \circ y)$	;
L11:	SubtypeLL1 $\trianglerighteq$ L10 $\gg$	$\quad\ell_1 x \circ y$	;
L12:	TypeCompose $\trianglerighteq$ L11 $\trianglerighteq$ L1 $\gg$	$\quad\ell\,\text{'}(x \circ y \circ x)$	;
L13:	TypeEquivK $\trianglerighteq$ L12 $\trianglerighteq$ L1 $\gg$	$\quad!x \circ y \circ x \sim x$	;
L14:	Block $\gg$	End	;
L15:	Define $\gg$	$y \equiv \lambda z.\varepsilon u{:}\, x\,\text{'}\,u \sim_p z$	;
L16:	Inverse1 $\trianglerighteq$ L5 $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}\,y$	;
L17:	Reduction $\gg$	$\neg(x \circ y \circ x)$	;
L18:	Block $\gg$	Begin	;
L19:	Hypothesis $\gg$	$\quad\ell\,\text{'}\,z$	;
L20:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L19 $\gg$	$\quad\ell\,\text{'}(x\,\text{'}\,z)$	;
L21:	RefEquivP $\trianglerighteq$ L5 $\trianglerighteq$ L20 $\gg$	$\quad x\,\text{'}\,z \sim_p x\,\text{'}\,z$	;
L22:	Block $\gg$	$\quad$Begin	;
L23:	Hypothesis $\gg$	$\qquad\ell\,\text{'}\,u$	;
L24:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L23 $\gg$	$\qquad\ell\,\text{'}(x\,\text{'}\,u)$	;
L25:	TypeEquivP$\trianglerighteq$L5$\trianglerighteq$L24$\trianglerighteq$L20$\gg$	$\qquad!x\,\text{'}\,u \sim_p x\,\text{'}\,z$	;
L26:	Block $\gg$	$\quad$End	;
L27:	IntroExists $\trianglerighteq$ (Gen $\triangleright$ L25) $\trianglerighteq$ L19 $\trianglerighteq$ L21 $\gg$	$\quad\exists u{:}\, x\,\text{'}\,u \sim_p x\,\text{'}\,z$	;
L28:	Define $\gg$	$\quad u \equiv \varepsilon u{:}\, x\,\text{'}\,u \sim_p x\,\text{'}\,z$	;
L29:	ElimExistsEll $\trianglerighteq$ L27 $\gg$	$\quad\ell\,\text{'}\,u$	;
L30:	ElimExists $\trianglerighteq$ L27 $\gg$	$\quad x\,\text{'}\,u \sim_p x\,\text{'}\,z$	;
L31:	ElimAll2 $\trianglerighteq$ L6 $\trianglerighteq$ L29 $\trianglerighteq$ L19 $\gg$	$\quad x\,\text{'}\,u \sim_p x\,\text{'}\,z \Rightarrow x\,\text{'}\,u \sim x\,\text{'}\,z$	;
L32:	Logic $\trianglerighteq$ L30 $\trianglerighteq$ L31 $\gg$	$\quad x\,\text{'}\,u \sim x\,\text{'}\,z$	;
L33:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L32 $\gg$	$\quad(x \circ y \circ x)\,\text{'}\,z \sim x\,\text{'}\,z$	;
L34:	Block $\gg$	End	;
L35:	IntroEquivK $\trianglerighteq$ L17 $\trianglerighteq$ L2 $\trianglerighteq$ L1 $\trianglerighteq$ (Gen $\triangleright$ L33) $\gg$	$x \circ y \circ x \sim x$	;
L36:	IntroExists $\trianglerighteq$ (Gen $\triangleright$ L13) $\trianglerighteq$ L16 $\trianglerighteq$ L35 $\gg$	$\exists y{:}\, x \circ y \circ x \sim x$	]*

[The Map proof of TypeInv reads

L01:	Hypothesis $\gg$	ℓ'x	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	ℓ'y	;
L04:	SubtypeLL1 $\trianglerighteq$ L1 $\gg$	$\ell_1 x$	;
L05:	TypeCompose $\trianglerighteq$ L4 $\trianglerighteq$ L3 $\gg$	ℓ'$(x \circ y)$	;
L06:	SubtypeLL1 $\trianglerighteq$ L5 $\gg$	$\ell_1 x \circ y$	;
L07:	TypeCompose $\trianglerighteq$ L6 $\trianglerighteq$ L1 $\gg$	ℓ'$(x \circ y \circ x)$	;
L08:	TypeEquivK $\trianglerighteq$ L7 $\trianglerighteq$ L1 $\gg$	$!x \circ y \circ x \sim x$	;
L09:	Block $\gg$	End	;
L10:	TypeChoice $\trianglerighteq$ (Gen $\triangleright$ L8) $\gg$	$\varepsilon y\colon x \circ y \circ x \sim x$	;
L11:	Repetition $\triangleright$ L10 $\gg$	ℓ'x^{-1}	]*

[The Map proof of IntroInv reads

L1:	Hypothesis $\gg$	ℓ'x	;
L2:	Hypothesis $\gg$	$\neg x$	;
L3:	Inverse $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\exists y\colon x \circ y \circ x \sim x$	;
L4:	ElimExists $\trianglerighteq$ L3 $\gg$	$x \circ x^{-1} \circ x \sim x$	]*

A.53 Elementary ZFC types

[Map lemma

z-TypeNot$_{218}$:	$!a\colon !\neg a$	;
z-TypeImply$_{218}$:	$!a, b\colon !(a \Rightarrow b)$	;
z-TypeExists$_{218}$:	$\langle x{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \ell$'$x \rightarrow !a \vdash !\forall x\colon a$	]*

[The Map proof of z-TypeNot reads
L1: Logic $\gg$ $!a\colon !\neg a$]*

[The Map proof of z-TypeImply reads
L1: Logic $\gg$ $!a, b\colon !(a \Rightarrow b)$]*

[The Map proof of z-TypeExists reads

L1:	Premise $\gg$	ℓ'$x \rightarrow !a$	;
L2:	TypeExists $\trianglerighteq$ (Gen $\triangleright$ L1) $\gg$	$!\forall x\colon a$	]*

A.54 Elementary ZFC rules

[Map lemma

z-mp$_{219}$:	$a \rightarrow a \Rightarrow b \rightarrow b$	;
z-gen$_{219}$:	$\langle x{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \ell$'$x \rightarrow a \vdash \forall x\colon a$	;
z-a1$_{219}$:	$!a, b\colon a \Rightarrow (b \Rightarrow a)$	;
z-a2$_{219}$:	$!a, b, c\colon (a \Rightarrow (b \Rightarrow c)) \Rightarrow ((a \Rightarrow b) \Rightarrow (a \Rightarrow c))$	;
z-a3$_{219}$:	$!a, b\colon (\neg b \Rightarrow \neg a) \Rightarrow ((\neg b \Rightarrow a) \Rightarrow b)$	;
z-a4$_{219}$:	$\langle x{\in}\mathcal{V}; a, b, t{\in}\mathcal{T}\rangle b \simeq \langle a \mid x{:=}t\rangle \Vdash$ ℓ'$x \rightarrow !a \vdash !b \vdash \ell$'$t \vdash \forall x\colon a \Rightarrow b$	;
z-a5$_{219}$:	$\langle x{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \text{notfree}(x; a) \Vdash$ $!a \vdash \ell$'$x \rightarrow !b \vdash \forall x\colon (a \Rightarrow b) \Rightarrow (a \Rightarrow \forall x\colon b)$	]*

[The Map proof of z-mp reads
L1: Logic $\gg$ $a \to a \Rightarrow b \to b$]*

[The Map proof of z-gen reads
L1: Premise $\gg$ $\ell\,'\,x \to a$;
L2: Gen $\rhd$ L1 $\gg$ $\forall x\colon a$]*

[The Map proof of z-a1 reads
L1: Logic $\gg$ $!a, b\colon a \Rightarrow (b \Rightarrow a)$]*

[The Map proof of z-a2 reads
L1: Logic $\gg$ $!a, b, c\colon (a \Rightarrow (b \Rightarrow c)) \Rightarrow ((a \Rightarrow b) \Rightarrow (a \Rightarrow c))$]*

[The Map proof of z-a3 reads
L1: Logic $\gg$ $!a, b\colon (\neg b \Rightarrow \neg a) \Rightarrow ((\neg b \Rightarrow a) \Rightarrow b)$]*

[The Map proof of z-a4 reads

L1:	Premise $\gg$	$\ell\,'\,x \to !a$	;
L2:	Premise $\gg$	$\ell\,'\,t$	;
L3:	L1 $\unrhd$ L2 $\gg$	$!b$	;
L4:	TypeAll $\unrhd$ (Gen $\rhd$ L1) $\gg$	$!\forall x\colon a$	;
L5:	Block $\gg$	Begin	;
L6:	Hypothesis $\gg$	$\forall x\colon a$	;
L7:	ElimAll $\unrhd$ L6 $\unrhd$ L2 $\gg$	b	;
L8:	Block $\gg$	End	;
L9:	CDeduction $\rhd$ L4 $\rhd$ L3 $\rhd$ L7 $\gg$	$\forall x\colon a \Rightarrow b$	]*

[The Map proof of z-a5 reads

L01:	Premise $\gg$	$!a$	;
L02:	Premise $\gg$	$\ell\,{}^{\prime}\,x \rightarrow\, !b$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\quad \ell\,{}^{\prime}\,x$	;
L05:	L2 $\unrhd$ L4 $\gg$	$\quad !b$	;
L06:	Logic $\unrhd$ L1 $\unrhd$ L5 $\gg$	$\quad !(a \Rightarrow b)$	;
L07:	Block $\gg$	End	;
L08:	TypeAll $\unrhd$ (Gen $\rhd$ L5) $\gg$	$!\forall x\colon b$	;
L09:	TypeAll $\unrhd$ (Gen $\rhd$ L6) $\gg$	$!\forall x\colon (a \Rightarrow b)$	;
L10:	Logic $\unrhd$ L1 $\unrhd$ L8 $\gg$	$!(a \Rightarrow \forall x\colon b)$	;
L11:	Block $\gg$	Begin	;
L12:	Hypothesis $\gg$	$\quad \forall x\colon (a \Rightarrow b)$	;
L13:	Block $\gg$	$\quad$ Begin	;
L14:	Hypothesis $\gg$	$\quad\quad a$	;
L15:	Block $\gg$	$\quad\quad$ Begin	;
L16:	Hypothesis $\gg$	$\quad\quad\quad \ell\,{}^{\prime}\,x$	;
L17:	ElimAll $\unrhd$ L12 $\unrhd$ L16 $\gg$	$\quad\quad\quad a \Rightarrow b$	;
L18:	Logic $\unrhd$ L14 $\unrhd$ L17 $\gg$	$\quad\quad\quad b$	;
L19:	Block $\gg$	$\quad\quad$ End	;
L20:	Gen $\rhd$ L18 $\gg$	$\quad\quad \forall x\colon b$	;
L21:	Block $\gg$	$\quad$ End	;
L22:	CDeduction $\rhd$ L1 $\rhd$ L8 $\rhd$ L20 $\gg$	$\quad a \Rightarrow \forall x\colon b$	;
L23:	Block $\gg$	End	;
L24:	CDeduction $\rhd$ L9 $\rhd$ L10 $\rhd$ L22 $\gg$	$\forall x\colon (a \Rightarrow b) \Rightarrow (a \Rightarrow \forall x\colon b)$	]*

A.55 Lemmas about $[\, x = y\,]$

$$[\, x = y \quad \doteq \quad x\left\{\begin{array}{l} y\left\{\begin{array}{l}\mathsf{T}\\ \mathsf{F}\end{array}\right. \\ \\ y\left\{\begin{array}{l}\mathsf{F}\\ (\forall u \exists v\colon x\,{}^{\prime}\,u = y\,{}^{\prime}\,v) \mathrel{\dot\wedge} (\forall u \exists v\colon x\,{}^{\prime}\,v = y\,{}^{\prime}\,u)\end{array}\right.\end{array}\right. \quad]^*$$

$$[\, v_x(x, y, u) \quad \doteq \quad \varepsilon v\colon x\,{}^{\prime}\,v = y\,{}^{\prime}\,u \quad]^*$$

$$[\, v_y(x, y, u) \quad \doteq \quad \varepsilon v\colon x\,{}^{\prime}\,u = y\,{}^{\prime}\,v \quad]^*$$

[Map lemma

z-TypeEqual$_{221}$:	$\ell x, y\colon !x = y$	;
z-ElimEqualXE$_{222}$:	$\ell u\colon x = y \to \exists v\colon x \text{'} v = y \text{'} u$	;
z-ElimEqualX$_{223}$:	$\ell u\colon x = y \to x \text{'} v_x(x, y, u) = y \text{'} u$	;
z-ElimEqualXEll$_{223}$:	$\ell u\colon x = y \to \ell \text{'} v_x(x, y, u)$	;
z-ElimEqualYE$_{223}$:	$\ell u\colon x = y \to \exists v\colon x \text{'} u = y \text{'} v$	;
z-ElimEqualY$_{223}$:	$\ell u\colon x = y \to x \text{'} u = y \text{'} v_y(x, y, u)$	;
z-ElimEqualYEll$_{223}$:	$\ell u\colon x = y \to \ell \text{'} v_y(x, y, u)$	;
z-IntroEqual$_{224}$:	$\langle u \in \mathcal{V}; a, b, r, s \in \mathcal{T} \rangle \text{notfree}(u; a, b) \Vdash$ $\ell \text{'} a \vdash \ell \text{'} b \vdash a \dot{\Leftrightarrow} b \vdash$ $\ell \text{'} u \to a \text{'} u = b \text{'} r \vdash \ell \text{'} u \to \ell \text{'} r \vdash$ $\ell \text{'} u \to a \text{'} s = b \text{'} u \vdash \ell \text{'} u \to \ell \text{'} s \vdash$ $a = b$	;
z-RefEqual$_{224}$:	$\ell x\colon x = x$	;
z-RefEqual'$_{225}$:	$x \sim y \to x = y$	;
z-ComEqual$_{226}$:	$\ell x, y\colon x = y \to y = x$	;
z-TransEqual$_{229}$:	$\ell x, y, z\colon x = y \to y = z \to x = z$	]*

[The Map proof of z-TypeEqual reads

```
L01: Block $\gg$                                          Begin                                      ;
L02: Hypothesis $\gg$                                       $\ell\,'\,y$                                   ;
L03: SubtypeLB $\unrhd$ L2 $\gg$                              $!y$                                       ;
L04: Logic $\unrhd$ L3 $\gg$                                  $!\mathsf{T} = y$                          ;
L05: Block $\gg$                                          End                                        ;
L06: Gen $\rhd$ L4 $\gg$                                    $\forall y\colon !\mathsf{T} = y$          ;
L07: Block $\gg$                                          Begin                                      ;
L08: Hypothesis $\gg$                                       $\neg x$                                   ;
L09: Hypothesis $\gg$                                       $\ell\,'\,x$                                   ;
L10: Hypothesis $\gg$                                       $\forall z\colon \forall y\colon !x\,'\,z = y$     ;
L11: Block $\gg$                                            Begin                                    ;
L12: Hypothesis $\gg$                                         $\ell\,'\,y$                                 ;
L13: SubtypeLB $\unrhd$ L12 $\gg$                               $!y$                                     ;
L14: Logic $\unrhd$ L8 $\gg$                                    $y \to !x = y$                           ;
L15: Block $\gg$                                              Begin                                  ;
L16: Hypothesis $\gg$                                           $\neg y$                               ;
L17: Block $\gg$                                                Begin                                ;
L18: Hypothesis $\gg$                                             $\ell\,'\,u$                             ;
L19: Block $\gg$                                                  Begin                              ;
L20: Hypothesis $\gg$                                               $\ell\,'\,v$                           ;
L21: TypeApply $\unrhd$ L12 $\unrhd$ L18 $\gg$                          $\ell\,'(y\,'\,u)$                       ;
L22: TypeApply $\unrhd$ L12 $\unrhd$ L20 $\gg$                          $\ell\,'(y\,'\,v)$                       ;
L23: ElimAll2 $\unrhd$ L10 $\unrhd$ L18 $\unrhd$ L22 $\gg$                  $!x\,'\,u = y\,'\,v$                 ;
L24: ElimAll2 $\unrhd$ L10 $\unrhd$ L20 $\unrhd$ L21 $\gg$                  $!x\,'\,v = y\,'\,u$                 ;
L25: Block $\gg$                                                  End                                ;
L26: TypeExists $\unrhd$ (Gen $\rhd$ L23) $\gg$                       $!\exists v\colon x\,'\,u = y\,'\,v$     ;
L27: TypeExists $\unrhd$ (Gen $\rhd$ L24) $\gg$                       $!\exists v\colon x\,'\,v = y\,'\,u$     ;
L28: Block $\gg$                                                End                                  ;
L29: TypeAll $\unrhd$ (Gen $\rhd$ L26) $\gg$                          $!\forall u\colon \exists v\colon x\,'\,u = y\,'\,v$ ;
L30: TypeAll $\unrhd$ (Gen $\rhd$ L27) $\gg$                          $!\forall u\colon \exists v\colon x\,'\,v = y\,'\,u$ ;
L31: Logic $\unrhd$ L8 $\unrhd$ L16 $\unrhd$ L29 $\unrhd$ L30 $\gg$            $!x = y$                           ;
L32: Block $\gg$                                              End                                    ;
L33: TND $\rhd$ L13 $\rhd$ L14 $\rhd$ L31 $\gg$                       $!x = y$                               ;
L34: Block $\gg$                                            End                                      ;
L35: Gen $\rhd$ L33 $\gg$                                     $\forall y\colon !x = y$                 ;
L36: Block $\gg$                                          End                                        ;
L37: Transfinite' $\rhd$ L6 $\rhd$ L35 $\gg$                    $\forall x\colon \forall y\colon !x = y$   ;
L38: ElimAll2 $\unrhd$ L37 $\gg$                              $\ell x, y\colon !x = y$                   ]*
```

[The Map proof of z-ElimEqualXE reads

L01:	Hypothesis $\gg$	$\ell\text{'}u$	;
L02:	Hypothesis $\gg$	$x = y$	;
L03:	Logic $\unrhd$ L2 $\gg$	$!x$	;
L04:	Logic $\unrhd$ L2 $\gg$	$x \rightarrow \exists v\colon x\text{'}v = y\text{'}u$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	$\neg x$	;
L07:	Logic $\unrhd$ L2 $\unrhd$ L6 $\gg$	$\forall u \exists v\colon x\text{'}v = y\text{'}u$	;
L08:	ElimAll $\unrhd$ L7 $\unrhd$ L1 $\gg$	$\exists v\colon x\text{'}v = y\text{'}u$	;
L09:	Block $\gg$	End	;
L10:	TND $\rhd$ L3 $\rhd$ L4 $\rhd$ L8 $\gg$	$\exists v\colon x\text{'}v = y\text{'}u$	]*

[The Map proof of z-ElimEqualX reads

L1:	Hypothesis $\gg$	$\ell\text{'}u$	;
L2:	Hypothesis $\gg$	$x = y$	;
L3:	z-ElimEqualXE $\unrhd$ L1 $\unrhd$ L2 $\gg$	$\exists v\colon x\text{'}v = y\text{'}u$	;
L4:	ElimExists $\unrhd$ L3 $\gg$	$x\text{'}v_x(x, y, u) = y\text{'}u$	]*

[The Map proof of z-ElimEqualXEll reads

L1:	Hypothesis $\gg$	$\ell\text{'}u$	;
L2:	Hypothesis $\gg$	$x = y$	;
L3:	z-ElimEqualXE $\unrhd$ L1 $\unrhd$ L2 $\gg$	$\exists v\colon x\text{'}v = y\text{'}u$	;
L4:	ElimExistsEll $\unrhd$ L3 $\gg$	$\ell\text{'}v_x(x, y, u)$	]*

[The Map proof of z-ElimEqualYE reads

L01:	Hypothesis $\gg$	$\ell\text{'}u$	;
L02:	Hypothesis $\gg$	$x = y$	;
L03:	SubtypeLB $\unrhd$ L2 $\gg$	$!x$	;
L04:	Logic $\unrhd$ L2 $\gg$	$x \rightarrow \exists v\colon x\text{'}u = y\text{'}v$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	$\neg x$	;
L07:	Logic $\unrhd$ L2 $\unrhd$ L6 $\gg$	$\forall u \exists v\colon x\text{'}u = y\text{'}v$	;
L08:	ElimAll $\unrhd$ L7 $\unrhd$ L1 $\gg$	$\exists v\colon x\text{'}u = y\text{'}v$	;
L09:	Block $\gg$	End	;
L10:	TND $\rhd$ L3 $\rhd$ L4 $\rhd$ L8 $\gg$	$\exists v\colon x\text{'}u = y\text{'}v$	]*

[The Map proof of z-ElimEqualY reads

L1:	Hypothesis $\gg$	$\ell\text{'}u$	;
L2:	Hypothesis $\gg$	$x = y$	;
L3:	z-ElimEqualYE $\unrhd$ L1 $\unrhd$ L2 $\gg$	$\exists v\colon x\text{'}u = y\text{'}v$	;
L4:	ElimExists $\unrhd$ L3 $\gg$	$x\text{'}u = y\text{'}v_y(x, y, u)$	]*

[The Map proof of z-ElimEqualYEll reads

L1:	Hypothesis $\gg$	$\ell\text{'}u$	;
L2:	Hypothesis $\gg$	$x = y$	;
L3:	z-ElimEqualYE $\unrhd$ L1 $\unrhd$ L2 $\gg$	$\exists v\colon x\text{'}u = y\text{'}v$	;
L4:	ElimExistsEll $\unrhd$ L3 $\gg$	$\ell\text{'}v_y(x, y, u)$	]*

[The Map proof of z-IntroEqual reads

L01:	Premise $\gg$	$\ell\,\text{'}\,a$	;
L02:	Premise $\gg$	$\ell\,\text{'}\,b$	;
L03:	Premise $\gg$	$a \Leftrightarrow b$	;
L04:	Premise $\gg$	$\ell\,\text{'}\,u \rightarrow a\,\text{'}\,u = b\,\text{'}\,r$	;
L05:	Premise $\gg$	$\ell\,\text{'}\,u \rightarrow \ell\,\text{'}\,r$	;
L06:	Premise $\gg$	$\ell\,\text{'}\,u \rightarrow a\,\text{'}\,s = b\,\text{'}\,u$	;
L07:	Premise $\gg$	$\ell\,\text{'}\,u \rightarrow \ell\,\text{'}\,s$	;
L08:	SubtypeLB $\trianglerighteq$ L1 $\gg$	$!a$	;
L09:	Logic $\trianglerighteq$ L3 $\gg$	$a \rightarrow a = b$	;
L10:	Block $\gg$	Begin	;
L11:	Hypothesis $\gg$	$\quad \neg a$	;
L12:	Logic $\trianglerighteq$ L3 $\trianglerighteq$ L11 $\gg$	$\quad \neg b$	;
L13:	Block $\gg$	$\quad$ Begin	;
L14:	Hypothesis $\gg$	$\quad\quad \ell\,\text{'}\,u$	;
L15:	Logic $\trianglerighteq$ L4 $\trianglerighteq$ L14 $\gg$	$\quad\quad a\,\text{'}\,u = b\,\text{'}\,r$	;
L16:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L14 $\gg$	$\quad\quad \ell\,\text{'}\,r$	;
L17:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L14 $\gg$	$\quad\quad a\,\text{'}\,s = b\,\text{'}\,u$	;
L18:	Logic $\trianglerighteq$ L7 $\trianglerighteq$ L14 $\gg$	$\quad\quad \ell\,\text{'}\,s$	;
L19:	Block $\gg$	$\quad\quad$ Begin	;
L20:	Hypothesis $\gg$	$\quad\quad\quad \ell\,\text{'}\,v$	;
L21:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L14 $\gg$	$\quad\quad\quad \ell\,\text{'}(a\,\text{'}\,u)$	;
L22:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L20 $\gg$	$\quad\quad\quad \ell\,\text{'}(b\,\text{'}\,v)$	;
L23:	z-TypeEqual $\trianglerighteq$ L21 $\trianglerighteq$ L22 $\gg$	$\quad\quad\quad !a\,\text{'}\,u = b\,\text{'}\,v$	;
L24:	Block $\gg$	$\quad\quad$ End	;
L25:	IntroExists $\trianglerighteq$ (Gen $\triangleright$ L23) $\trianglerighteq$ L16 $\trianglerighteq$ L15 $\gg$	$\quad\quad \exists v\colon a\,\text{'}\,u = b\,\text{'}\,v$	;
L26:	Block $\gg$	$\quad\quad$ Begin	;
L27:	Hypothesis $\gg$	$\quad\quad\quad \ell\,\text{'}\,v$	;
L28:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L27 $\gg$	$\quad\quad\quad \ell\,\text{'}(a\,\text{'}\,v)$	;
L29:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L14 $\gg$	$\quad\quad\quad \ell\,\text{'}(b\,\text{'}\,u)$	;
L30:	z-TypeEqual $\trianglerighteq$ L28 $\trianglerighteq$ L29 $\gg$	$\quad\quad\quad !a\,\text{'}\,u = b\,\text{'}\,v$	;
L31:	Block $\gg$	$\quad\quad$ End	;
L32:	IntroExists $\trianglerighteq$ (Gen $\triangleright$ L30) $\trianglerighteq$ L18 $\trianglerighteq$ L17 $\gg$	$\quad\quad \exists v\colon a\,\text{'}\,v = b\,\text{'}\,u$	;
L33:	Block $\gg$	$\quad$ End	;
L34:	Logic$\trianglerighteq$L11$\trianglerighteq$L12$\trianglerighteq$(Gen$\triangleright$L25)$\trianglerighteq$(Gen$\triangleright$L32)$\gg$	$\quad a = b$	;
L35:	Block $\gg$	End	;
L36:	TND $\triangleright$ L8 $\triangleright$ L9 $\triangleright$ L34 $\gg$	$a = b$	]*

[The Map proof of z-RefEqual reads

L01:	Reduction $\gg$	$\mathsf{T} = \mathsf{T}$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\quad \neg x$	;
L04:	Hypothesis $\gg$	$\quad \ell \,\text{'}\, x$	;
L05:	Hypothesis $\gg$	$\quad \forall u\colon x \,\text{'}\, u = x \,\text{'}\, u$	;
L06:	Logic $\trianglerighteq$ L3 $\gg$	$\quad x \dot{\Leftrightarrow} x$	;
L07:	Block $\gg$	$\quad$ Begin	;
L08:	Hypothesis $\gg$	$\qquad \ell \,\text{'}\, u$	;
L09:	ElimAll $\trianglerighteq$ L5 $\trianglerighteq$ L5 $\gg$	$\qquad x \,\text{'}\, u = x \,\text{'}\, u$	;
L10:	Block $\gg$	$\quad$ End	;
L11:	z-IntroEqual$\triangleright$L4$\triangleright$L4$\triangleright$L6$\triangleright$L9$\triangleright$L8$\triangleright$L9$\triangleright$L8$\gg$	$\quad x = x$	;
L12:	Block $\gg$	End	;
L13:	Transfinite' $\triangleright$ L1 $\triangleright$ L11 $\gg$	$\forall x\colon x = x$	;
L14:	ElimAll $\trianglerighteq$ L13 $\gg$	$\ell x\colon x = x$	]*

[The Map proof of z-RefEqual' reads

L01:	Hypothesis $\gg$	$x \sim y$	;
L02:	StrictEquivKX $\trianglerighteq$ L1 $\gg$	$\ell \,\text{'}\, x$	;
L03:	z-RefEqual $\trianglerighteq$ L2 $\gg$	$x = x$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\quad \ell \,\text{'}\, z$	;
L06:	z-TypeEqual $\trianglerighteq$ L2 $\trianglerighteq$ L5 $\gg$	$\quad !x = z$	;
L07:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L6 $\gg$	$\quad !(\lambda z.x = z) \,\text{'}\, z$	;
L08:	Block $\gg$	End	;
L09:	Gen $\triangleright$ L7 $\gg$	$!_1 \lambda z.x = z$	;
L10:	ExtBool $\trianglerighteq$ L9 $\trianglerighteq$ L1 $\gg$	$(\lambda z.x = z) \,\text{'}\, x \dot{\Leftrightarrow} (\lambda z.x = z) \,\text{'}\, y$	;
L11:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L10 $\gg$	$x = x \dot{\Leftrightarrow} x = y$	;
L12:	Logic $\triangleright$ L11 $\triangleright$ L3 $\gg$	$x = y$	]*

[Map lemma

z-ComEqual1$_{225}$:	$\forall y\colon (\mathsf{T} = y \dot{\Rightarrow} y = \mathsf{T})$	;
z-ComEqual2$_{225}$:	$\neg x \to \ell \,\text{'}\, x \to \forall z\colon \forall y\colon (x \,\text{'}\, z = y \dot{\Rightarrow} y = x \,\text{'}\, z) \to$	
	$\forall y\colon (x = y \dot{\Rightarrow} y = x)$	]*

[The Map proof of z-ComEqual1 reads

L1:	Block $\gg$	Begin	;
L2:	Hypothesis $\gg$	$\quad \ell \,\text{'}\, y$	;
L3:	TypeT $\gg$	$\quad \ell \,\text{'}\, \mathsf{T}$	;
L4:	z-TypeEqual $\trianglerighteq$ L3 $\trianglerighteq$ L2 $\gg$	$\quad !\mathsf{T} = y$	;
L5:	Logic $\gg$	$\quad \mathsf{T} = y \to y = \mathsf{T}$	;
L6:	Deduction" $\triangleright$ L4 $\triangleright$ L5 $\gg$	$\quad \mathsf{T} = y \dot{\Rightarrow} y = \mathsf{T}$	;
L7:	Block $\gg$	End	;
L8:	Gen $\triangleright$ L6 $\gg$	$\forall y\colon (\mathsf{T} = y \dot{\Rightarrow} y = \mathsf{T})$	]*

[The Map proof of z-ComEqual2 reads

L01:	Hypothesis $\gg$	$\neg x$	;
L02:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L03:	Hypothesis $\gg$	$\forall z\colon\forall y\colon(x\,\text{'}\,z = y \mathrel{\dot{\Rightarrow}} y = x\,\text{'}\,z)$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\quad \ell\,\text{'}\,y$	;
L06:	z-TypeEqual $\trianglerighteq$ L2 $\trianglerighteq$ L5 $\gg$	$\quad !x = y$	;
L07:	Block $\gg$	$\quad$ Begin	;
L08:	Hypothesis $\gg$	$\quad\quad x = y$	;
L09:	Logic $\trianglerighteq$ L8 $\gg$	$\quad\quad x \mathrel{\dot{\Leftrightarrow}} y$	;
L10:	Block $\gg$	$\quad\quad$ Begin	;
L11:	Hypothesis $\gg$	$\quad\quad\quad \ell\,\text{'}\,u$	;
L12:	z-ElimEqualX $\trianglerighteq$ L11 $\trianglerighteq$ L8 $\gg$	$\quad\quad\quad x\,\text{'}\,v_x(x,y,u) = y\,\text{'}\,u$	;
L13:	z-ElimEqualXEll $\trianglerighteq$ L11 $\trianglerighteq$ L8 $\gg$	$\quad\quad\quad \ell\,\text{'}\,v_x(x,y,u)$	;
L14:	z-ElimEqualY $\trianglerighteq$ L11 $\trianglerighteq$ L8 $\gg$	$\quad\quad\quad x\,\text{'}\,u = y\,\text{'}\,v_y(x,y,u)$	;
L15:	z-ElimEqualYEll $\trianglerighteq$ L11 $\trianglerighteq$ L8 $\gg$	$\quad\quad\quad \ell\,\text{'}\,v_y(x,y,u)$	;
L16:	TypeApply $\trianglerighteq$ L5 $\trianglerighteq$ L11 $\gg$	$\quad\quad\quad \ell\,\text{'}(y\,\text{'}\,u)$	;
L17:	ElimAll2 $\trianglerighteq$ L3 $\trianglerighteq$ L13 $\trianglerighteq$ L16 $\gg$	$\quad\quad\quad x\,\text{'}\,v_x(x,y,u) = y\,\text{'}\,u \mathrel{\dot{\Rightarrow}}$ $\quad\quad\quad y\,\text{'}\,u = x\,\text{'}\,v_x(x,y,u)$	;
L18:	Logic $\trianglerighteq$ L12 $\trianglerighteq$ L17 $\gg$	$\quad\quad\quad y\,\text{'}\,u = x\,\text{'}\,v_x(x,y,u)$	;
L19:	TypeApply $\trianglerighteq$ L5 $\trianglerighteq$ L15 $\gg$	$\quad\quad\quad \ell\,\text{'}(y\,\text{'}\,v_y(x,y,u))$	;
L20:	ElimAll2 $\trianglerighteq$ L3 $\trianglerighteq$ L11 $\trianglerighteq$ L19 $\gg$	$\quad\quad\quad x\,\text{'}\,u = y\,\text{'}\,v_y(x,y,u) \mathrel{\dot{\Rightarrow}}$ $\quad\quad\quad y\,\text{'}\,v_y(x,y,u) = x\,\text{'}\,u$	;
L21:	Logic $\trianglerighteq$ L14 $\trianglerighteq$ L20 $\gg$	$\quad\quad\quad y\,\text{'}\,v_y(x,y,u) = x\,\text{'}\,u$	;
L22:	Block $\gg$	$\quad\quad$ End	;
L23:	z-IntroEqual $\triangleright$ L5 $\triangleright$ L2 $\triangleright$ L9 $\triangleright$ L18 $\triangleright$ L13 $\triangleright$ L21 $\triangleright$ L15 $\gg$	$\quad\quad y = x$	;
L24:	Block $\gg$	$\quad$ End	;
L25:	Deduction" $\triangleright$ L6 $\triangleright$ L23 $\gg$	$\quad x = y \mathrel{\dot{\Rightarrow}} y = x$	;
L26:	Block $\gg$	End	;
L27:	Gen $\triangleright$ L25 $\gg$	$\forall y\colon(x = y \mathrel{\dot{\Rightarrow}} y = x)$	]*

[The Map proof of z-ComEqual reads

L1:	Transfinite' $\triangleright$ z-ComEqual1 $\triangleright$ z-ComEqual2 $\gg$	$\forall x\colon\forall y\colon(x = y \mathrel{\dot{\Rightarrow}} y = x)$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad \ell\,\text{'}\,x$	;
L4:	Hypothesis $\gg$	$\quad \ell\,\text{'}\,y$	;
L5:	Hypothesis $\gg$	$\quad x = y$	;
L6:	ElimAll2 $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$\quad x = y \mathrel{\dot{\Rightarrow}} y = x$	;
L7:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$\quad y = x$	;
L8:	Block $\gg$	End	;
L9:	Repetition $\triangleright$ L7 $\gg$	$\ell x,y\colon x = y \to y = x$	]*

[Map lemma

z-TransEqual1$_{227}$:	$\forall y{:}\,\forall z{:}\,(\mathsf{T} = y \mathbin{\dot\wedge} y = z \mathbin{\dot\Rightarrow} \mathsf{T} = z)$	;
z-TransEqual2$_{227}$:	$\neg x \rightarrow \ell\,{}'\,x \rightarrow$	
	$\forall u{:}\,\forall y{:}\,\forall z{:}\,(x\,{}'\,u = y \mathbin{\dot\wedge} y = z \mathbin{\dot\Rightarrow} x\,{}'\,u = z) \rightarrow$	
	$\forall y{:}\,\forall z{:}\,(x = y \mathbin{\dot\wedge} y = z \mathbin{\dot\Rightarrow} x = z)$	]*

[The Map proof of z-TransEqual1 reads

L01:	Block $\gg$	Begin	;
L02:	Hypothesis $\gg$	$\ell\,{}'\,y$	;
L03:	Hypothesis $\gg$	$\ell\,{}'\,z$	;
L04:	TypeT $\gg$	$\ell\,{}'\,\mathsf{T}$	;
L05:	z-TypeEqual $\trianglerighteq$ L4 $\trianglerighteq$ L2 $\gg$	$!\mathsf{T} = y$	;
L06:	z-TypeEqual $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$!y = z$	;
L07:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$!(\mathsf{T} = y \mathbin{\dot\wedge} y = z)$	;
L08:	Logic $\gg$	$\mathsf{T} = y \mathbin{\dot\wedge} y = z \rightarrow \mathsf{T} = z$	;
L09:	Deduction” $\triangleright$ L7 $\triangleright$ L8 $\gg$	$\mathsf{T} = y \mathbin{\dot\wedge} y = z \mathbin{\dot\Rightarrow} \mathsf{T} = z$	;
L10:	Block $\gg$	End	;
L11:	Gen2 $\triangleright$ L9 $\gg$	$\forall y{:}\,\forall z{:}\,(\mathsf{T} = y \mathbin{\dot\wedge} y = z \mathbin{\dot\Rightarrow} \mathsf{T} = z)$	]*

[The Map proof of z-TransEqual2 reads

L01:	Hypothesis $\gg$	$\neg x$	;
L02:	Hypothesis $\gg$	$\ell\,'\,x$	;
L03:	Hypothesis $\gg$	$\forall u\colon \forall y\colon \forall z\colon$ $(x\,'\,u = y \mathrel{\dot\wedge} y = z \mathrel{\dot\Rightarrow} x\,'\,u = z)$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\ell\,'\,y$	;
L06:	Hypothesis $\gg$	$\ell\,'\,z$	;
L07:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L5 $\gg$	$!x = y$	;
L08:	TypeApply $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$!y = z$	;
L09:	Logic $\trianglerighteq$ L7 $\trianglerighteq$ L8 $\gg$	$!(x = y \mathrel{\dot\wedge} y = z)$	;
L10:	Block $\gg$	Begin	;
L11:	Hypothesis $\gg$	$x = y \mathrel{\dot\wedge} y = z$	;
L12:	Logic $\trianglerighteq$ L11 $\gg$	$x = y$	;
L13:	Logic $\trianglerighteq$ L11 $\gg$	$y = z$	;
L14:	Logic $\trianglerighteq$ L11 $\gg$	$x \mathrel{\dot\Leftrightarrow} z$	;
L15:	Block $\gg$	Begin	;
L16:	Hypothesis $\gg$	$\ell\,'\,u$	;
L17:	Define $\gg$	$p \equiv v_y(x,y,u)$	;
L18:	Define $\gg$	$q \equiv v_y(y,z,p)$	;
L19:	Define $\gg$	$r \equiv v_x(y,z,u)$	;
L20:	Define $\gg$	$s \equiv v_x(x,y,r)$	;
L21:	z-ElimEqualY$\trianglerighteq$L12$\trianglerighteq$L16$\gg$	$x\,'\,u = y\,'\,p$	;
L22:	z-ElimEqualYEll$\trianglerighteq$L12$\trianglerighteq$L16$\gg$	$\ell\,'\,p$	;
L23:	z-ElimEqualY$\trianglerighteq$L13$\trianglerighteq$L22$\gg$	$y\,'\,p = z\,'\,q$	;
L24:	z-ElimEqualYEll$\trianglerighteq$L13$\trianglerighteq$L22$\gg$	$\ell\,'\,q$	;
L25:	z-ElimEqualX$\trianglerighteq$L13$\trianglerighteq$L16$\gg$	$y\,'\,r = z\,'\,u$	;
L26:	z-ElimEqualXEll$\trianglerighteq$L13$\trianglerighteq$L16$\gg$	$\ell\,'\,r$	;
L27:	z-ElimEqualX$\trianglerighteq$L12$\trianglerighteq$L26$\gg$	$x\,'\,s = y\,'\,r$	;
L28:	z-ElimEqualXEll$\trianglerighteq$L12$\trianglerighteq$L26$\gg$	$\ell\,'\,s$	;
L29:	TypeApply $\trianglerighteq$ L5 $\trianglerighteq$ L22 $\gg$	$\ell\,'(y\,'\,p)$	;
L30:	TypeApply $\trianglerighteq$ L6 $\trianglerighteq$ L24 $\gg$	$\ell\,'(z\,'\,q)$	;
L31:	ElimAll3$\trianglerighteq$L3$\trianglerighteq$L16$\trianglerighteq$L29$\trianglerighteq$L30$\gg$	$x'u = y'p \mathrel{\dot\wedge} y'p = z'q \mathrel{\dot\Rightarrow}$ $x\,'\,u = z\,'\,q$	;
L32:	Logic $\trianglerighteq$ L21 $\trianglerighteq$ L23 $\trianglerighteq$ L31 $\gg$	$x\,'\,u = z\,'\,q$	;
L33:	TypeApply $\trianglerighteq$ L5 $\trianglerighteq$ L26 $\gg$	$\ell\,'(y\,'\,r)$	;
L34:	TypeApply $\trianglerighteq$ L6 $\trianglerighteq$ L16 $\gg$	$\ell\,'(z\,'\,u)$	;
L35:	ElimAll3$\trianglerighteq$L3$\trianglerighteq$L28$\trianglerighteq$L33$\trianglerighteq$L34$\gg$	$x's = y'r \mathrel{\dot\wedge} y'r = z'u \mathrel{\dot\Rightarrow}$ $x\,'\,s = z\,'\,u$	;
L36:	Logic $\trianglerighteq$ L25 $\trianglerighteq$ L27 $\trianglerighteq$ L35 $\gg$	$x\,'\,s = z\,'\,u$	;
L37:	Block $\gg$	End	;
L38:	z-IntroEqual$\triangleright$L2$\triangleright$L6$\triangleright$L14$\triangleright$ L32 $\triangleright$ L24 $\triangleright$ L36 $\triangleright$ L28 $\gg$	$x = z$	;
L39:	Block $\gg$	End	;
L40:	Deduction" $\triangleright$ L9 $\triangleright$ L38 $\gg$	$x = y \mathrel{\dot\wedge} y = z \mathrel{\dot\Rightarrow} x = z$	;
L41:	Block $\gg$	End	;
L42:	Gen2 $\triangleright$ L40 $\gg$	$\forall y\colon \forall z\colon (x = y \mathrel{\dot\wedge} y = z \mathrel{\dot\Rightarrow} x = z)$	]*

[The Map proof of z-TransEqual reads

L01:	Transfinite' ▷		
	z-TransEqual1 ▷		
	z-TransEqual2 ≫	$\forall x\colon \forall y\colon \forall z\colon (x = y \mathrel{\dot{\wedge}} y = z \mathrel{\dot{\Rightarrow}} x = z)$	;
L02:	Block ≫	Begin	;
L03:	Hypothesis ≫	$\ell\,'\,x$	;
L04:	Hypothesis ≫	$\ell\,'\,y$	;
L05:	Hypothesis ≫	$\ell\,'\,z$	;
L06:	Hypothesis ≫	$x = y$	;
L07:	Hypothesis ≫	$y = z$	;
L08:	ElimAll3⊵L1⊵L3⊵L4⊵L5≫	$x = y \mathrel{\dot{\wedge}} y = z \mathrel{\dot{\Rightarrow}} x = z$	;
L09:	Logic ⊵ L6 ⊵ L7 ⊵ L8 ≫	$x = z$	;
L10:	Block ≫	End	;
L11:	Repetition ▷ L9 ≫	$\ell x, y, z\colon x = y \to y = z \to x = z$	]*

A.56 Lemmas about $[\,x \in y\,]$

$[\,x \in y \doteq \mathrm{if}(y, \mathsf{F}, \exists z\colon x = y\,'\,z)\,]^*$

[Map lemma

z-TypeIn$_{229}$:	$\ell x, y\colon !x \in y$	;
z-ElimIn$_{230}$:	$x \in y \to x = y\,'\,\varepsilon z\colon x = y\,'\,z$	;
z-ElimInEll$_{230}$:	$x \in y \to \ell\,'\,\varepsilon z\colon x = y\,'\,z$	;
z-ElimInNot$_{230}$:	$x \in y \to \neg y$	;
z-IntroIn$_{230}$:	$\ell x, y, z\colon \neg y \to x = y\,'\,z \to x \in y$	;
z-Member$_{230}$:	$\ell y, z\colon \neg y \to y\,'\,z \in y$	;
z-Member'$_{230}$:	$\ell y\colon \neg y \to \exists x\colon x \in y$	;
z-NotIn$_{231}$:	$\ell y\colon \forall x\colon \neg x \in y \to y$	;
z-ext1$_{231}$:	$\ell x, y, z\colon x = y \to y \in z \to x \in z$	;
z-ext2$_{231}$:	$\ell x, y, z\colon x \in y \to y = z \to x \in z$	;
z-ext3$_{232}$:	$\ell x, y\colon \forall z\colon (z \in x \mathrel{\dot{\Leftrightarrow}} z \in y) \to x = y$	;
z-q$_{233}$:	$\forall x \forall y\colon (x = y \mathrel{\dot{\Leftrightarrow}} \forall z\colon (z \in x \mathrel{\dot{\Leftrightarrow}} z \in y))$	;
z-e$_{234}$:	$\forall x \forall y \forall z\colon (x = y \Rightarrow (x \in z \Rightarrow y \in z))$	]*

[The Map proof of z-TypeIn reads

L01:	Hypothesis ≫	$\ell\,'\,x$	;
L02:	Hypothesis ≫	$\ell\,'\,y$	;
L03:	SubtypeLB ⊵ L2 ≫	$!y$	;
L04:	Logic ≫	$y \to !\mathrm{if}(y, \mathsf{F}, \exists z\colon x = y\,'\,z)$	;
L05:	Block ≫	Begin	;
L06:	Hypothesis ≫	$\ell\,'\,z$	;
L07:	TypeApply ⊵ L2 ⊵ L6 ≫	$\ell\,'(y\,'\,z)$	;
L08:	z-TypeEqual ⊵ L1 ⊵ L7 ≫	$!(x = y\,'\,z)$	;
L09:	Block ≫	End	;
L10:	TypeExists ⊵ (Gen ▷ L8) ≫	$!\exists x\colon x = y\,'\,z$	;
L11:	Logic ⊵ L10 ≫	$\neg y \to !\mathrm{if}(y, \mathsf{F}, \exists z\colon x = y\,'\,z)$	;
L12:	TND ▷ L3 ▷ L4 ▷ L11 ≫	$!x \in y$	]*

[The Map proof of z-ElimIn reads

L1:	Hypothesis $\gg$	$x \in y$	;
L2:	Logic $\trianglerighteq$ L1 $\gg$	$\exists z\colon x = y\text{'}z$	;
L3:	ElimExists $\trianglerighteq$ L2 $\gg$	$x = y\text{'}\varepsilon z\colon x = y\text{'}z$	]*

[The Map proof of z-ElimInEll reads

L1:	Hypothesis $\gg$	$x \in y$	;
L2:	Logic $\trianglerighteq$ L1 $\gg$	$\exists z\colon x = y\text{'}z$	;
L3:	ElimExistsEll $\trianglerighteq$ L2 $\gg$	$\ell\text{'}\varepsilon z\colon x = y\text{'}z$	]*

[The Map proof of z-ElimInNot reads

L1:	Logic $\gg$	$x \in y \rightarrow \neg y$	]*

[The Map proof of z-IntroIn reads

L01:	Hypothesis $\gg$	$\ell\text{'}x$	;
L02:	Hypothesis $\gg$	$\ell\text{'}y$	;
L03:	Hypothesis $\gg$	$\ell\text{'}z$	;
L04:	Hypothesis $\gg$	$\neg y$	;
L05:	Hypothesis $\gg$	$x = y\text{'}z$	;
L06:	Block $\gg$	Begin	;
L07:	Hypothesis $\gg$	$\ell\text{'}u$	;
L08:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L7 $\gg$	$\ell\text{'}(y\text{'}u)$	;
L09:	z-TypeEqual $\trianglerighteq$ L1 $\trianglerighteq$ L8 $\gg$	$!x = y\text{'}u$	;
L10:	Block $\gg$	End	;
L11:	IntroExists $\trianglerighteq$ (Gen $\triangleright$ L9) $\trianglerighteq$ L3 $\trianglerighteq$ L5 $\gg$	$\exists z\colon x = y\text{'}z$	;
L12:	Logic $\trianglerighteq$ L4 $\trianglerighteq$ L11 $\gg$	$x \in y$	]*

[The Map proof of z-Member reads

L1:	Hypothesis $\gg$	$\ell\text{'}y$	;
L2:	Hypothesis $\gg$	$\ell\text{'}z$	;
L3:	Hypothesis $\gg$	$\neg y$	;
L4:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\text{'}(y\text{'}z)$	;
L5:	z-RefEqual $\trianglerighteq$ L4 $\gg$	$y\text{'}z = y\text{'}z$	;
L6:	z-IntroIn $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\trianglerighteq$ L5 $\gg$	$y\text{'}z \in y$	]*

[The Map proof of z-Member' reads

L1:	Hypothesis $\gg$	$\ell\text{'}y$	;
L2:	Hypothesis $\gg$	$\neg y$	;
L3:	Block $\gg$	Begin	;
L4:	Hypothesis $\gg$	$\ell\text{'}x$	;
L5:	z-TypeIn $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\gg$	$!x \in y$	;
L6:	Block $\gg$	End	;
L7:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\gg$	$\ell\text{'}(y\text{'}y)$	;
L8:	z-Member $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$y\text{'}y \in y$	;
L9:	IntroExists $\trianglerighteq$ (Gen $\triangleright$ L5) $\trianglerighteq$ L7 $\trianglerighteq$ L8 $\gg$	$\exists x\colon x \in y$	]*

[The Map proof of z-NotIn reads

L01:	Hypothesis $\gg$	$\ell\,\text{'}\,y$	;
L02:	Hypothesis $\gg$	$\forall x\colon \neg x \in y$	;
L03:	SubtypeLB $\trianglerighteq$ L1 $\gg$	$!y$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\neg y$	;
L06:	z-Member $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\trianglerighteq$ L5 $\gg$	$y\,\text{'}\,y \in y$	;
L07:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}(y\,\text{'}\,y)$	;
L08:	ElimAll $\trianglerighteq$ L2 $\trianglerighteq$ L7 $\gg$	$\neg y\,\text{'}\,y \in y$	;
L09:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L8 $\gg$	F	;
L10:	Block $\gg$	End	;
L11:	Indirect $\triangleright$ L3 $\triangleright$ L9 $\gg$	y	]*

[The Map proof of z-ext1 reads

L01:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L02:	Hypothesis $\gg$	$\ell\,\text{'}\,y$	;
L03:	Hypothesis $\gg$	$\ell\,\text{'}\,z$	;
L04:	Hypothesis $\gg$	$x = y$	;
L05:	Hypothesis $\gg$	$y \in z$	;
L06:	Define $\gg$	$u \equiv \varepsilon u\colon y = z\,\text{'}\,u$	;
L07:	z-ElimInNot $\trianglerighteq$ L5 $\gg$	$\neg z$	;
L08:	z-ElimInEll $\trianglerighteq$ L5 $\gg$	$\ell\,\text{'}\,u$	;
L09:	z-ElimIn $\trianglerighteq$ L5 $\gg$	$y = z\,\text{'}\,u$	;
L10:	TypeApply $\trianglerighteq$ L3 $\trianglerighteq$ L8 $\gg$	$\ell\,\text{'}(z\,\text{'}\,u)$	;
L11:	z-TransEqual $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L10 $\trianglerighteq$ L4 $\trianglerighteq$ L9 $\gg$	$x = z\,\text{'}\,u$	;
L12:	z-IntroIn $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\trianglerighteq$ L8 $\trianglerighteq$ L7 $\trianglerighteq$ L11 $\gg$	$x \in z$	]*

[The Map proof of z-ext2 reads

L01:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L02:	Hypothesis $\gg$	$\ell\,\text{'}\,y$	;
L03:	Hypothesis $\gg$	$\ell\,\text{'}\,z$	;
L04:	Hypothesis $\gg$	$x \in y$	;
L05:	Hypothesis $\gg$	$y = z$	;
L06:	Define $\gg$	$u \equiv \varepsilon u\colon x = y\,\text{'}\,u$	;
L07:	z-ElimInNot $\trianglerighteq$ L4 $\gg$	$\neg y$	;
L08:	z-ElimInEll $\trianglerighteq$ L4 $\gg$	$\ell\,\text{'}\,u$	;
L09:	z-ElimIn $\trianglerighteq$ L4 $\gg$	$x = y\,\text{'}\,u$	;
L10:	Define $\gg$	$v \equiv v_y(y, z, u)$	;
L11:	z-ElimEqualYEll $\trianglerighteq$ L8 $\trianglerighteq$ L5 $\gg$	$\ell\,\text{'}\,v$	;
L12:	z-ElimEqualY $\trianglerighteq$ L8 $\trianglerighteq$ L5 $\gg$	$y\,\text{'}\,u = z\,\text{'}\,v$	;
L13:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L8 $\gg$	$\ell\,\text{'}(y\,\text{'}\,u)$	;
L14:	TypeApply $\trianglerighteq$ L3 $\trianglerighteq$ L11 $\gg$	$\ell\,\text{'}(z\,\text{'}\,v)$	;
L15:	z-TransEqual$\trianglerighteq$L1$\trianglerighteq$L13$\trianglerighteq$L14$\trianglerighteq$L9$\trianglerighteq$L12$\gg$	$x = z\,\text{'}\,v$	;
L16:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L7 $\gg$	$\neg z$	;
L17:	z-IntroIn $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\trianglerighteq$ L11 $\trianglerighteq$ L16 $\trianglerighteq$ L15 $\gg$	$x \in z$	]*

[The Map proof of z-ext3 reads

```
L01: Hypothesis ≫                          $\ell\,'\,x$                                   ;
L02: Hypothesis ≫                          $\ell\,'\,y$                                   ;
L03: Hypothesis ≫                          $\forall z\colon (z \in x \dot\Leftrightarrow z \in y)$        ;
L04: SubtypeLB ⊵ L1 ≫                      $!x$                                           ;
L05: Block ≫                               Begin                                          ;
L06: Hypothesis ≫                            $x$                                          ;
L07: Block ≫                                 Begin                                        ;
L08: Hypothesis ≫                              $\ell\,'\,z$                               ;
L09: ElimAll ⊵ L3 ⊵ L8 ≫                       $z \in x \dot\Leftrightarrow z \in y$      ;
L10: Logic ⊵ L6 ⊵ L9 ≫                         $\neg z \in y$                             ;
L11: Block ≫                                 End                                          ;
L12: z-NotIn ⊵ L2 ⊵ (Gen ▷ L10) ≫            $y$                                          ;
L13: Logic ⊵ L6 ⊵ L12 ≫                      $x = y$                                      ;
L14: Block ≫                               End                                            ;
L15: Block ≫                               Begin                                          ;
L16: Hypothesis ≫                            $\neg x$                                     ;
L17: TypeApply ⊵ L1 ⊵ TypeT ≫                $\ell\,'(x\,'\,\mathsf{T})$                  ;
L18: ElimAll ⊵ L3 ⊵ TypeT ≫                  $x\,'\,\mathsf{T} \in x \dot\Leftrightarrow x\,'\,\mathsf{T} \in y$ ;
L19: z-Member ⊵ L1 ⊵ L16 ≫                   $x\,'\,\mathsf{T} \in x$                     ;
L20: Logic ⊵ L18 ⊵ L19 ≫                     $x\,'\,\mathsf{T} \in y$                     ;
L21: z-ElimInNot ⊵ L20 ≫                     $\neg y$                                     ;
L22: Logic ⊵ L16 ⊵ L21 ≫                     $x \dot\Leftrightarrow y$                    ;
L23: Block ≫                                 Begin                                        ;
L24: Hypothesis ≫                              $\ell\,'\,u$                               ;
L25: TypeApply ⊵ L1 ⊵ L24 ≫                    $\ell\,'(x\,'\,u)$                         ;
L26: ElimAll ⊵ L3 ⊵ L25 ≫                      $x\,'\,u \in x \dot\Leftrightarrow x\,'\,u \in y$ ;
L27: z-Member ⊵ L1 ⊵ L24 ⊵ L16 ≫               $x\,'\,u \in x$                            ;
L28: Logic ⊵ L26 ⊵ L27 ≫                       $x\,'\,u \in y$                            ;
L29: Define ≫                                  $w \equiv \varepsilon w.x\,'\,u = y\,'\,w$ ;
L30: z-ElimIn ⊵ L28 ≫                          $x\,'\,u = y\,'\,w$                        ;
L31: z-ElimInEll ⊵ L28 ≫                       $\ell\,'\,w$                               ;
L32: TypeAll ⊵ L2 ⊵ L24 ≫                      $\ell\,'(y\,'\,u)$                         ;
L33: ElimAll ⊵ L3 ⊵ L32 ≫                      $y\,'\,u \in x \dot\Leftrightarrow y\,'\,u \in y$ ;
L34: z-Member ⊵ L2 ⊵ L24 ⊵ L21 ≫               $y\,'\,u \in y$                            ;
L35: Logic ⊵ L33 ⊵ L34 ≫                       $y\,'\,u \in x$                            ;
L36: Define ≫                                  $v \equiv \varepsilon v.x\,'\,v = y\,'\,u$ ;
L37: z-ElimIn ⊵ L35 ≫                          $y\,'\,u = x\,'\,v$                        ;
L38: z-ElimInEll ⊵ L35 ≫                       $\ell\,'\,v$                               ;
L39: TypeApply ⊵ L1 ⊵ L38 ≫                    $\ell\,'(x\,'\,v)$                         ;
L40: z-ComEqual⊵L32⊵L39⊵L37≫                   $x\,'\,v = y\,'\,u$                        ;
L41: Block ≫                                 End                                          ;
L42: z-IntroEqual ⊵ L1 ⊵ L2 ⊵ L22 ⊵
     L30 ⊵ L31 ⊵ L40 ⊵ L38 ≫                 $x = y$                                      ;
L43: Block ≫                               End                                            ;
L44: TND ⊵ L4 ⊵ L13 ⊵ L42 ≫                $x = y$                                        ]*
```

[The Map proof of z-q reads

```
L01: Block ≫                          Begin                              ;
L02: Hypothesis ≫                       ℓ’x                              ;
L03: Hypothesis ≫                       ℓ’y                              ;
L04: z-TypeEqual ⊵ L2 ⊵ L3 ≫            !x = y                           ;
L05: Block ≫                            Begin                            ;
L06: Hypothesis ≫                         ℓ’z                            ;
L07: z-TypeIn ⊵ L6 ⊵ L2 ≫                 !z ∈ x                         ;
L08: z-TypeIn ⊵ L6 ⊵ L3 ≫                 !z ∈ y                         ;
L09: Logic ⊵ L7 ⊵ L8 ≫                    !(z ∈ x ⇔ z ∈ y)               ;
L10: Block ≫                            End                              ;
L11: TypeAll ⊵ (Gen ▷ L9) ≫             !∀z: (z ∈ x ⇔ z ∈ y)             ;
L12: Block ≫                            Begin                            ;
L13: Hypothesis ≫                         x = y                          ;
L14: z-ComEqual ⊵ L2 ⊵ L3 ⊵ L13 ≫         y = x                          ;
L15: Block ≫                              Begin                          ;
L16: Hypothesis ≫                           ℓ’z                          ;
L17: z-TypeIn ⊵ L16 ⊵ L2 ≫                  !z ∈ x                       ;
L18: z-TypeIn ⊵ L16 ⊵ L3 ≫                  !z ∈ y                       ;
L19: Block ≫                                Begin                        ;
L20: Hypothesis ≫                             z ∈ x                      ;
L21: z-ext2⊵L16⊵L2⊵L3⊵L20⊵L13≫                z ∈ y                      ;
L22: Block ≫                                End                          ;
L23: CDeduction▷L17▷L18▷L21≫                z ∈ x ⇒ z ∈ y                ;
L24: Block ≫                                Begin                        ;
L25: Hypothesis ≫                             z ∈ y                      ;
L26: z-ext2⊵L16⊵L3⊵L2⊵L25⊵L14≫                z ∈ x                      ;
L27: Block ≫                                End                          ;
L28: CDeduction▷L18▷L17▷L26≫                z ∈ y ⇒ z ∈ x                ;
L29: Logic ⊵ L23 ⊵ L28 ≫                    z ∈ x ⇔ z ∈ y                ;
L30: Block ≫                              End                            ;
L31: Gen ▷ L29 ≫                          ∀z: (z ∈ x ⇔ z ∈ y)            ;
L32: Block ≫                            End                              ;
L33: CDeduction ▷ L4 ▷ L11 ▷ L31 ≫      x=y ⇒ ∀z: (z∈x ⇔ z∈y)            ;
L34: z-ext3 ⊵ L2 ⊵ L3 ≫                 ∀z: (z∈x ⇔ z∈y) → x=y            ;
L35: CDeduction ▷ L11 ▷ L4 ▷ L34 ≫      ∀z: (z∈x ⇔ z∈y) ⇒ x=y            ;
L36: Logic ⊵ L33 ⊵ L35 ≫                x=y ⇔ ∀z: (z∈x ⇔ z∈y)            ;
L37: Block ≫                          End                                ;
L38: Gen2 ▷ L36 ≫                     ∀x∀y:(x=y⇔∀z:(z∈x⇔z∈y))            ]*
```

[The Map proof of z-e reads

L01:	Block $\gg$	Begin	;
L02:	Hypothesis $\gg$	$\ell\,'\,x$	;
L03:	Hypothesis $\gg$	$\ell\,'\,y$	;
L04:	Hypothesis $\gg$	$\ell\,'\,z$	;
L05:	z-TypeEqual $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$!x = y$	;
L06:	z-TypeIn $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\gg$	$!x \in z$	;
L07:	z-TypeIn $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$!y \in z$	;
L08:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$!(x \in z \Rightarrow y \in z)$	;
L09:	Block $\gg$	Begin	;
L10:	Hypothesis $\gg$	$x = y$	;
L11:	z-ComEqual $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\trianglerighteq$ L10 $\gg$	$y = x$	;
L12:	Block $\gg$	Begin	;
L13:	Hypothesis $\gg$	$x \in z$	;
L14:	z-ext1$\trianglerighteq$L3$\trianglerighteq$L2$\trianglerighteq$L4$\trianglerighteq$L11$\trianglerighteq$L13$\gg$	$y \in z$	;
L15:	Block $\gg$	End	;
L16:	CDeduction $\triangleright$ L6 $\triangleright$ L7 $\triangleright$ L14 $\gg$	$x \in z \Rightarrow y \in z$	;
L17:	Block $\gg$	End	;
L18:	CDeduction $\triangleright$ L5 $\triangleright$ L8 $\triangleright$ L16 $\gg$	$x = y \Rightarrow (x \in z \Rightarrow y \in z)$	;
L19:	Block $\gg$	End	;
L20:	Gen3 $\triangleright$ L18 $\gg$	$\forall x \forall y \forall z\colon (x{=}y{\Rightarrow}(x{\in}z{\Rightarrow}y{\in}z))$	]*

A.57 Lemmas about [ZfcExt(x)]

[Map lemma

z-IntroZfcExt'$_{234}$:	$\ell u, v\colon u = v \to \ell\,'(f\,'\,u) \vdash$	
	$\ell u, v\colon u = v \to f\,'\,u = f\,'\,v \vdash \mathrm{ZfcExt}'(f)$	;
z-ElimZfcExt'$_{235}$:	$\ell u, v\colon \mathrm{ZfcExt}'(f) \to u = v \to f\,'\,u = f\,'\,v$	;
z-SubtypeExtL'$_{235}$:	$\ell x\colon \mathrm{ZfcExt}'(f) \to \ell\,'(f\,'\,x)$	;
z-SubtypeExtL1'$_{235}$:	$\mathrm{ZfcExt}'(f) \to \ell_1 f$	;
z-IntroZfcExtIff$_{235}$:	$\ell u, v\colon u = v \to f\,'\,u \Leftrightarrow f\,'\,v \vdash \mathrm{ZfcExt}(f)$	;
z-IntroZfcExt$_{236}$:	$\ell u, v\colon u = v \to !f\,'\,u \vdash$	
	$\ell u, v\colon u = v \to f\,'\,u \to f\,'\,v \vdash \mathrm{ZfcExt}(f)$	;
z-ElimZfcExt$_{236}$:	$\ell u, v\colon \mathrm{ZfcExt}(f) \to u = v \to f\,'\,u \Leftrightarrow f\,'\,v$	;
z-SubtypeExtB$_{236}$:	$\ell x\colon \mathrm{ZfcExt}(f) \to !f\,'\,x$	;
z-SubtypeExtB1$_{237}$:	$\mathrm{ZfcExt}(f) \to !_1 f$	;
z-IntroZfcExtIff'$_{237}$:	$\langle x, y{\in}\mathcal{V}; a, b{\in}\mathcal{T}\rangle \mathrm{notfree}(x; a) \Vdash \ell x, y\colon \ell\,'\,a \vdash$	
	$\ell x{,}y\colon x{\in}a{\Leftrightarrow}b \vdash \ell x{,}y\colon \mathrm{ZfcExt}(\lambda y.b) \vdash \mathrm{ZfcExt}'(\lambda y.a)$	]*

[The Map proof of z-IntroZfcExt' reads

L01:	Premise ≫	$\ell u,v\colon u=v\to\ell\text{'}(f\text{'}u)$	;
L02:	Premise ≫	$\ell u,v\colon u=v\to f\text{'}u=f\text{'}v$	;
L03:	Block ≫	Begin	;
L04:	Hypothesis ≫	$\quad\ell\text{'}u$	;
L05:	Hypothesis ≫	$\quad\ell\text{'}v$	;
L06:	z-TypeEqual ⊵ L4 ⊵ L5 ≫	$\quad !u=v$	;
L07:	z-RefEqual ⊵ L4 ≫	$\quad u=u$	;
L08:	z-RefEqual ⊵ L5 ≫	$\quad v=v$	;
L09:	L1 ⊵ L4 ⊵ L4 ⊵ L7 ≫	$\quad\ell\text{'}(f\text{'}u)$	;
L10:	L1 ⊵ L5 ⊵ L5 ⊵ L8 ≫	$\quad\ell\text{'}(f\text{'}v)$	;
L11:	z-TypeEqual ⊵ L9 ⊵ L10 ≫	$\quad !f\text{'}u=f\text{'}v$	;
L12:	Logic ⊵ L9 ⊵ L11 ≫	$\quad !(\ell\text{'}(f\text{'}u)\mathrel{\dot\wedge}f\text{'}u=f\text{'}v)$	;
L13:	Block ≫	$\quad$Begin	;
L14:	Hypothesis ≫	$\qquad u=v$	;
L15:	L2 ⊵ L4 ⊵ L5 ⊵ L14 ≫	$\qquad f\text{'}u=f\text{'}v$	;
L16:	Logic ⊵ L9 ⊵ L15 ≫	$\qquad\ell\text{'}(f\text{'}u)\mathrel{\dot\wedge}f\text{'}u=f\text{'}v$	;
L17:	Block ≫	$\quad$End	;
L18:	CDeduction⊳L6⊳L11⊳L16≫	$\quad u=v\Rightarrow\ell\text{'}(f\text{'}u)\mathrel{\dot\wedge}f\text{'}u=f\text{'}v$	;
L19:	Block ≫	End	;
L20:	Gen2 ⊳ L18 ≫	$\mathrm{ZfcExt}'(f)$	]*

[The Map proof of z-ElimZfcExt’ reads

L1:	Hypothesis ≫	$\ell\text{'}u$	;
L2:	Hypothesis ≫	$\ell\text{'}v$	;
L3:	Hypothesis ≫	$\mathrm{ZfcExt}'(f)$	;
L4:	Hypothesis ≫	$u=v$	;
L5:	ElimAll2 ⊵ L3 ⊵ L1 ⊵ L2 ≫	$u=v\Rightarrow\ell\text{'}(f\text{'}u)\mathrel{\dot\wedge}f\text{'}u=f\text{'}v$	;
L6:	Logic ⊵ L4 ⊵ L5 ≫	$f\text{'}u=f\text{'}v$	]*

[The Map proof of z-SubtypeExtL’ reads

L1:	Hypothesis ≫	$\ell\text{'}x$	;
L2:	Hypothesis ≫	$\mathrm{ZfcExt}'(f)$	;
L3:	z-RefEqual ⊵ L1 ≫	$x=x$	;
L4:	ElimAll2 ⊵ L2 ⊵ L1 ⊵ L1 ≫	$x=x\Rightarrow\ell\text{'}(f\text{'}x)\mathrel{\dot\wedge}f\text{'}x=f\text{'}x$	;
L5:	Logic ⊵ L3 ⊵ L4 ≫	$\ell\text{'}(f\text{'}x)$	]*

[The Map proof of z-SubtypeExtL1’ reads

L1:	Hypothesis ≫	$\mathrm{ZfcExt}'(f)$	;
L2:	Block ≫	Begin	;
L3:	Hypothesis ≫	$\quad\ell\text{'}x$	;
L4:	z-SubtypeExtL’ ⊵ L3 ⊵ L1 ≫	$\quad\ell\text{'}(f\text{'}x)$	;
L5:	Block ≫	End	;
L6:	Gen ⊳ L4 ≫	$\ell_1 f$	]*

[The Map proof of z-IntroZfcExtIff reads

L01:	Premise $\gg$	$\ell u, v \colon u = v \rightarrow f\,'\,u \Leftrightarrow f\,'\,v$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\ell\,'\,u$	;
L04:	Hypothesis $\gg$	$\ell\,'\,v$	;
L05:	z-RefEqual $\trianglerighteq$ L3 $\gg$	$u = u$	;
L06:	z-RefEqual $\trianglerighteq$ L4 $\gg$	$v = v$	;
L07:	L1 $\trianglerighteq$ L3 $\trianglerighteq$ L3 $\trianglerighteq$ L5 $\gg$	$f\,'\,u \Leftrightarrow f\,'\,u$	;
L08:	L1 $\trianglerighteq$ L4 $\trianglerighteq$ L4 $\trianglerighteq$ L6 $\gg$	$f\,'\,v \Leftrightarrow f\,'\,v$	;
L09:	Logic $\trianglerighteq$ L7 $\gg$	$!f\,'\,u$	;
L10:	Logic $\trianglerighteq$ L8 $\gg$	$!f\,'\,v$	;
L11:	z-TypeEqual $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$!u = v$	;
L12:	Logic $\trianglerighteq$ L9 $\trianglerighteq$ L10 $\gg$	$!(f\,'\,u \Leftrightarrow f\,'\,v)$	;
L13:	L1 $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$u = v \rightarrow f\,'\,u \Leftrightarrow f\,'\,v$	;
L14:	CDeduction $\triangleright$ L11 $\triangleright$ L12 $\triangleright$ L13 $\gg$	$u = v \Rightarrow (f\,'\,u \Leftrightarrow f\,'\,v)$	;
L15:	Block $\gg$	End	;
L16:	Gen2 $\triangleright$ L14 $\gg$	ZfcExt(f)	]*

[The Map proof of z-IntroZfcExt reads

L01:	Premise $\gg$	$\ell u, v \colon u = v \rightarrow\, !f\,'\,u$	;
L02:	Premise $\gg$	$\ell u, v \colon u = v \rightarrow f\,'\,u \rightarrow f\,'\,v$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\ell\,'\,u$	;
L05:	Hypothesis $\gg$	$\ell\,'\,v$	;
L06:	Hypothesis $\gg$	$u = v$	;
L07:	z-ComEqual $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$v = u$	;
L08:	L1 $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$!f\,'\,u$	;
L09:	L1 $\trianglerighteq$ L5 $\trianglerighteq$ L4 $\trianglerighteq$ L7 $\gg$	$!f\,'\,v$	;
L10:	L2 $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$f\,'\,u \rightarrow f\,'\,v$	;
L11:	L2 $\trianglerighteq$ L5 $\trianglerighteq$ L4 $\trianglerighteq$ L7 $\gg$	$f\,'\,v \rightarrow f\,'\,u$	;
L12:	CDeduction $\triangleright$ L8 $\triangleright$ L9 $\triangleright$ L10 $\gg$	$f\,'\,u \Rightarrow f\,'\,v$	;
L13:	CDeduction $\triangleright$ L9 $\triangleright$ L8 $\triangleright$ L11 $\gg$	$f\,'\,v \Rightarrow f\,'\,u$	;
L14:	Logic $\trianglerighteq$ L12 $\trianglerighteq$ L13 $\gg$	$f\,'\,u \Leftrightarrow f\,'\,v$	;
L15:	Block $\gg$	End	;
L16:	z-IntroZfcExtIff $\triangleright$ L14 $\gg$	ZfcExt(f)	]*

[The Map proof of z-ElimZfcExt reads

L1:	Hypothesis $\gg$	$\ell\,'\,u$	;
L2:	Hypothesis $\gg$	$\ell\,'\,v$	;
L3:	Hypothesis $\gg$	ZfcExt(f)	;
L4:	Hypothesis $\gg$	$u = v$	;
L5:	ElimAll2 $\trianglerighteq$ L3 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$u = v \Rightarrow f\,'\,u \Leftrightarrow f\,'\,v$	;
L6:	Logic $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\gg$	$f\,'\,u \Leftrightarrow f\,'\,v$	]*

[The Map proof of z-SubtypeExtB reads

L1:	Hypothesis $\gg$	$\ell\,'\,x$	;
L2:	Hypothesis $\gg$	ZfcExt(f)	;
L4:	ElimAll2 $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\trianglerighteq$ L3 $\gg$	$x = x \Rightarrow (f\,'\,x \Leftrightarrow f\,'\,x)$	;
L5:	Logic $\trianglerighteq$ L4 $\gg$	$!f\,'\,x$	]*

[The Map proof of z-SubtypeExtB1 reads

L1:	Hypothesis $\gg$	$\mathrm{ZfcExt}(f)$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\ell\text{'}x$	;
L4:	z-SubtypeExtB $\trianglerighteq$ L3 $\trianglerighteq$ L1 $\gg$	$!f\text{'}x$	;
L5:	Block $\gg$	End	;
L6:	Gen $\triangleright$ L4 $\gg$	$!_1 f$	]*

[The Map proof of z-IntroZfcExtIff' reads

L01:	Premise $\gg$	$\ell x, y{:}\, \ell\text{'}a$	;
L02:	Premise $\gg$	$\ell x, y{:}\, x \in a \Leftrightarrow b$	;
L03:	Premise $\gg$	$\ell x, y{:}\, \mathrm{ZfcExt}(\lambda y.b)$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\ell\text{'}u$	;
L06:	Hypothesis $\gg$	$\ell\text{'}v$	;
L07:	Hypothesis $\gg$	$u = v$	;
L08:	Block $\gg$	Begin	;
L09:	Hypothesis $\gg$	$\ell\text{'}x$	;
L10:	L3 $\trianglerighteq$ L9 $\trianglerighteq$ L9 $\gg$	$\mathrm{ZfcExt}(\lambda y.b)$	;
L11:	z-ElimZfcExt$\trianglerighteq$		
	L5 $\trianglerighteq$ L6 $\trianglerighteq$ L10 $\trianglerighteq$ L7 $\gg$	$(\lambda y.b)\text{'}u \Leftrightarrow (\lambda y.b)\text{'}v$	;
L12:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L2 $\gg$	$\ell x, y{:}\, x{\in}(\lambda y.a)\text{'}y{\Leftrightarrow}(\lambda y.b)\text{'}y$	;
L13:	L12 $\trianglerighteq$ L9 $\trianglerighteq$ L5 $\gg$	$x{\in}(\lambda y.a)\text{'}u{\Leftrightarrow}(\lambda y.b)\text{'}u$	;
L14:	L12 $\trianglerighteq$ L9 $\trianglerighteq$ L6 $\gg$	$x{\in}(\lambda y.a)\text{'}v{\Leftrightarrow}(\lambda y.b)\text{'}v$	;
L15:	Logic $\trianglerighteq$ L11 $\trianglerighteq$ L13 $\trianglerighteq$ L14 $\gg$	$x{\in}(\lambda y.a)\text{'}u{\Leftrightarrow}x{\in}(\lambda y.a)\text{'}v$	;
L16:	Block $\gg$	End	;
L17:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L1 $\gg$	$\ell y{:}\, \ell\text{'}((\lambda y.a)\text{'}y)$	;
L18:	L17 $\trianglerighteq$ L5 $\trianglerighteq$ L5 $\gg$	$\ell\text{'}((\lambda y.a)\text{'}u)$	;
L19:	L17 $\trianglerighteq$ L6 $\trianglerighteq$ L6 $\gg$	$\ell\text{'}((\lambda y.a)\text{'}v)$	;
L20:	z-ext3$\trianglerighteq$L18$\trianglerighteq$L19$\trianglerighteq$(Gen$\triangleright$L15)$\gg$	$(\lambda y.a)\text{'}u = (\lambda y.a)\text{'}v$	;
L21:	Block $\gg$	End	;
L22:	z-IntroZfcExt' $\triangleright$ L18 $\triangleright$ L20 $\gg$	$\mathrm{ZfcExt}'(\lambda y.a)$	]*

A.58 Elementary extensionality lemmas

[Map lemma

z-ExtVarX$_{238}$:	$\ell\,'\,x \to \mathrm{ZfcExt}'(\lambda y.x)$	;
z-ExtVarY$_{238}$:	$\mathrm{ZfcExt}'(\lambda y.y)$	;
z-ExtIn$_{238}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(b) \to \mathrm{ZfcExt}(\lambda y.a\,'\,y \in b\,'\,y)$	;
z-ExtEqual$_{239}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(b) \to \mathrm{ZfcExt}(\lambda y.a\,'\,y = b\,'\,y)$	;
z-ExtNot$_{240}$:	$\mathrm{ZfcExt}(a) \to \mathrm{ZfcExt}(\lambda y.\neg a\,'\,y)$	;
z-ExtImply$_{240}$:	$\mathrm{ZfcExt}(a) \to \mathrm{ZfcExt}(b) \to \mathrm{ZfcExt}(\lambda y.a\,'\,y \Rightarrow b\,'\,y)$	;
z-ExtOr$_{241}$:	$\mathrm{ZfcExt}(a) \to \mathrm{ZfcExt}(b) \to \mathrm{ZfcExt}(\lambda y.a\,'\,y \dot{\vee} b\,'\,y)$	;
z-ExtAnd$_{241}$:	$\mathrm{ZfcExt}(a) \to \mathrm{ZfcExt}(b) \to \mathrm{ZfcExt}(\lambda y.a\,'\,y \dot{\wedge} b\,'\,y)$	;
z-ExtIff$_{241}$:	$\mathrm{ZfcExt}(a) \to \mathrm{ZfcExt}(b) \to \mathrm{ZfcExt}(\lambda y.a\,'\,y \dot{\Leftrightarrow} b\,'\,y)$	;
AckermannE$_{241}$:	$\langle x{\in}\mathcal{V}; a,b{\in}\mathcal{T}\rangle \ell x{:}\, a \dot{\Leftrightarrow} b \vdash \exists x{:}\, a \dot{\Leftrightarrow} \exists x{:}\, b$	;
z-ExtExists$_{242}$:	$\langle x,y{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \mathrm{notfree}(x;y) \Vdash$	
	$\ell\,'\,x \to \mathrm{ZfcExt}(\lambda y.a) \vdash \mathrm{ZfcExt}(\lambda y.\exists x{:}\, a)$	;
z-ExtAll$_{242}$:	$\langle x,y{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \mathrm{notfree}(x;y) \Vdash$	
	$\ell\,'\,x \to \mathrm{ZfcExt}(\lambda y.a) \vdash \mathrm{ZfcExt}(\lambda y.\forall x{:}\, a)$	;
AckermannC$_{243}$:	$\langle x{\in}\mathcal{V}; a,b{\in}\mathcal{T}\rangle \ell x{:}\, a \dot{\Leftrightarrow} b \vdash \varepsilon x{:}\, a = \varepsilon x{:}\, b$	;
z-ExtChoice'$_{243}$:	$\langle x,y{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \mathrm{notfree}(x;y) \Vdash$	
	$\ell\,'\,x \to \mathrm{ZfcExt}(\lambda y.a) \vdash \mathrm{ZfcExt}'(\lambda y.\varepsilon x{:}\, a)$	]*

[The Map proof of z-ExtVarX reads

L01:	Hypothesis $\gg$	$\ell\,'\,x$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\ell\,'\,u$	;
L04:	Hypothesis $\gg$	$\ell\,'\,v$	;
L05:	Hypothesis $\gg$	$u = v$	;
L06:	Trans $\rhd \bar{\mathcal{R}} \rhd$ L1 $\gg$	$\ell\,'((\lambda y.x)\,'\,u)$	;
L07:	z-RefEqual $\unrhd$ L1 $\gg$	$x = x$	;
L08:	Trans $\rhd \bar{\mathcal{R}} \rhd$ L7 $\gg$	$(\lambda y.x)\,'\,u = (\lambda y.x)\,'\,v$	;
L09:	Block $\gg$	End	;
L10:	z-IntroZfcExt' $\rhd$ L6 $\rhd$ L8 $\gg$	$\mathrm{ZfcExt}'(\lambda y.x)$	]*

[The Map proof of z-ExtVarY reads

L1:	Block $\gg$	Begin	;
L2:	Hypothesis $\gg$	$\ell\,'\,u$	;
L3:	Hypothesis $\gg$	$\ell\,'\,v$	;
L4:	Hypothesis $\gg$	$u = v$	;
L5:	Trans $\rhd \bar{\mathcal{R}} \rhd$ L2 $\gg$	$\ell\,'((\lambda y.y)\,'\,u)$	;
L6:	Trans $\rhd \bar{\mathcal{R}} \rhd$ L4 $\gg$	$(\lambda y.y)\,'\,u = (\lambda y.y)\,'\,v$	;
L7:	Block $\gg$	End	;
L8:	z-IntroZfcExt' $\rhd$ L5 $\rhd$ L6 $\gg$	$\mathrm{ZfcExt}'(\lambda y.y)$	]*

[The Map proof of z-ExtIn reads

```
L01:  Hypothesis $\gg$                                                                   $\mathrm{ZfcExt}'(a)$                           ;
L02:  Hypothesis $\gg$                                                                   $\mathrm{ZfcExt}'(b)$                           ;
L03:  Block $\gg$                                                                        Begin                                           ;
L04:  Hypothesis $\gg$                                                                     $\ell\,'\,u$                                  ;
L05:  Hypothesis $\gg$                                                                     $\ell\,'\,v$                                  ;
L06:  Hypothesis $\gg$                                                                     $u = v$                                       ;
L07:  z-ElimZfcExt' $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L1 $\trianglerighteq$ L6 $\gg$        $a\,'\,u = a\,'\,v$      ;
L08:  z-ElimZfcExt' $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\trianglerighteq$ L6 $\gg$        $b\,'\,u = b\,'\,v$      ;
L09:  z-SubtypeExtL' $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\gg$                     $\ell\,'(a\,'\,u)$                            ;
L10:  z-SubtypeExtL' $\trianglerighteq$ L5 $\trianglerighteq$ L1 $\gg$                     $\ell\,'(a\,'\,v)$                            ;
L11:  z-SubtypeExtL' $\trianglerighteq$ L4 $\trianglerighteq$ L2 $\gg$                     $\ell\,'(b\,'\,u)$                            ;
L12:  z-SubtypeExtL' $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\gg$                     $\ell\,'(b\,'\,v)$                            ;
L13:  z-TypeIn $\trianglerighteq$ L9 $\trianglerighteq$ L11 $\gg$                          $!a\,'\,u \in b\,'\,u$                        ;
L14:  Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L13 $\gg$                $!(\lambda y.a\,'\,y \in b\,'\,y)\,'\,u$      ;
L15:  Block $\gg$                                                                          Begin                                         ;
L16:  Hypothesis $\gg$                                                                       $(\lambda y.a\,'\,y \in b\,'\,y)\,'\,u$     ;
L17:  Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L16 $\gg$                  $a\,'\,u \in b\,'\,u$                       ;
L18:  z-ComEqual $\trianglerighteq$ L10 $\trianglerighteq$ L9 $\trianglerighteq$ L7 $\gg$    $a\,'\,v = a\,'\,u$                         ;
L19:  z-ext1 $\trianglerighteq$ L10 $\trianglerighteq$ L9 $\trianglerighteq$ L11 $\trianglerighteq$ L18 $\trianglerighteq$ L17 $\gg$    $a\,'\,v \in b\,'\,u$    ;
L20:  z-ext2 $\trianglerighteq$ L10 $\trianglerighteq$ L11 $\trianglerighteq$ L12 $\trianglerighteq$ L19 $\trianglerighteq$ L8 $\gg$    $a\,'\,v \in b\,'\,v$    ;
L21:  Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L20 $\gg$                  $(\lambda y.a\,'\,y \in b\,'\,y)\,'\,v$     ;
L22:  Block $\gg$                                                                          End                                           ;
L23:  Block $\gg$                                                                        End                                             ;
L24:  z-IntroZfcExt $\triangleright$ L14 $\triangleright$ L21 $\gg$                      $\mathrm{ZfcExt}(\lambda y.a\,'\,y \in b\,'\,y)$  ]*
```

[The Map proof of z-ExtEqual reads

L01:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(a)$	;
L02:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(b)$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\ell\,’u$	;
L05:	Hypothesis $\gg$	$\ell\,’v$	;
L06:	Hypothesis $\gg$	$u = v$	;
L07:	z-ElimZfcExt’ $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L1 $\trianglerighteq$ L6 $\gg$	$a\,’u = a\,’v$	;
L08:	z-ElimZfcExt’ $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\trianglerighteq$ L6 $\gg$	$b\,’u = b\,’v$	;
L09:	z-SubtypeExtL’ $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\gg$	$\ell\,’(a\,’u)$	;
L10:	z-SubtypeExtL’ $\trianglerighteq$ L5 $\trianglerighteq$ L1 $\gg$	$\ell\,’(a\,’v)$	;
L11:	z-SubtypeExtL’ $\trianglerighteq$ L4 $\trianglerighteq$ L2 $\gg$	$\ell\,’(b\,’u)$	;
L12:	z-SubtypeExtL’ $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\gg$	$\ell\,’(b\,’v)$	;
L13:	z-TypeEqual $\trianglerighteq$ L9 $\trianglerighteq$ L11 $\gg$	$!a\,’u = b\,’u$	;
L14:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L13 $\gg$	$!(\lambda y.a\,’y = b\,’y)\,’u$	;
L15:	Block $\gg$	Begin	;
L16:	Hypothesis $\gg$	$(\lambda y.a\,’y = b\,’y)’u$	;
L17:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L16 $\gg$	$a\,’u = b\,’u$	;
L18:	z-ComEqual $\trianglerighteq$ L10 $\trianglerighteq$ L9 $\trianglerighteq$ L7 $\gg$	$a\,’v = a\,’u$	;
L19:	z-TransEqual$\trianglerighteq$L10$\trianglerighteq$L9$\trianglerighteq$L11$\trianglerighteq$L18$\trianglerighteq$L17$\gg$	$a\,’v = b\,’u$	;
L20:	z-TransEqual$\trianglerighteq$L10$\trianglerighteq$L11$\trianglerighteq$L12$\trianglerighteq$L19$\trianglerighteq$L8$\gg$	$a\,’v = b\,’v$	;
L21:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L20 $\gg$	$(\lambda y.a\,’y = b\,’y)’v$	;
L22:	Block $\gg$	End	;
L23:	Block $\gg$	End	;
L24:	z-IntroZfcExt $\triangleright$ L14 $\triangleright$ L21 $\gg$	$\mathrm{ZfcExt}(\lambda y.a\,’y = b\,’y)$	]*

[The Map proof of z-ExtNot reads

L01:	Hypothesis $\gg$	$\mathrm{ZfcExt}(a)$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\ell\,’u$	;
L04:	Hypothesis $\gg$	$\ell\,’v$	;
L05:	Hypothesis $\gg$	$u = v$	;
L06:	z-ElimZfcExt$\trianglerighteq$L3$\trianglerighteq$L4$\trianglerighteq$L1$\trianglerighteq$L5$\gg$	$a\,’u \dot{\Leftrightarrow} a\,’v$	;
L07:	Logic $\trianglerighteq$ L6 $\gg$	$\neg a\,’u \dot{\Leftrightarrow} \neg a\,’v$	;
L08:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L7 $\gg$	$(\lambda y.\neg a\,’y)’u \dot{\Leftrightarrow} (\lambda y.\neg a\,’y)’v$	;
L09:	Block $\gg$	End	;
L10:	z-IntroZfcExtIff $\triangleright$ L8 $\gg$	$\mathrm{ZfcExt}(\lambda y.\neg a\,’y)$	]*

[The Map proof of z-ExtImply reads

L01:	Hypothesis $\gg$	$\mathrm{ZfcExt}(a)$	;
L02:	Hypothesis $\gg$	$\mathrm{ZfcExt}(b)$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\ell\,'\,u$	;
L05:	Hypothesis $\gg$	$\ell\,'\,v$	;
L06:	Hypothesis $\gg$	$u = v$	;
L07:	z-ElimZfcExt$\trianglerighteq$L4$\trianglerighteq$L5$\trianglerighteq$L1$\trianglerighteq$L6$\gg$	$a\,'\,u \dot{\Leftrightarrow} a\,'\,v$	;
L08:	z-ElimZfcExt$\trianglerighteq$L4$\trianglerighteq$L5$\trianglerighteq$L2$\trianglerighteq$L6$\gg$	$b\,'\,u \dot{\Leftrightarrow} b\,'\,v$	;
L09:	Logic $\trianglerighteq$ L7 $\trianglerighteq$ L8 $\gg$	$(a'u \Rightarrow b'u) \dot{\Leftrightarrow} (a'v \Rightarrow b'v)$	;
L10:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L9 $\gg$	$(\lambda y.a\,'\,y \Rightarrow b\,'\,y)\,'\,u \dot{\Leftrightarrow}$	
		$(\lambda y.a\,'\,y \Rightarrow b\,'\,y)\,'\,v$	;
L11:	Block $\gg$	End	;
L12:	z-IntroZfcExtIff $\triangleright$ L10 $\gg$	$\mathrm{ZfcExt}(\lambda y.a\,'\,y \Rightarrow b\,'\,y)$	]*

[The Map proof of z-ExtOr reads

L1:	Hypothesis $\gg$	$\mathrm{ZfcExt}(a)$	;
L2:	Hypothesis $\gg$	$\mathrm{ZfcExt}(b)$	;
L3:	z-ExtNot $\trianglerighteq$ L1 $\gg$	$\mathrm{ZfcExt}(\lambda y.\neg a\,'\,y)$	;
L4:	z-ExtImply $\trianglerighteq$ L3 $\trianglerighteq$ L2 $\gg$	$\mathrm{ZfcExt}(\lambda y.\neg a\,'\,y \Rightarrow b\,'\,y)$	;
L5:	Repetition $\triangleright$ L4 $\gg$	$\mathrm{ZfcExt}(\lambda y.a\,'\,y \dot{\vee} b\,'\,y)$	]*

[The Map proof of z-ExtAnd reads

L1:	Hypothesis $\gg$	$\mathrm{ZfcExt}(a)$	;
L2:	Hypothesis $\gg$	$\mathrm{ZfcExt}(b)$	;
L3:	z-ExtNot $\trianglerighteq$ L1 $\gg$	$\mathrm{ZfcExt}(\lambda y.\neg a\,'\,y)$	;
L4:	z-ExtNot $\trianglerighteq$ L2 $\gg$	$\mathrm{ZfcExt}(\lambda y.\neg b\,'\,y)$	;
L5:	z-ExtOr $\trianglerighteq$ L3 $\trianglerighteq$ L2 $\gg$	$\mathrm{ZfcExt}(\lambda y.\neg a\,'\,y \dot{\vee} \neg b\,'\,y)$	;
L6:	z-ExtNot $\trianglerighteq$ L5 $\gg$	$\mathrm{ZfcExt}(\lambda y.\neg(\neg a\,'\,y \dot{\vee} \neg b\,'\,y))$	;
L7:	Repetition $\triangleright$ L6 $\gg$	$\mathrm{ZfcExt}(\lambda y.a\,'\,y \dot{\wedge} b\,'\,y)$	]*

[The Map proof of z-ExtIff reads

L1:	Hypothesis $\gg$	$\mathrm{ZfcExt}(a)$	;
L2:	Hypothesis $\gg$	$\mathrm{ZfcExt}(b)$	;
L3:	z-ExtImply $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\mathrm{ZfcExt}(\lambda y.a\,'\,y \Rightarrow b\,'\,y)$	;
L4:	z-ExtImply $\trianglerighteq$ L2 $\trianglerighteq$ L1 $\gg$	$\mathrm{ZfcExt}(\lambda y.b\,'\,y \Rightarrow a\,'\,y)$	;
L5:	z-ExtAnd $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$\mathrm{ZfcExt}(\lambda y.(a\,'\,y \Rightarrow b\,'\,y) \dot{\wedge} (b\,'\,y \Rightarrow a\,'\,y)$	;
L6:	Repetition $\triangleright$ L5 $\gg$	$\mathrm{ZfcExt}(\lambda y.a\,'\,y \dot{\Leftrightarrow} b\,'\,y)$	]*

[The Map proof of AckermannE reads

```
 L01:  Premise $\gg$                          $\ell ' x : a \dot{\Leftrightarrow} b$                                ;
 L02:  Repetition $\triangleright$ L1 $\gg$                $\ell ' x \tilde{\wedge} a ' x \dot{\Leftrightarrow} b ' x \equiv \ell ' x \tilde{\wedge} \mathsf{T}$          ;
 L03:  Block $\gg$                            Begin                                        ;
 L04:  Algebra $\gg$                            $\exists x : a ' x$                                  ;
 L05:  AckermannExists $\gg$                    $\exists x : \ell ' x \wedge a ' x$                           ;
 L06:  Logic $\gg$                              $\exists x : (\ell ' x \tilde{\wedge} \mathsf{T}) \wedge a ' x$                     ;
 L07:  Replace $\triangleright$ L2 $\gg$                     $\exists x : (\ell ' x \tilde{\wedge} a ' x \dot{\Leftrightarrow} b ' x) \wedge a ' x$          ;
 L08:  Replace $\triangleright$ RangeEll $\gg$               $\exists x : (?\ell ' x \tilde{\wedge} a ' x \dot{\Leftrightarrow} b ' x) \wedge a ' x$         ;
 L09:  Logic $\gg$                              $\exists x : (?\ell ' x \tilde{\wedge} a ' x \dot{\Leftrightarrow} b ' x) \wedge b ' x$         ;
 L10:  Replace $\triangleright$ RangeEll $\gg$               $\exists x : (\ell ' x \tilde{\wedge} a ' x \dot{\Leftrightarrow} b ' x) \wedge b ' x$          ;
 L11:  Replace $\triangleright$ L2 $\gg$                     $\exists x : (\ell ' x \tilde{\wedge} \mathsf{T}) \wedge b ' x$                     ;
 L12:  Logic $\gg$                              $\exists x : \ell ' x \wedge b ' x$                           ;
 L13:  AckermannExists $\gg$                    $\exists x : b ' x$                                  ;
 L14:  Block $\gg$                            End                                          ;
 L15:  Block $\gg$                            Begin                                        ;
 L16:  Hypothesis $\gg$                         $\ell ' x$                                       ;
 L17:  L1 $\trianglerighteq$ L16 $\gg$                          $a \dot{\Leftrightarrow} b$                                     ;
 L18:  Logic $\trianglerighteq$ L17 $\gg$                       $!a$                                         ;
 L19:  Block $\gg$                            End                                          ;
 L20:  TypeExists $\trianglerighteq$ (Gen $\triangleright$ L18) $\gg$      $!\exists x : a$                                     ;
 L21:  Logic $\trianglerighteq$ L19 $\gg$                     $\exists x : a \dot{\Leftrightarrow} \exists x : a$                          ;
 L22:  Replace' $\trianglerighteq$ L13 $\trianglerighteq$ L21 $\gg$           $\exists x : a \dot{\Leftrightarrow} \exists x : b$                          ]*
```

[The Map proof of z-ExtExists reads

```
 L01:  Premise $\gg$                             $\ell ' x \rightarrow \mathrm{ZfcExt}(\lambda y . a)$                       ;
 L02:  Block $\gg$                               Begin                                     ;
 L03:  Hypothesis $\gg$                            $\ell ' u$                                    ;
 L04:  Hypothesis $\gg$                            $\ell ' v$                                    ;
 L05:  Hypothesis $\gg$                            $u = v$                                   ;
 L06:  Block $\gg$                                 Begin                                   ;
 L07:  Hypothesis $\gg$                              $\ell ' x$                                  ;
 L08:  L1 $\trianglerighteq$ L7 $\gg$                                $\mathrm{ZfcExt}(\lambda y . a)$                        ;
 L09:  z-ElimZfcExt$\trianglerighteq$L3$\trianglerighteq$L4$\trianglerighteq$L8$\trianglerighteq$L5$\gg$          $(\lambda y . a) ' u \dot{\Leftrightarrow} (\lambda y . a) ' v$                ;
 L10:  Block $\gg$                                 End                                     ;
 L11:  AckermannE $\triangleright$ L9 $\gg$                      $\exists x : (\lambda y . a) ' u \dot{\Leftrightarrow} \exists x : (\lambda y . a) ' v$    ;
 L12:  Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L11 $\gg$                      $(\lambda y . \exists x : a) ' u \dot{\Leftrightarrow} (\lambda y . \exists x : a) ' v$    ;
 L13:  Block $\gg$                               End                                       ;
 L14:  z-IntroZfcExtIff $\triangleright$ L12 $\gg$               $\mathrm{ZfcExt}(\lambda y . \exists x : a)$                   ]*
```

[The Map proof of z-ExtAll reads

L1:	Premise $\gg$	$\ell\,{}'\,x \to \mathrm{ZfcExt}(\lambda y.a)$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\ell\,{}'\,x$	;
L4:	$\mathsf{L1} \trianglerighteq \mathsf{L3} \gg$	$\mathrm{ZfcExt}(\lambda y.a)$	;
L5:	z-ExtNot $\trianglerighteq \mathsf{L4} \gg$	$\mathrm{ZfcExt}(\lambda y.\neg a)$	;
L6:	Block $\gg$	End	;
L7:	z-ExtExists $\triangleright \mathsf{L5} \gg$	$\mathrm{ZfcExt}(\lambda y.\exists x\colon \neg a)$	;
L8:	z-ExtNot $\trianglerighteq \mathsf{L7} \gg$	$\mathrm{ZfcExt}(\lambda y.\neg\exists x\colon \neg a)$	;
L9:	Repetition $\triangleright \mathsf{L8} \gg$	$\mathrm{ZfcExt}(\lambda y.\forall x\colon a)$	]*

[The Map proof of AckermannC reads

L01:	Premise $\gg$	$\ell x\colon a \mathrel{\dot{\Leftrightarrow}} b$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\ell\,{}'\,x$	;
L04:	$\mathsf{L1} \trianglerighteq \mathsf{L3} \gg$	$a \mathrel{\dot{\Leftrightarrow}} b$	;
L05:	Logic $\trianglerighteq \mathsf{L4} \gg$	$a \Leftrightarrow b$	;
L06:	Logic $\trianglerighteq \mathsf{L4} \gg$	$!a$	;
L07:	Block $\gg$	End	;
L08:	Ackermann $\trianglerighteq$ (Gen $\triangleright \mathsf{L5}$) $\gg$	$\varepsilon x\colon a \equiv \varepsilon x\colon b$	;
L09:	TypeChoice $\trianglerighteq$ (Gen $\triangleright \mathsf{L6}$) $\gg$	$\ell\,{}'\,\varepsilon x\colon a$	;
L10:	z-RefEqual $\trianglerighteq \mathsf{L9} \gg$	$\varepsilon x\colon a = \varepsilon x\colon a$	;
L11:	Replace' $\triangleright \mathsf{L8} \triangleright \mathsf{L10} \gg$	$\varepsilon x\colon a = \varepsilon x\colon b$	]*

[The Map proof of z-ExtChoice' reads

L01:	Premise $\gg$	$\ell\,{}'\,x \to \mathrm{ZfcExt}(\lambda y.a)$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\ell\,{}'\,u$	;
L04:	Hypothesis $\gg$	$\ell\,{}'\,v$	;
L05:	Hypothesis $\gg$	$u = v$	;
L06:	Block $\gg$	Begin	;
L07:	Hypothesis $\gg$	$\ell\,{}'\,x$	;
L08:	$\mathsf{L1} \trianglerighteq \mathsf{L7} \gg$	$\mathrm{ZfcExt}(\lambda y.a)$	;
L09:	z-ElimZfcExt$\trianglerighteq\mathsf{L3}\trianglerighteq\mathsf{L4}\trianglerighteq\mathsf{L8}\trianglerighteq\mathsf{L5}\gg$	$(\lambda y.a)\,{}'\,u \mathrel{\dot{\Leftrightarrow}} (\lambda y.a)\,{}'\,v$	;
L10:	z-SubtypeExtB $\trianglerighteq \mathsf{L3} \trianglerighteq \mathsf{L8} \gg$	$!(\lambda y.a)\,{}'\,u$	;
L11:	Block $\gg$	End	;
L12:	AckermannC $\triangleright \mathsf{L9} \gg$	$\varepsilon x\colon (\lambda y.a)\,{}'\,u = \varepsilon x\colon (\lambda y.a)\,{}'\,v$	;
L13:	Trans $\triangleright \bar{\mathcal{R}} \triangleright \mathsf{L12} \gg$	$(\lambda y.\varepsilon x\colon a)\,{}'\,u = (\lambda y.\varepsilon x\colon a)\,{}'\,v$	;
L14:	TypeChoice $\trianglerighteq$ (Gen $\triangleright \mathsf{L10}$) $\gg$	$\ell\,{}'\,\varepsilon x\colon (\lambda y.a)\,{}'\,u$	;
L15:	Trans $\triangleright \bar{\mathcal{R}} \triangleright \mathsf{L14} \gg$	$\ell\,{}'((\lambda y.\varepsilon x\colon a)\,{}'\,u)$	;
L16:	Block $\gg$	End	;
L17:	z-IntroZfcExt' $\triangleright \mathsf{L15} \triangleright \mathsf{L13} \gg$	$\mathrm{ZfcExt}'(\lambda y.\varepsilon x\colon a)$	]*

A.59 Lemmas about $[\,\emptyset\,]$

$[\,\emptyset \doteq \mathsf{T}\,]^*$

[Map lemma
z-TypeEmpty$_{244}$: $\ell\,'\,\emptyset$;
z-ExtEmpty$_{244}$: $\mathrm{ZfcExt}'(\lambda y.\emptyset)$;
z-n$_{244}$: $\forall x{:}\, x \notin \emptyset$]*

[The Map proof of z-TypeEmpty reads
L1: TypeT $\gg$ $\ell\,'\,\emptyset$]*

[The Map proof of z-ExtEmpty reads
L1: z-TypeEmpty $\gg$ $\ell\,'\,\emptyset$;
L2: z-ExtVarX $\trianglerighteq$ L1 $\gg$ $\mathrm{ZfcExt}'(\lambda y.\emptyset)$]*

[The Map proof of z-n reads
L1: Block $\gg$ Begin ;
L2: Hypothesis $\gg$ $\ell\,'\,x$;
L3: Reduction $\gg$ $x \notin \emptyset$;
L4: Block $\gg$ End ;
L5: Gen $\triangleright$ L3 $\gg$ $\forall x{:}\, x \notin \emptyset$]*

A.60 Lemmas about $[\,\{x, y\}\,]$

$[\,\{x, y\} \doteq x :: y\,]^*$

[Map lemma
z-TypePair$_{244}$: $\ell x, y{:}\, \ell\,'\{x, y\}$;
z-p1$_{245}$: $\ell x, y, z{:}\, x \in \{y, z\} \Rightarrow x = y \,\dot\vee\, x = z$;
z-p2$_{246}$: $\ell x, y, z{:}\, x = y \,\dot\vee\, x = z \Rightarrow x \in \{y, z\}$;
z-p$_{246}$: $\forall x \forall y \forall z{:}\, (x \in \{y, z\} \Leftrightarrow x = y \,\dot\vee\, x = z)$;
z-ExtPair$_{247}$: $\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(b) \to \mathrm{ZfcExt}'(\lambda y.\{a\,'\,y, b\,'\,y\})$]*

[The Map proof of z-TypePair reads
L1: TypePair $\gg$ $\ell x, y{:}\, \ell\,'\{x, y\}$]*

[The Map proof of z-p1 reads

L01:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L02:	Hypothesis $\gg$	$\ell\,\text{'}\,y$	;
L03:	Hypothesis $\gg$	$\ell\,\text{'}\,z$	;
L04:	z-TypePair $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$\ell\,\text{'}\{y,z\}$	;
L05:	z-TypeIn $\trianglerighteq$ L1 $\trianglerighteq$ L4 $\gg$	$!x \in \{y,z\}$	;
L06:	z-TypeEqual $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$!x = y$	;
L07:	z-TypeEqual $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$!x = z$	;
L08:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$!(x = y \mathbin{\dot{\vee}} x = z)$	;
L09:	Block $\gg$	Begin	;
L10:	Hypothesis $\gg$	$\quad x \in \{y,z\}$	;
L11:	Define $\gg$	$\quad u \equiv \varepsilon u\colon x = \{y,z\}\,\text{'}\,u$	;
L12:	z-ElimIn $\trianglerighteq$ L10 $\gg$	$\quad x = \{y,z\}\,\text{'}\,u$	;
L13:	z-ElimInEll $\trianglerighteq$ L10 $\gg$	$\quad \ell\,\text{'}\,u$	;
L14:	SubtypeLB $\trianglerighteq$ L13 $\gg$	$\quad !u$	;
L15:	Block $\gg$	$\quad$ Begin	;
L16:	Hypothesis $\gg$	$\qquad u$	;
L17:	Logic $\trianglerighteq$ L16 $\gg$	$\qquad \{y,z\}\,\text{'}\,u \equiv y$	;
L18:	Replace' $\triangleright$ L17 $\triangleright$ L12 $\gg$	$\qquad x = y$	;
L19:	Logic $\trianglerighteq$ L7 $\trianglerighteq$ L18 $\gg$	$\qquad x = y \mathbin{\dot{\vee}} x = z$	;
L20:	Block $\gg$	$\quad$ End	;
L21:	Block $\gg$	$\quad$ Begin	;
L22:	Hypothesis $\gg$	$\qquad \neg u$	;
L23:	Logic $\trianglerighteq$ L22 $\gg$	$\qquad \{y,z\}\,\text{'}\,u \equiv z$	;
L24:	Replace' $\triangleright$ L23 $\triangleright$ L12 $\gg$	$\qquad x = z$	;
L25:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L24 $\gg$	$\qquad x = y \mathbin{\dot{\vee}} x = z$	;
L26:	Block $\gg$	$\quad$ End	;
L27:	TND $\triangleright$ L14 $\triangleright$ L19 $\triangleright$ L25 $\gg$	$\quad x = y \mathbin{\dot{\vee}} x = z$	;
L28:	Block $\gg$	End	;
L29:	CDeduction $\triangleright$ L5 $\triangleright$ L8 $\triangleright$ L27 $\gg$	$x \in \{y,z\} \Rightarrow x = y \mathbin{\dot{\vee}} x = z$	$]^*$

[The Map proof of z-p2 reads

L01:	Hypothesis $\gg$	$\ell\,{}'x$	;
L02:	Hypothesis $\gg$	$\ell\,{}'y$	;
L03:	Hypothesis $\gg$	$\ell\,{}'z$	;
L04:	z-TypePair $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$\ell\,{}'\{y,z\}$	;
L05:	z-TypeIn $\trianglerighteq$ L1 $\trianglerighteq$ L4 $\gg$	$!x \in \{y,z\}$	;
L06:	z-TypeEqual $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$!x = y$	;
L07:	z-TypeEqual $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$!x = z$	;
L08:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$!(x = y \mathbin{\dot\vee} x = z)$	;
L09:	Block $\gg$	Begin	;
L10:	Hypothesis $\gg$	$x = y \mathbin{\dot\vee} x = z$	;
L11:	Reduction $\gg$	$\neg\{y,z\}$	;
L12:	Block $\gg$	Begin	;
L13:	Hypothesis $\gg$	$x = y$	;
L14:	TypeT $\gg$	$\ell\,{}'\mathsf{T}$	;
L15:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L13 $\gg$	$x = \{y,z\}\,{}'\mathsf{T}$	;
L16:	z-IntroIn$\trianglerighteq$L1$\trianglerighteq$L4$\trianglerighteq$L14$\trianglerighteq$L11$\trianglerighteq$L15$\gg$	$x \in \{y,z\}$	;
L17:	Block $\gg$	End	;
L18:	Block $\gg$	Begin	;
L19:	Hypothesis $\gg$	$\neg x = y$	;
L20:	Logic $\trianglerighteq$ L10 $\trianglerighteq$ L19 $\gg$	$x = z$	;
L21:	TypeF $\gg$	$\ell\,{}'\mathsf{F}$	;
L22:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L20 $\gg$	$x = \{y,z\}\,{}'\mathsf{F}$	;
L23:	z-IntroIn$\trianglerighteq$L1$\trianglerighteq$L4$\trianglerighteq$L21$\trianglerighteq$L11$\trianglerighteq$L22$\gg$	$x \in \{y,z\}$	;
L24:	Block $\gg$	End	;
L25:	TND $\rhd$ L6 $\rhd$ L16 $\rhd$ L23 $\gg$	$x \in \{y,z\}$	;
L26:	Block $\gg$	End	;
L27:	CDeduction $\rhd$ L8 $\rhd$ L5 $\rhd$ L25 $\gg$	$x = y \mathbin{\dot\vee} x = z \Rightarrow x \in \{y,z\}$	]*

[The Map proof of z-p reads

L1:	Block $\gg$	Begin	;
L2:	Hypothesis $\gg$	$\ell\,{}'x$	;
L3:	Hypothesis $\gg$	$\ell\,{}'y$	;
L4:	Hypothesis $\gg$	$\ell\,{}'z$	;
L5:	z-p1 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$x \in \{y,z\} \Rightarrow x = y \mathbin{\dot\vee} x = z$	;
L6:	z-p2 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$x = y \mathbin{\dot\vee} x = z \Rightarrow x \in \{y,z\}$	;
L7:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$x \in \{y,z\} \Leftrightarrow x = y \mathbin{\dot\vee} x = z$	;
L8:	Block $\gg$	End	;
L9:	Gen3 $\rhd$ L7 $\gg$	$\forall x\colon \forall y\colon \forall z\colon (x \in \{y,z\} \Leftrightarrow x = y \mathbin{\dot\vee} x = z)$	]*

[The Map proof of z-ExtPair reads

L01:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(a)$	;
L02:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(b)$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\ell\,'\,x$	;
L05:	Hypothesis $\gg$	$\ell\,'\,y$	;
L06:	z-SubtypeExtL' $\trianglerighteq$ L5 $\trianglerighteq$ L1 $\gg$	$\ell\,'(a\,'\,y)$	;
L07:	z-SubtypeExtL' $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\gg$	$\ell\,'(b\,'\,y)$	;
L08:	z-TypePair $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$\ell\,'\{a\,'\,y, b\,'\,y\}$	;
L09:	ElimAll3 $\trianglerighteq$ z-p$\trianglerighteq$L4$\trianglerighteq$L6$\trianglerighteq$L7$\gg$	$x{\in}\{a'y, b'y\}{\dot\Leftrightarrow}x{=}a'y{\vee}x{=}b'y$	;
L10:	z-ExtVarX $\trianglerighteq$ L4 $\gg$	$\mathrm{ZfcExt}'(\lambda y.x)$	;
L11:	z-ExtEqual $\trianglerighteq$ L10 $\trianglerighteq$ L1 $\gg$	$\mathrm{ZfcExt}(\lambda y.x = a\,'\,y)$	;
L12:	z-ExtEqual $\trianglerighteq$ L10 $\trianglerighteq$ L2 $\gg$	$\mathrm{ZfcExt}(\lambda y.x = b\,'\,y)$	;
L13:	z-ExtOr $\trianglerighteq$ L11 $\trianglerighteq$ L12 $\gg$	$\mathrm{ZfcExt}(\lambda y.x{=}a'y{\vee}x{=}b'y)$	;
L14:	Block $\gg$	End	;
L15:	z-IntroZfcExtIff'$\triangleright$L8$\triangleright$L9$\triangleright$L13$\gg$	$\mathrm{ZfcExt}'(\lambda y.\{a\,'\,y, b\,'\,y\})$	]*

A.61 Lemmas about $[\,\{x \in y|\mathcal{A}\}\,]$

[$\mathsf{S}'(y,f)$	$\doteq$	$\exists x\colon x \in y \mathbin{\dot\wedge} f\,'\,x$	]*
[$\mathsf{S}''(y,f)$	$\doteq$	$\lambda x.\mathrm{if}(f\,'(y\,'\,x), y\,'\,x, \varepsilon x\colon x \in y \mathbin{\dot\wedge} f\,'\,x)$	]*
[$\mathsf{S}(y,f)$	$\doteq$	$\mathrm{if}(\mathsf{S}'(y,f), \mathsf{S}''(y,f), \emptyset)$	]*
[$\{x{\in}y|\mathcal{A}\}$	$\doteq$	$\mathsf{S}(y, \lambda x.\mathcal{A})$	]*

[Map lemma

z-TypeS1$_{247}$:	$\ell y\colon !_1 f\colon \ell x\colon !x \in y \mathbin{\dot\wedge} f\,'\,x$	;
z-TypeS'$_{248}$:	$\ell y\colon !_1 f\colon !\mathsf{S}'(y,f)$	;
z-TypeS"$_{248}$:	$\ell y\colon !_1 f\colon \ell\,'\,\mathsf{S}''(y,f)$	;
z-TypeS$_{248}$:	$\ell y\colon !_1 f\colon \ell\,'\,\mathsf{S}(y,f)$	;
z-SImply$_{248}$:	$\ell x,y\colon \mathrm{ZfcExt}(f) \to x \in \mathsf{S}(y,f) \to x \in y \mathbin{\dot\wedge} f\,'\,x$	;
z-SImplied$_{249}$:	$\ell x,y\colon \mathrm{ZfcExt}(f) \to x \in y \mathbin{\dot\wedge} f\,'\,x \to x \in \mathsf{S}(y,f)$	;
z-SIff$_{250}$:	$\ell x,y\colon \mathrm{ZfcExt}(f) \to x \in \mathsf{S}(y,f) \mathbin{\dot\Leftrightarrow} x \in y \mathbin{\dot\wedge} f\,'\,x$	;
z-TypeSubset$_{250}$:	$\langle x{\in}\mathcal{V}; a,b{\in}\mathcal{T}\rangle \ell\,'\,a \vdash \mathrm{ZfcExt}(\lambda x.b) \vdash \ell\,'\{x{\in}a|b\}$	;
z-Subset$_{251}$:	$\langle x{\in}\mathcal{V}; a,b,c,c'{\in}\mathcal{T}\rangle c' \simeq \langle c \mid x{:=}a\rangle \Vdash \ell\,'\,a \vdash \ell\,'\,b \vdash$ $\mathrm{ZfcExt}(\lambda x.c) \vdash a \in \{x{\in}b|c\} \mathbin{\dot\Leftrightarrow} a \in b \mathbin{\dot\wedge} c'$	;
z-s$_{251}$:	$\langle x,y,z{\in}\mathcal{V}; a{\in}\mathcal{T}\rangle \mathrm{notfree}(y,z;a) \Vdash \mathrm{ZfcExt}(\lambda x.a) \to$ $\forall y \exists z \forall x\colon (x \in z \mathbin{\dot\Leftrightarrow} x \in y \mathbin{\dot\wedge} a)$	]*

[The Map proof of z-TypeS1 reads

L1:	Hypothesis $\gg$	$\ell\,'\,y$	;
L2:	Hypothesis $\gg$	$!_1 f$	;
L3:	Hypothesis $\gg$	$\ell\,'\,x$	;
L4:	z-TypeIn $\trianglerighteq$ L3 $\trianglerighteq$ L1 $\gg$	$!x \in y$	;
L5:	ElimAll $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$!f\,'\,x$	;
L6:	Logic $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\gg$	$!(x \in y \mathbin{\dot\wedge} f\,'\,x)$	]*

[The Map proof of z-TypeS' reads

L1:	Hypothesis $\gg$	$\ell\text{'}y$	;
L2:	Hypothesis $\gg$	$!_1 f$	;
L3:	TypeExists $\trianglerighteq$ (z-TypeS1 $\trianglerighteq$ L1 $\trianglerighteq$ L2) $\gg$	$!\exists x\colon x \in y \mathbin{\dot\wedge} f\text{'}x$	;
L4:	Repetition $\triangleright$ L3 $\gg$	$!\mathsf{S}'(y,f)$	]*

[The Map proof of z-TypeS" reads

L01:	Hypothesis $\gg$	$\ell\text{'}y$	;
L02:	Hypothesis $\gg$	$!_1 f$	;
L03:	Define $\gg$	$p \equiv \mathcal{I}y$	;
L04:	ElimEll $\trianglerighteq$ L1 $\gg$	$y \in_\ell p$	;
L05:	StrictInEllP $\trianglerighteq$ L4 $\gg$	$\ell\text{'}p$	;
L06:	Block $\gg$	Begin	;
L07:	Hypothesis $\gg$	$u \sim_p v$	;
L08:	ElimInEll $\trianglerighteq$ L4 $\trianglerighteq$ L7 $\gg$	$y\text{'}u \sim y\text{'}v$	;
L09:	ExtBool $\trianglerighteq$ L2 $\trianglerighteq$ L8 $\gg$	$f\text{'}(y\text{'}u) \Leftrightarrow f\text{'}(y\text{'}v)$	;
L10:	z-TypeS1 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell x\colon !(x \in y \mathbin{\dot\wedge} f\text{'}x)$	;
L11:	TypeChoice $\trianglerighteq$ L10 $\gg$	$\ell\text{'}\varepsilon x\colon x \in y \mathbin{\dot\wedge} f\text{'}x$	;
L12:	RefEquivK $\trianglerighteq$ L11 $\gg$	$(\varepsilon x{:}x{\in}y\dot\wedge f\text{'}x) \sim (\varepsilon x{:}x{\in}y\dot\wedge f\text{'}x)$	;
L13:	ExtIfBKK$\trianglerighteq$L9$\trianglerighteq$L8$\trianglerighteq$L12$\gg$	$\mathrm{if}(f\text{'}(y\text{'}u), y\text{'}u, \varepsilon x{:}x{\in}y\dot\wedge f\text{'}x) \sim$ $\mathrm{if}(f\text{'}(y\text{'}v), y\text{'}v, \varepsilon x{:}x{\in}y\dot\wedge f\text{'}x)$	;
L14:	Repetition $\triangleright$ L13 $\gg$	$\mathsf{S}''(y,f)\text{'}u \sim \mathsf{S}''(y,f)\text{'}v$	;
L15:	Block $\gg$	End	;
L16:	IntroInEll $\triangleright$ L5 $\triangleright$ L14 $\gg$	$\mathsf{S}''(y,f) \in_\ell p$	;
L17:	IntroEll $\triangleright$ L16 $\gg$	$\ell\text{'}\mathsf{S}''(y,f)$	]*

[The Map proof of z-TypeS reads

L01:	Hypothesis $\gg$	$\ell\text{'}y$	;
L02:	Hypothesis $\gg$	$!_1 f$	;
L03:	z-TypeS' $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$!\mathsf{S}'(y,f)$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\mathsf{S}'(y,f)$	;
L06:	Logic $\trianglerighteq$ L5 $\gg$	$\mathsf{S}(y,f) \equiv \mathsf{S}''(y,f)$	;
L07:	z-TypeS" $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\text{'}\mathsf{S}''(y,f)$	;
L08:	Replace' $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$\ell\text{'}\mathsf{S}(y,f)$	;
L09:	Block $\gg$	End	;
L10:	Logic $\gg$	$\neg\mathsf{S}'(y,f) \rightarrow \ell\text{'}\mathsf{S}(y,f)$	;
L11:	TND $\triangleright$ L3 $\triangleright$ L8 $\triangleright$ L10 $\gg$	$\ell\text{'}\mathsf{S}(y,f)$	]*

[The Map proof of z-SImply reads

L01:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L02:	Hypothesis $\gg$	$\ell\,\text{'}\,y$	;
L03:	Hypothesis $\gg$	$\mathrm{ZfcExt}(f)$	;
L04:	Hypothesis $\gg$	$x \in \mathsf{S}(y,f)$	;
L05:	z-SubtypeExtB1 $\unrhd$ L3 $\gg$	$!_1 f$	;
L06:	Logic $\unrhd$ L4 $\gg$	$\mathsf{S}'(y,f)$	;
L07:	Logic $\unrhd$ L6 $\gg$	$\mathsf{S}(y,f) \equiv \mathsf{S}''(y,f)$	;
L08:	Replace' $\rhd$ L7 $\rhd$ L4 $\gg$	$x \in \mathsf{S}''(y,f)$	;
L09:	Define $\gg$	$z \equiv \varepsilon z \colon x = \mathsf{S}''(y,f)\,\text{'}\,z$	;
L10:	z-ElimInEll $\unrhd$ L8 $\gg$	$\ell\,\text{'}\,z$	;
L11:	z-ElimIn $\unrhd$ L8 $\gg$	$x = \mathsf{S}''(y,f)\,\text{'}\,z$	;
L12:	Define $\gg$	$u \equiv \varepsilon u \colon u \in y \mathrel{\dot\wedge} f\,\text{'}\,u$	;
L13:	ElimExistsEll $\unrhd$ L6 $\gg$	$\ell\,\text{'}\,u$	;
L14:	ElimExists $\unrhd$ L6 $\gg$	$u \in y \mathrel{\dot\wedge} f\,\text{'}\,u$	;
L15:	Trans $\rhd \bar{\mathcal{R}} \rhd$ L11 $\gg$	$x = \mathrm{if}(f\,\text{'}(y\,\text{'}\,z), y\,\text{'}\,z, u)$	;
L16:	TypeApply $\unrhd$ L2 $\unrhd$ L10 $\gg$	$\ell\,\text{'}(y\,\text{'}\,z)$	;
L17:	ElimAll $\unrhd$ L5 $\unrhd$ L16 $\gg$	$!f\,\text{'}(y\,\text{'}\,z)$	;
L18:	Block $\gg$	Begin	;
L19:	Hypothesis $\gg$	$\quad f\,\text{'}(y\,\text{'}\,z)$	;
L20:	Logic $\unrhd$ L19 $\gg$	$\quad \mathrm{if}(f\,\text{'}(y\,\text{'}\,z), y\,\text{'}\,z, u) \equiv y\,\text{'}\,z$	;
L21:	Replace' $\rhd$ L20 $\rhd$ L15 $\gg$	$\quad x = y\,\text{'}\,z$	;
L22:	Logic $\unrhd$ L14 $\gg$	$\quad \neg y$	;
L23:	z-IntroIn$\unrhd$L1$\unrhd$L2$\unrhd$L10$\unrhd$L22$\unrhd$L21$\gg$	$\quad x \in y$	;
L24:	ElimAll2 $\unrhd$ L3 $\unrhd$ L1 $\unrhd$ L16 $\gg$	$\quad x{=}y\text{'}z \Rightarrow (f\text{'}x \Leftrightarrow f\text{'}(y\text{'}z))$	;
L25:	Logic $\unrhd$ L19 $\unrhd$ L21 $\unrhd$ L23 $\unrhd$ L24 $\gg$	$\quad x \in y \mathrel{\dot\wedge} f\,\text{'}\,x$	;
L26:	Block $\gg$	End	;
L27:	Block $\gg$	Begin	;
L28:	Hypothesis $\gg$	$\quad \neg f\,\text{'}(y\,\text{'}\,z)$	;
L29:	Logic $\unrhd$ L28 $\gg$	$\quad \mathrm{if}(f\,\text{'}(y\,\text{'}\,z), y\,\text{'}\,z, u) \equiv u$	;
L30:	Replace' $\rhd$ L29 $\rhd$ L15 $\gg$	$\quad x = u$	;
L31:	Logic $\unrhd$ L14 $\gg$	$\quad u \in y$	;
L32:	z-ext1$\unrhd$L1$\unrhd$L13$\unrhd$L2$\unrhd$L30$\unrhd$L31$\gg$	$\quad x \in y$	;
L33:	ElimAll2 $\unrhd$ L3 $\rhd$ L1 $\unrhd$ L13 $\gg$	$\quad x = u \Rightarrow (f\,\text{'}\,x \Leftrightarrow f\,\text{'}\,u)$	;
L34:	Logic $\unrhd$ L14 $\unrhd$ L30 $\unrhd$ L32 $\unrhd$ L33 $\gg$	$\quad x \in y \mathrel{\dot\wedge} f\,\text{'}\,x$	;
L35:	Block $\gg$	End	;
L36:	TND $\rhd$ L17 $\rhd$ L25 $\rhd$ L34 $\gg$	$x \in y \mathrel{\dot\wedge} f\,\text{'}\,x$	]*

[The Map proof of z-SImplied reads

L01:	Hypothesis $\gg$	$\ell\,'\,x$	;
L02:	Hypothesis $\gg$	$\ell\,'\,y$	;
L03:	Hypothesis $\gg$	$\mathrm{ZfcExt}(f)$	;
L04:	Hypothesis $\gg$	$x \in y \mathbin{\dot\wedge} f\,'\,x$	;
L05:	Logic $\trianglerighteq$ L4 $\gg$	$x \in y$	;
L06:	Define $\gg$	$z \equiv \varepsilon z\colon x = y\,'\,z$	;
L07:	z-ElimInEll $\trianglerighteq$ L5 $\gg$	$\ell\,'\,z$	;
L08:	z-ElimIn $\trianglerighteq$ L5 $\gg$	$x = y\,'\,z$	;
L09:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L7 $\gg$	$\ell\,'(y\,'\,z)$	;
L10:	ElimAll2$\trianglerighteq$L3$\trianglerighteq$L1$\trianglerighteq$L9$\gg$	$x = y\,'\,z \Rightarrow (f\,'\,x \Leftrightarrow f\,'(y\,'\,z))$	;
L11:	Logic $\trianglerighteq$ L4 $\trianglerighteq$ L8 $\trianglerighteq$ L10 $\gg$	$f\,'(y\,'\,z)$	;
L12:	z-SubtypeExtB1 $\trianglerighteq$ L3 $\gg$	$!_1 f$	;
L13:	Block $\gg$	Begin	;
L14:	Hypothesis $\gg$	$\quad\ell\,'\,u$	;
L15:	z-TypeIn $\trianglerighteq$ L14 $\trianglerighteq$ L2 $\gg$	$\quad!u \in y$	;
L16:	ElimAll $\trianglerighteq$ L12 $\trianglerighteq$ L14 $\gg$	$\quad!f\,'\,u$	;
L17:	Logic $\trianglerighteq$ L15 $\trianglerighteq$ L16 $\gg$	$\quad!(u \in y \mathbin{\dot\wedge} f\,'\,u)$	;
L18:	Block $\gg$	End	;
L19:	IntroExists $\trianglerighteq$ (Gen $\triangleright$ L17) $\trianglerighteq$ L1 $\trianglerighteq$ L4 $\gg$	$\mathsf{S}'(y,f)$	;
L20:	Logic $\trianglerighteq$ L19 $\gg$	$\mathsf{S}(y,f) \equiv \mathsf{S}''(y,f)$	;
L21:	z-TypeS" $\trianglerighteq$ L2 $\trianglerighteq$ L12 $\gg$	$\ell\,'\,\mathsf{S}''(y,f)$	;
L22:	Reduction $\gg$	$\neg\mathsf{S}''(y,f)$	;
L23:	Logic $\trianglerighteq$ L11 $\gg$	$\mathrm{if}(f\,'(y\,'\,z), y\,'\,z, \varepsilon x{:}x{\in}y{\dot\wedge}f'x) \equiv y\,'\,z$	;
L24:	Replace' $\triangleright$ L23 $\triangleright$ L8 $\gg$	$x = \mathrm{if}(f\,'(y\,'\,z), y\,'\,z, \varepsilon x{:}x{\in}y{\dot\wedge}f'x)$	;
L25:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L24 $\gg$	$x = \mathsf{S}''(y,f)\,'\,z$	;
L26:	z-IntroIn $\trianglerighteq$ L1 $\trianglerighteq$ L21 $\trianglerighteq$ L7 $\trianglerighteq$ L22 $\trianglerighteq$ L25 $\gg$	$x \in \mathsf{S}''(y,f)$	;
L27:	Replace' $\trianglerighteq$ L20 $\trianglerighteq$ L26 $\gg$	$x \in \mathsf{S}(y,f)$	]*

[The Map proof of z-SIff reads

L01:	Hypothesis $\gg$	$\ell\,'\,x$	;
L02:	Hypothesis $\gg$	$\ell\,'\,y$	;
L03:	Hypothesis $\gg$	$\mathrm{ZfcExt}(f)$	;
L04:	z-SubtypeExtB1 $\trianglerighteq$ L3 $\gg$	$!_1 f$	;
L05:	z-TypeS $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\gg$	$\ell\,'\,\mathsf{S}(y,f)$	;
L06:	z-TypeIn $\trianglerighteq$ L1 $\trianglerighteq$ L5 $\gg$	$!x \in \mathsf{S}(y,f)$	;
L07:	z-TypeIn $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$!x \in y$	;
L08:	ElimAll $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\gg$	$!f\,'\,x$	;
L09:	Logic $\trianglerighteq$ L7 $\trianglerighteq$ L8 $\gg$	$!(x \in y \mathbin{\dot\wedge} f\,'\,x)$	;
L10:	z-SImply $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$x \in \mathsf{S}(y,f) \rightarrow x \in y \mathbin{\dot\wedge} f\,'\,x$	;
L11:	z-SImplied $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$x \in y \mathbin{\dot\wedge} f\,'\,x \rightarrow x \in \mathsf{S}(y,f)$	;
L12:	CDeduction $\triangleright$ L6 $\triangleright$ L9 $\triangleright$ L10 $\gg$	$x \in \mathsf{S}(y,f) \Rightarrow x \in y \mathbin{\dot\wedge} f\,'\,x$	;
L13:	CDeduction $\triangleright$ L9 $\triangleright$ L6 $\triangleright$ L11 $\gg$	$x \in y \mathbin{\dot\wedge} f\,'\,x \Rightarrow x \in \mathsf{S}(y,f)$	;
L14:	Logic $\trianglerighteq$ L12 $\trianglerighteq$ L13 $\gg$	$x \in \mathsf{S}(y,f) \Leftrightarrow x \in y \mathbin{\dot\wedge} f\,'\,x$	]*

[The Map proof of z-TypeSubset reads

L1:	Premise $\gg$	ℓ ' a	;
L2:	Premise $\gg$	ZfcExt$(\lambda x.b)$	;
L3:	z-SubtypeExtB1 $\trianglerighteq$ L2 $\gg$	$!_1 \lambda x.b$	;
L4:	z-TypeS $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	ℓ ' $\{x{\in}a\|b\}$	]*

[The Map proof of z-Subset reads

L1:	Premise $\gg$	ℓ ' a	;
L2:	Premise $\gg$	ℓ ' b	;
L3:	Premise $\gg$	ZfcExt$(\lambda x.c)$	;
L4:	z-SIff $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$a \in \mathsf{S}(b, \lambda x.c) \Leftrightarrow a \in b \dot{\wedge} (\lambda x.c)$ ' a	;
L5:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L4 $\gg$	$a \in \{x{\in}b\|c\} \Leftrightarrow a \in b \dot{\wedge} c'$	]*

[The Map proof of z-s reads

L01:	Hypothesis $\gg$	ZfcExt$(\lambda x.\mathcal{A})$	;
L02:	z-SubtypeExtB1 $\trianglerighteq$ L1 $\gg$	$!_1 \lambda x.\mathcal{A}$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\quad \ell$ ' y	;
L05:	Block $\gg$	$\quad$ Begin	;
L06:	Hypothesis $\gg$	$\quad\quad \ell$ ' z	;
L07:	Block $\gg$	$\quad\quad$ Begin	;
L08:	Hypothesis $\gg$	$\quad\quad\quad \ell$ ' x	;
L09:	z-TypeIn $\trianglerighteq$ L8 $\trianglerighteq$ L6 $\gg$	$\quad\quad\quad !x \in z$	;
L10:	z-TypeIn $\trianglerighteq$ L8 $\trianglerighteq$ L4 $\gg$	$\quad\quad\quad !x \in y$	;
L11:	ElimAll $\trianglerighteq$ L2 $\trianglerighteq$ L8 $\gg$	$\quad\quad\quad !(\lambda x.\mathcal{A})$ ' a	;
L12:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L11 $\gg$	$\quad\quad\quad !\mathcal{A}$	;
L13:	Logic $\trianglerighteq$ L9 $\trianglerighteq$ L10 $\trianglerighteq$ L12 $\gg$	$\quad\quad\quad !(x \in z \Leftrightarrow x \in y \dot{\wedge} \mathcal{A})$	;
L14:	Block $\gg$	$\quad\quad$ End	;
L15:	z-TypeSubset $\triangleright$ L4 $\triangleright$ L1 $\gg$	$\quad\quad \ell$ ' $\{x{\in}y\|\mathcal{A}\}$	;
L16:	Block $\gg$	$\quad\quad$ Begin	;
L17:	Hypothesis $\gg$	$\quad\quad\quad \ell$ ' x	;
L18:	z-Subset $\triangleright$ L17 $\triangleright$ L4 $\triangleright$ L1 $\gg$	$\quad\quad\quad x \in \{x{\in}y\|\mathcal{A}\} \Leftrightarrow x \in y \dot{\wedge} \mathcal{A}$	;
L19:	Block $\gg$	$\quad\quad$ End	;
L20:	Gen $\triangleright$ L18 $\gg$	$\quad\quad \forall x\colon x \in \{x{\in}y\|\mathcal{A}\} \Leftrightarrow x \in y \dot{\wedge} \mathcal{A}$	;
L21:	IntroExists $\trianglerighteq$ (Gen $\triangleright$ L13) $\trianglerighteq$ L14 $\trianglerighteq$ L20 $\gg$	$\quad\quad \exists z \forall x\colon (x \in z \Leftrightarrow x \in y \dot{\wedge} \mathcal{A})$	;
L22:	Block $\gg$	$\quad$ End	;
L23:	Gen $\triangleright$ L21 $\gg$	$\quad \forall y \exists z \forall x\colon (x \in z \Leftrightarrow x \in y \dot{\wedge} \mathcal{A})$	]*

A.62 Lemmas about $[\cup x]$

[$\bigcup x \doteq \{u \in u_4(x) \| \exists v\colon u \in v \dot{\wedge} v \in x\}$]*

[Map lemma

z-ExtUnion'$_{252}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(b) \to$ $\mathrm{ZfcExt}(\lambda u.\exists v\colon a\,{'}\,u \in v \mathbin{\dot\wedge} v \in b\,{'}\,u)$	;
z-TypeUnion$_{252}$:	$\ell x\colon \ell\,{'} \bigcup x$	;
z-u1$_{253}$:	$\ell x, y, z\colon x \in z \to z \in y \to x \in u_4(y))$	;
z-u2$_{253}$:	$\ell x, y\colon \exists z\colon x \in z \mathbin{\dot\wedge} z \in y \to x \in u_4(y))$	;
z-u3$_{254}$:	$\ell x, y\colon \exists z\colon x{\in}z{\dot\wedge}z{\in}y \equiv x \in u_4(y) \mathbin{\dot\wedge} \exists z\colon x{\in}z{\dot\wedge}z{\in}y)$	;
z-u$_{254}$:	$\forall x \forall y\colon (x \in \bigcup y \mathbin{\dot\Leftrightarrow} \exists z\colon x \in z \mathbin{\dot\wedge} z \in y)$	;
z-ExtUnion$_{254}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(\lambda y. \bigcup a\,{'}\,y)$	]*

[The Map proof of z-ExtUnion' reads

L01:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(a)$	;
L02:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(b)$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\ell\,{'}\,v$	;
L05:	z-ExtVarX $\trianglerighteq$ L4 $\gg$	$\mathrm{ZfcExt}'(\lambda u.v)$	;
L06:	z-ExtIn $\trianglerighteq$ L1 $\trianglerighteq$ L5 $\gg$	$\mathrm{ZfcExt}(\lambda u.a\,{'}\,u \in v)$	;
L07:	z-ExtIn $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\gg$	$\mathrm{ZfcExt}(\lambda u.v \in b\,{'}\,u)$	;
L08:	z-ExtAnd $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$\mathrm{ZfcExt}(\lambda u.a\,{'}\,u \in v \mathbin{\dot\wedge} v \in b\,{'}\,u)$	;
L09:	Block $\gg$	End	;
L10:	z-ExtExists $\rhd$ L8 $\gg$	$\mathrm{ZfcExt}(\lambda u.\exists v\colon a\,{'}\,u \in v \mathbin{\dot\wedge} v \in b\,{'}\,u)$	]*

[The Map proof of z-TypeUnion reads

L1:	Hypothesis $\gg$	$\ell\,{'}\,x$	;
L2:	TypeU4 $\trianglerighteq$ L1 $\gg$	$\ell\,{'}\,u_4(x)$	;
L3:	z-ExtVarY $\gg$	$\mathrm{ZfcExt}'(\lambda u.u)$	;
L4:	z-ExtVarX $\trianglerighteq$ L1 $\gg$	$\mathrm{ZfcExt}'(\lambda u.x)$	;
L5:	z-ExtUnion' $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$\mathrm{ZfcExt}(\lambda u.\exists v\colon u \in v \mathbin{\dot\wedge} v \in x)$	;
L6:	z-TypeSubset $\rhd$ L2 $\rhd$ L5 $\gg$	$\ell\,{'}\{u \in u_4(x) \mid \exists v\colon u \in v \mathbin{\dot\wedge} v \in x\}$	;
L7:	Repetition $\rhd$ L6 $\gg$	$\ell\,{'} \bigcup x$	]*

[The Map proof of z-u1 reads

L01:	Hypothesis $\gg$	$\ell\,’x$	;
L02:	Hypothesis $\gg$	$\ell\,’y$	;
L03:	Hypothesis $\gg$	$\ell\,’z$	;
L04:	Hypothesis $\gg$	$x \in z$	;
L05:	Hypothesis $\gg$	$z \in y$	;
L06:	TypeU4 $\trianglerighteq$ L2 $\gg$	$\ell\,’u_4(y)$	;
L07:	Reduction $\gg$	$\neg u_4(y)$	;
L08:	Define $\gg$	$a \equiv \varepsilon a\colon x = z\,’a$	;
L09:	z-ElimInEll $\trianglerighteq$ L4 $\gg$	$\ell\,’a$	;
L10:	z-ElimIn $\trianglerighteq$ L4 $\gg$	$x = z\,’a$	;
L11:	Define $\gg$	$b \equiv \varepsilon b\colon z = y\,’b$	;
L12:	z-ElimInEll $\trianglerighteq$ L5 $\gg$	$\ell\,’b$	;
L13:	z-ElimIn $\trianglerighteq$ L5 $\gg$	$z = y\,’b$	;
L14:	Define $\gg$	$c \equiv \varepsilon c\colon z\,’a = y\,’b\,’c$	;
L15:	z-ElimEqualYEll $\trianglerighteq$ L9 $\trianglerighteq$ L13 $\gg$	$\ell\,’c$	;
L16:	z-ElimEqualY $\trianglerighteq$ L9 $\trianglerighteq$ L13 $\gg$	$z\,’a = y\,’b\,’c$	;
L17:	TypeApply $\trianglerighteq$ L3 $\trianglerighteq$ L9 $\gg$	$\ell\,’(z\,’a)$	;
L18:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L12 $\gg$	$\ell\,’(y\,’b)$	;
L19:	TypeApply $\trianglerighteq$ L18 $\trianglerighteq$ L15 $\gg$	$\ell\,’(y\,’b\,’c)$	;
L20:	z-TransEqual$\trianglerighteq$L1$\trianglerighteq$L17$\trianglerighteq$L19$\trianglerighteq$L10$\trianglerighteq$L16$\gg$	$x = y\,’b\,’c$	;
L21:	Define $\gg$	$w \equiv b :: c$	;
L22:	TypePair $\trianglerighteq$ L12 $\trianglerighteq$ L15 $\gg$	$\ell\,’w$	;
L23:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L20 $\gg$	$x = y\,’(w\,’\mathsf{T})\,’(w\,’\mathsf{F})$	;
L24:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L23 $\gg$	$x = (u_4(y))\,’w$	;
L25:	z-IntroIn $\trianglerighteq$ L1 $\trianglerighteq$ L6 $\trianglerighteq$ L22 $\trianglerighteq$ L7 $\trianglerighteq$ L24 $\gg$	$x \in u_4(y)$	]*

[The Map proof of z-u2 reads

L1:	Hypothesis $\gg$	$\ell\,’x$	;
L2:	Hypothesis $\gg$	$\ell\,’y$	;
L3:	Hypothesis $\gg$	$\exists z\colon x \in z \mathbin{\dot\wedge} z \in y$	;
L4:	Define $\gg$	$z \equiv \varepsilon z\colon x \in z \mathbin{\dot\wedge} z \in y$	;
L5:	ElimExistsEll $\trianglerighteq$ L3 $\gg$	$\ell\,’z$	;
L6:	ElimExists $\trianglerighteq$ L3 $\gg$	$x \in z \mathbin{\dot\wedge} z \in y$	;
L7:	Logic $\trianglerighteq$ L6 $\gg$	$x \in z$	;
L8:	Logic $\trianglerighteq$ L6 $\gg$	$z \in y$	;
L9:	z-u1 $\trianglerighteq$ L1 $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\trianglerighteq$ L7 $\trianglerighteq$ L8 $\gg$	$x \in u_4(y)$	]*

[The Map proof of z-u3 reads

L01:	Hypothesis $\gg$	$\ell\text{'}x$	;
L02:	Hypothesis $\gg$	$\ell\text{'}y$	;
L03:	TypeU4 $\trianglerighteq$ L2 $\gg$	$\ell\text{'}u_4(y)$	;
L04:	z-TypeIn $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$!x \in u_4(y)$	;
L05:	Block $\gg$	Begin	;
L06:	Algebra $\gg$	$\exists z\colon x \in z \mathbin{\dot\wedge} z \in y$	;
L07:	Logic $\gg$	$(\exists z\colon x \in z \mathbin{\dot\wedge} z \in y) \mathbin{\tilde\wedge} \mathsf{T}$	;
L08:	Rev $\triangleright$ z-u2 $\gg$	$(\exists z\colon x \in z \mathbin{\dot\wedge} z \in y) \mathbin{\tilde\wedge} x \in u_4(y)$	;
L09:	Logic $\trianglerighteq$ L4 $\gg$	$x \in u_4(y) \mathbin{\dot\wedge} \exists z\colon x \in z \mathbin{\dot\wedge} z \in y)$	;
L10:	Block $\gg$	End	]*

[The Map proof of z-u reads

L01:	Block $\gg$	Begin	;
L02:	Hypothesis $\gg$	$\ell\text{'}x$	;
L03:	Hypothesis $\gg$	$\ell\text{'}y$	;
L04:	z-ExtVarY $\gg$	$\mathrm{ZfcExt}'(\lambda u.u)$	;
L05:	z-ExtVarX $\trianglerighteq$ L3 $\gg$	$\mathrm{ZfcExt}'(\lambda u.y)$	;
L06:	z-ExtUnion'$\trianglerighteq$L4$\trianglerighteq$L5$\gg$	$\mathrm{ZfcExt}(\lambda u.\exists z\colon u \in z \mathbin{\dot\wedge} z \in y)$	;
L07:	TypeU4 $\trianglerighteq$ L3 $\gg$	$\ell\text{'}u_4(y)$	;
L08:	z-Subset$\triangleright$L2$\triangleright$L7$\triangleright$L6$\gg$	$x \in \{u{\in}u_4(y) \mid \exists z\colon u \in z \mathbin{\dot\wedge} z \in y\} \mathbin{\dot\Leftrightarrow}$ $x \in u_4(y) \mathbin{\dot\wedge} \exists z\colon x \in z \mathbin{\dot\wedge} z \in y$	;
L09:	z-u3 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$\exists z\colon x \in z \mathbin{\dot\wedge} z \in y \equiv$ $x \in u_4(y) \mathbin{\dot\wedge} \exists z\colon x \in z \mathbin{\dot\wedge} z \in y$	;
L10:	Replace' $\triangleright$ L9 $\triangleright$ L8 $\gg$	$x \in \bigcup y \mathbin{\dot\Leftrightarrow} \exists z\colon x \in z \mathbin{\dot\wedge} z \in y$	;
L11:	Block $\gg$	End	;
L12:	Gen2 $\triangleright$ L10 $\gg$	$\forall x \forall y\colon (x \in \bigcup y \mathbin{\dot\Leftrightarrow} \exists z\colon x \in z \mathbin{\dot\wedge} z \in y)$	]*

[The Map proof of z-ExtUnion reads

L01:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(a)$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\ell\text{'}x$	;
L04:	Hypothesis $\gg$	$\ell\text{'}y$	;
L05:	z-SubtypeExtL' $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\gg$	$\ell\text{'}(a\text{'}y)$	;
L06:	z-TypeUnion $\trianglerighteq$ L5 $\gg$	$\ell\text{'}\bigcup a\text{'}y$	;
L07:	ElimAll2 $\trianglerighteq$ z-u $\trianglerighteq$ L3 $\trianglerighteq$ L5 $\gg$	$x \in \bigcup a\text{'}y \mathbin{\dot\Leftrightarrow} \exists z\colon x{\in}z \mathbin{\dot\wedge} z{\in}a\text{'}y$	;
L08:	z-ExtVarX $\trianglerighteq$ L3 $\gg$	$\mathrm{ZfcExt}'(\lambda y.x)$	;
L09:	z-ExtUnion' $\trianglerighteq$ L8 $\trianglerighteq$ L1 $\gg$	$\mathrm{ZfcExt}(\lambda y.\exists z\colon x{\in}z \mathbin{\dot\wedge} z{\in}a\text{'}y)$	;
L10:	Block $\gg$	End	;
L11:	z-IntroZfcExtIff'$\triangleright$L6$\triangleright$L7$\triangleright$L9$\gg$	$\mathrm{ZfcExt}'(\lambda y.\bigcup a\text{'}y)$	]*

A.63 Lemmas about $[\,\mathcal{P}x\,]$

$$\begin{array}{llll}
[\,x \subseteq y & \doteq & \forall z\colon (z \in x \Rightarrow z \in y) &]^{*} \\
[\,W(x) & \doteq & \lambda y.\mathrm{if}(y, \emptyset, \lambda z.x\,\text{'}(y\,\text{'}(x\,\text{'}\,z))) &]^{*} \\
[\,\overline{W}(x,y) & \doteq & \lambda z.\mathrm{if}(y\,\text{'}(y^{-1}\,\text{'}\,z) \in x, y^{-1}\,\text{'}\,z, \varepsilon v\colon y\,\text{'}\,v \in x) &]^{*} \\
[\,\tilde{W}(x,y,z) & \doteq & \lambda u.\mathrm{if}(u\,\text{'}\,z \in x, u\,\text{'}\,z, y\,\text{'}\,\varepsilon v\colon y\,\text{'}\,v \in x) & \\
[\,\mathcal{P}x & \doteq & \{u \in W(x) | \forall v\colon (v \in u \Rightarrow v \in x)\} &]^{*}
\end{array}$$

$$\begin{array}{lll}
[\ \text{Map lemma} & & \\
\text{z-TypePower'}_{256}\colon & \ell x\colon \ell\,\text{'}\,W(x) & ; \\
\text{z-ExtPower'}_{256}\colon & \mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(b) \to & \\
 & \mathrm{ZfcExt}(\lambda u.\forall v\colon (v \in a\,\text{'}\,u \Rightarrow v \in b\,\text{'}\,u)) & ; \\
\text{z-TypePower}_{257}\colon & \ell x\colon \ell\,\text{'}\,\mathcal{P}x & ; \\
\text{z-TypePower2}_{257}\colon & \ell x,y\colon \ell\,\text{'}\,\overline{W}(x,y) & ; \\
\text{z-w7}_{258}\colon & \ell x,y,z\colon W(y)\text{'}\overline{W}(x,y)\text{'}z \equiv \tilde{W}(x,y,z)\,\text{'}(y \circ y^{-1} \circ y) & ; \\
\text{z-w8}_{258}\colon & \ell x,y,z\colon \ell_1 \tilde{W}(x,y,z) & ; \\
\text{z-w3}_{259}\colon & \ell x,y,z\colon \neg x \to x \subseteq y \to & \\
 & W(y)\text{'}\overline{W}(x,y)\text{'}z \sim \mathrm{if}(y\text{'}z{\in}x, y\text{'}z, y\text{'}\varepsilon z\colon y\text{'}z{\in}x) & ; \\
\text{z-w4}_{260}\colon & \ell x,y,u\colon \neg x \to x \subseteq y \to u \in x \to u \in W(y)\,\text{'}\,\overline{W}(x,y) & ; \\
\text{z-w9}_{261}\colon & \ell x,y\colon \neg x \to x \subseteq y \to \exists v\colon y\,\text{'}\,v \in x & ; \\
\text{z-w5}_{262}\colon & \ell x,y,u\colon \neg x \to x \subseteq y \to u \in W(y)\,\text{'}\,\overline{W}(x,y) \to u \in x & ; \\
\text{z-w6}_{263}\colon & \ell x,y\colon \neg x \to x \subseteq y \to x = W(y)\,\text{'}\,\overline{W}(x,y) & ; \\
\text{z-w1}_{263}\colon & \ell x,y\colon x \subseteq y \to x \in W(y)) & ; \\
\text{z-w2}_{264}\colon & \ell x,y\colon x \subseteq y \equiv x \in W(y) \mathbin{\dot{\wedge}} x \subseteq y & ; \\
\text{z-w}_{264}\colon & \forall x \forall y\colon (x \in \mathcal{P}y \mathrel{\dot{\Leftrightarrow}} \forall z\colon (z \in x \Rightarrow z \in y)) & ; \\
\text{z-ExtPower}_{264}\colon & \mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(\lambda y.\mathcal{P}a\,\text{'}\,y) &]^{*}
\end{array}$$

[The Map proof of z-TypePower' reads

L01:	Hypothesis $\gg$	$\ell\text{'}x$	;
L02:	ExistsHull $\trianglerighteq$ L1 $\gg$	$\exists p\colon x \in_2 p \wedge \mathrm{Tr}(p)$	;
L03:	Define $\gg$	$p \equiv \varepsilon p\colon x \in_2 p \wedge \mathrm{Tr}(p)$	;
L04:	ElimExistsEll $\trianglerighteq$ L2 $\gg$	$\ell\text{'}p$	;
L05:	ElimExists $\trianglerighteq$ L2 $\gg$	$x \in_2 p \wedge \mathrm{Tr}(p)$	;
L06:	Logic $\trianglerighteq$ L5 $\gg$	$x \in_2 p$	;
L07:	Logic $\trianglerighteq$ L5 $\gg$	$\mathrm{Tr}(p)$	;
L08:	Block $\gg$	Begin	;
L09:	Hypothesis $\gg$	$\quad y \sim_p \underline{y}$	;
L10:	Block $\gg$	$\quad$ Begin	;
L11:	Hypothesis $\gg$	$\quad\quad z \sim_p \underline{z}$	;
L12:	RefEquivK $\trianglerighteq$ L1 $\gg$	$\quad\quad x \sim x$	;
L13:	ElimTrInEll $\trianglerighteq$ L7 $\trianglerighteq$ L6 $\gg$	$\quad\quad x \in_\ell p$	;
L14:	StrictEquivPX $\trianglerighteq$ L11 $\gg$	$\quad\quad \ell\text{'}z$	;
L15:	ElimTrInTwo $\trianglerighteq$ L14 $\trianglerighteq$ L7 $\trianglerighteq$ L6 $\gg$	$\quad\quad x\text{'}z \in_2 p$	;
L16:	ExtKApplyP$\trianglerighteq$L13$\trianglerighteq$L12$\trianglerighteq$L11$\gg$	$\quad\quad x\text{'}z \sim$	
		$\quad\quad x\text{'}\underline{z}$	;
L17:	ExtPApplyK $\trianglerighteq$ L15 $\trianglerighteq$ L9 $\trianglerighteq$ L16 $\gg$	$\quad\quad y\text{'}(x\text{'}z) \sim_p$	
		$\quad\quad \underline{y}\text{'}(x\text{'}\underline{z})$	;
L18:	ExtKApplyP$\trianglerighteq$L13$\trianglerighteq$L12$\trianglerighteq$L17$\gg$	$\quad\quad x\text{'}(y\text{'}(x\text{'}z)) \sim$	
		$\quad\quad x\text{'}(\underline{y}\text{'}(x\text{'}\underline{z}))$	;
L19:	ExtT $\gg$	$\quad\quad \mathsf{T} \sim \mathsf{T}$	;
L20:	ExtIfPKK $\trianglerighteq$ L9 $\trianglerighteq$ L19 $\trianglerighteq$ L18 $\gg$	$\quad\quad \mathrm{if}(y, \emptyset, x\text{'}(y\text{'}(x\text{'}z))) \sim$	
		$\quad\quad \mathrm{if}(\underline{y}, \emptyset, x\text{'}(\underline{y}\text{'}(x\text{'}\underline{z})))$	;
L21:	Block $\gg$	$\quad$ End	;
L22:	ExtLambda $\triangleright$ L4 $\triangleright$ L18 $\gg$	$\quad \lambda z.\mathrm{if}(y, \emptyset, x\text{'}(y\text{'}(x\text{'}z))) \sim$	
		$\quad \lambda \underline{z}.\mathrm{if}(\underline{y}, \emptyset, x\text{'}(\underline{y}\text{'}(x\text{'}\underline{z})))$	;
L23:	Block $\gg$	End	;
L24:	ExtLambda $\triangleright$ L4 $\triangleright$ L20 $\gg$	$\lambda y.\lambda z.\mathrm{if}(y, \emptyset, x\text{'}(y\text{'}(x\text{'}z))) \sim$	
		$\lambda \underline{y}.\lambda \underline{z}.\mathrm{if}(\underline{y}, \emptyset, x\text{'}(\underline{y}\text{'}(x\text{'}\underline{z})))$	;
L25:	StrictEquivKX $\trianglerighteq$ L22 $\gg$	$\lambda y.\lambda z.\mathrm{if}(y, \emptyset, x\text{'}(y\text{'}(x\text{'}z)))$	]*

[The Map proof of z-ExtPower' reads

L01:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(a)$	;
L02:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(b)$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$\quad \ell\text{'}v$	;
L05:	z-ExtVarX $\trianglerighteq$ L4 $\gg$	$\quad \mathrm{ZfcExt}'(\lambda u.v)$	;
L06:	z-ExtIn $\trianglerighteq$ L5 $\trianglerighteq$ L1 $\gg$	$\quad \mathrm{ZfcExt}(\lambda u.v \in a\text{'}u)$	;
L07:	z-ExtIn $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\gg$	$\quad \mathrm{ZfcExt}(\lambda u.v \in b\text{'}u)$	;
L08:	z-ExtImply $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$	$\quad \mathrm{ZfcExt}(\lambda u.v \in a\text{'}u \Rightarrow v \in b\text{'}u)$	;
L09:	Block $\gg$	End	;
L10:	z-ExtAll $\triangleright$ L8 $\gg$	$\mathrm{ZfcExt}(\lambda u.\forall v\colon (v \in a\text{'}u \Rightarrow v \in b\text{'}u)$	]*

[The Map proof of z-TypePower reads

L1: Hypothesis $\gg$	$\ell\,'\,x$	;
L2: z-TypePower' $\trianglerighteq$ L1 $\gg$	$\ell\,'\,W(x)$	;
L3: z-ExtVarY $\gg$	$\mathrm{ZfcExt}'(\lambda u.u)$	;
L4: z-ExtVarX $\trianglerighteq$ L1 $\gg$	$\mathrm{ZfcExt}'(\lambda u.x)$	;
L5: z-ExtPower' $\trianglerighteq$ L3 $\trianglerighteq$ L4 $\gg$	$\mathrm{ZfcExt}(\lambda u.\forall v\colon (v \in a\,'\,u \Rightarrow v \in b\,'\,u))$	;
L6: z-TypeSubset $\triangleright$ L2 $\triangleright$ L5 $\gg$	$\ell\,'\{u \in W(x) \mid \forall v\colon (v{\in}a'u{\Rightarrow}v{\in}b'u)\}$	;
L7: Repetition $\triangleright$ L6 $\gg$	$\ell\,'\,\mathcal{P}x$	]*

[The Map proof of z-TypePower2 reads

L01: Hypothesis $\gg$	$\ell\,'\,x$	;
L02: Hypothesis $\gg$	$\ell\,'\,y$	;
L03: Define $\gg$	$w \equiv \lambda z.\mathrm{if}(y'z{\in}x, z, \varepsilon v\colon y'v{\in}x)$	;
L04: Reduction $\gg$	$\overline{W}(x,y) \equiv w \circ y^{-1}$	;
L05: TypeInv $\trianglerighteq$ L2 $\gg$	$\ell\,'\,y^{-1}$	;
L06: Block $\gg$	Begin	;
L07: Hypothesis $\gg$	$\quad \ell\,'\,z$	;
L08: TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L7 $\gg$	$\quad \ell\,'(y\,'\,z)$	;
L09: z-TypeIn $\trianglerighteq$ L8 $\trianglerighteq$ L1 $\gg$	$\quad !y\,'\,z \in x$	;
L10: Block $\gg$	$\quad$ Begin	;
L11: Hypothesis $\gg$	$\qquad \ell\,'\,v$	;
L12: TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L11 $\gg$	$\qquad \ell\,'(y\,'\,v)$	;
L13: z-TypeIn $\trianglerighteq$ L12 $\trianglerighteq$ L1 $\gg$	$\qquad !y\,'\,v \in x$	;
L14: Block $\gg$	$\quad$ End	;
L15: TypeChoice $\trianglerighteq$ (Gen $\triangleright$ L13) $\gg$	$\quad \ell\,'\,\varepsilon v\colon y\,'\,v \in x$	;
L16: TypeIfBLL $\trianglerighteq$ L9 $\trianglerighteq$ L7 $\trianglerighteq$ L15 $\gg$	$\quad \ell\,'\,\mathrm{if}(y'z{\in}x, z, \varepsilon v\colon y'v{\in}x)$	;
L17: Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L16 $\gg$	$\quad \ell\,'(w\,'\,z)$	;
L18: Block $\gg$	End	;
L19: TypeCompose$\trianglerighteq$(Gen$\triangleright$L17)$\trianglerighteq$L5$\gg$	$\ell\,'(w \circ y^{-1})$	;
L20: Replace' $\triangleright$ L4 $\triangleright$ L19 $\gg$	$\overline{W}(x,y)$	]*

[The Map proof of z-w7 reads

L01:	Hypothesis $\gg$	$\ell ' x$	;
L02:	Hypothesis $\gg$	$\ell ' y$	;
L03:	Hypothesis $\gg$	$\ell ' z$	;
L04:	TypeInv $\trianglerighteq$ L2 $\gg$	$\ell ' y^{-1}$	;
L05:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$\ell '(y ' z)$	;
L06:	TypeApply $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\gg$	$\ell '(y^{-1} '(y ' z))$	;
L07:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L6 $\gg$	$\ell '(y '(y^{-1} '(y ' z)))$	;
L08:	z-TypeIn $\trianglerighteq$ L7 $\trianglerighteq$ L1 $\gg$	$!y '(y^{-1} '(y ' z)) \in x$	;
L09:	Block $\gg$	Begin	;
L10:	Algebra $\gg$	$\quad W(y) ' \overline{W}(x,y) ' z$	;
L11:	Reduction $\gg$	$\quad y' \left(y'(y^{-1}'(y'z)) \in x \begin{cases} y^{-1}'(y'z) \\ \varepsilon v \colon y'v \in x \end{cases} \right)$	;
L12:	Logic $\trianglerighteq$ L8 $\gg$	$\quad y'(y^{-1}'(y'z)) \in x \begin{cases} y'(y^{-1}'(y'z)) \\ y'\varepsilon v \colon y'v \in x \end{cases}$	;
L13:	Reduction $\gg$	$\quad \tilde{W}(x,y,z) '(y \circ y^{-1} \circ y)$	;
L14:	Block $\gg$	End	]*

[The Map proof of z-w8 reads

L01:	Hypothesis $\gg$	$\ell ' x$	;
L02:	Hypothesis $\gg$	$\ell ' y$	;
L03:	Hypothesis $\gg$	$\ell ' z$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\quad \ell ' u$	;
L06:	TypeApply $\trianglerighteq$ L5 $\trianglerighteq$ L3 $\gg$	$\quad \ell '(u ' z)$	;
L07:	z-TypeIn $\trianglerighteq$ L6 $\trianglerighteq$ L1 $\gg$	$\quad !u ' z \in x$	;
L08:	Block $\gg$	$\quad$ Begin	;
L09:	Hypothesis $\gg$	$\qquad \ell ' v$	;
L10:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L9 $\gg$	$\qquad \ell '(y ' v)$	;
L11:	z-TypeIn $\trianglerighteq$ L10 $\trianglerighteq$ L1 $\gg$	$\qquad !y ' v \in x$	;
L12:	Block $\gg$	$\quad$ End	;
L13:	TypeChoice $\trianglerighteq$ (Gen $\rhd$ L11) $\gg$	$\quad \ell ' \varepsilon v \colon y ' v \in x$	;
L14:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L13 $\gg$	$\quad \ell '(y ' \varepsilon v \colon y ' v \in x)$	;
L15:	TypeIfBLL$\trianglerighteq$L7$\trianglerighteq$L6$\trianglerighteq$L14$\gg$	$\quad \ell ' \left(u ' z \in x \begin{cases} u ' z \\ y ' \varepsilon v \colon y'v \in x \end{cases} \right)$	;
L16:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L15 $\gg$	$\quad \ell '(\tilde{W}(x,y,z) ' u)$	;
L17:	Block $\gg$	End	;
L18:	Gen $\rhd$ L16 $\gg$	$!_1 \tilde{W}(x,y,z)$	]*

[The Map proof of z-w3 reads

L01:	Hypothesis $\gg$	$\ell\,'\,x$	;
L02:	Hypothesis $\gg$	$\ell\,'\,y$	;
L03:	Hypothesis $\gg$	$\ell\,'\,z$	;
L04:	Hypothesis $\gg$	$\neg x$	;
L05:	Hypothesis $\gg$	$x \subseteq y$	;
L06:	z-Member$\trianglerighteq$L1$\trianglerighteq$L1$\trianglerighteq$L4$\gg$	$x\,'\,x \in x$	;
L07:	TypeApply $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\gg$	$\ell\,'(x\,'\,x)$	;
L08:	ElimAll $\trianglerighteq$ L5 $\trianglerighteq$ L7 $\gg$	$x\,'\,x \in x \Rightarrow x\,'\,x \in y$	;
L09:	Logic $\trianglerighteq$ L6 $\trianglerighteq$ L8 $\gg$	$x\,'\,x \in y$	;
L10:	z-ElimInNot $\trianglerighteq$ L9 $\gg$	$\neg y$	;
L11:	z-w8 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$!_1 \tilde{W}(x,y,z)$	;
L12:	IntroInv $\trianglerighteq$ L2 $\trianglerighteq$ L10 $\gg$	$y \circ y^{-1} \circ y \sim y$	;
L13:	Ext $\trianglerighteq$ L11 $\trianglerighteq$ L12 $\gg$	$\tilde{W}\,'(y \circ y^{-1} \circ y) \sim \tilde{W}(x,y,z)\,'\,y$	;
L14:	z-w7 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$W(y)\,'\,\overline{W}(x,y)\,'\,z \equiv$ $\tilde{W}(x,y,z)\,'(y \circ y^{-1} \circ y)$	;
L15:	Replace' $\triangleright$ L14 $\triangleright$ L13 $\gg$	$W(y)\,'\,\overline{W}(x,y)\,'\,z \sim \tilde{W}(x,y,z)\,'\,y$	;
L16:	Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L15 $\gg$	$W(y)\,'\,\overline{W}(x,y)\,'\,z \sim$ $\mathrm{if}(y\,'\,z \in x, y\,'\,z, y\,'\,\varepsilon v \colon y\,'\,v \in x)$	]*

[The Map proof of z-w4 reads

L01:	Hypothesis ≫	$\ell\,\text{'}\,x$	;
L02:	Hypothesis ≫	$\ell\,\text{'}\,y$	;
L03:	Hypothesis ≫	$\ell\,\text{'}\,u$	;
L04:	Hypothesis ≫	$\neg x$	;
L05:	Hypothesis ≫	$x \subseteq y$	;
L06:	Hypothesis ≫	$u \in x$	;
L07:	ElimAll ⊵ L5 ⊵ L3 ≫	$u \in x \Rightarrow u \in y$	;
L08:	Logic ⊵ L6 ⊵ L7 ≫	$u \in y$	;
L09:	Define ≫	$z \equiv \varepsilon z\colon u = y\,\text{'}\,z$	;
L10:	z-ElimInEll ⊵ L6 ≫	$\ell\,\text{'}\,z$	;
L11:	z-ElimIn ⊵ L6 ≫	$u = y\,\text{'}\,z$	;
L12:	TypeApply ⊵ L2 ⊵ L10 ≫	$\ell\,\text{'}(y\,\text{'}\,z)$	;
L13:	z-ComEqual ⊵ L3 ⊵ L12 ⊵ L11 ≫	$y\,\text{'}\,z = u$	;
L14:	z-ext1⊵L12⊵L3⊵L1⊵L13⊵L6≫	$y\,\text{'}\,z \in x$	;
L15:	Define ≫	$w{\equiv}\mathrm{if}(y\text{'}z{\in}x, y\text{'}z, y\text{'}\varepsilon v\colon y\text{'}v{\in}x)$	;
L16:	Logic ⊵ L14 ≫	$y\,\text{'}\,z \equiv w$	;
L17:	Replace' ▷ L16 ▷ L11 ≫	$u = w$	;
L18:	z-w3⊵L1⊵L2⊵L10⊵L4⊵L5≫	$W(y)\,\text{'}\,\overline{W}(x,y)\,\text{'}\,z \sim w$	;
L19:	RefEquivK ⊵ L18 ≫	$w \sim W(y)\,\text{'}\,\overline{W}(x,y)\,\text{'}\,z$	;
L20:	z-RefEqual' ⊵ L19 ≫	$w = W(y)\,\text{'}\,\overline{W}(x,y)\,\text{'}\,z$	;
L21:	StrictEquivKX ⊵ L19 ≫	$\ell\,\text{'}\,w$	;
L22:	StrictEquivKY ⊵ L19 ≫	$\ell\,\text{'}(W(y)\,\text{'}\,\overline{W}(x,y)\,\text{'}\,z)$	;
L23:	z-TransEqual ⊵		
	L3 ⊵ L21 ⊵ L22 ⊵ L17 ⊵ L20 ≫	$u = W(y)\,\text{'}\,\overline{W}(x,y)\,\text{'}\,z$	;
L24:	Reduction ≫	$\neg W(y)\,\text{'}\,\overline{W}(x,y)$	;
L25:	z-TypePower' ⊵ L2 ≫	$\ell\,\text{'}\,W(y)$	;
L26:	z-TypePower2 ⊵ L1 ⊵ L2 ≫	$\ell\,\text{'}\,\overline{W}(x,y)$	;
L27:	TypeApply ⊵ L25 ⊵ L26 ≫	$\ell\,\text{'}(W(y)\,\text{'}\,\overline{W}(x,y))$	;
L28:	z-IntroIn ⊵		
	L3 ⊵ L27 ⊵ L10 ⊵ L24 ⊵ L23 ≫	$u \in W(y)\,\text{'}\,\overline{W}(x,y)$	]*

[The Map proof of z-w9 reads

L01:	Hypothesis $\gg$	$\ell \text{'} x$	;
L02:	Hypothesis $\gg$	$\ell \text{'} y$	;
L03:	Hypothesis $\gg$	$\neg x$	;
L04:	Hypothesis $\gg$	$x \subseteq y$	;
L05:	z-Member $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$x \text{'} x \in x$	;
L06:	TypeAll $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\gg$	$\ell \text{'}(x \text{'} x)$	;
L07:	ElimAll $\trianglerighteq$ L4 $\trianglerighteq$ L6 $\gg$	$x \text{'} x \in x \Rightarrow x \text{'} x \in y$	;
L08:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L7 $\gg$	$x \text{'} x \in y$	;
L09:	Define $\gg$	$s \equiv \varepsilon s\colon x \text{'} x = y \text{'} s$	;
L10:	z-ElimInEll $\trianglerighteq$ L8 $\gg$	$\ell \text{'} s$	;
L11:	z-ElimIn $\trianglerighteq$ L8 $\gg$	$x \text{'} x = y \text{'} s$	;
L12:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L10 $\gg$	$\ell \text{'}(y \text{'} s)$	;
L13:	z-ComEqual $\trianglerighteq$ L6 $\trianglerighteq$ L12 $\trianglerighteq$ L11 $\gg$	$y \text{'} s = x \text{'} x$	;
L14:	z-ext1 $\trianglerighteq$ L12 $\trianglerighteq$ L6 $\trianglerighteq$ L1 $\trianglerighteq$ L13 $\trianglerighteq$ L8 $\gg$	$y \text{'} s \in x$	;
L15:	Block $\gg$	Begin	;
L16:	Hypothesis $\gg$	$\quad \ell \text{'} v$	;
L17:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L16 $\gg$	$\quad \ell \text{'}(y \text{'} v)$	;
L18:	z-TypeIn $\trianglerighteq$ L17 $\trianglerighteq$ L1 $\gg$	$\quad !y \text{'} v \in x$	;
L19:	Block $\gg$	End	;
L20:	IntroExists $\trianglerighteq$ (Gen $\triangleright$ L18) $\trianglerighteq$ L10 $\trianglerighteq$ L14 $\gg$	$\exists v\colon y \text{'} v \in x$	]*

[The Map proof of z-w5 reads

L01:	Hypothesis $\gg$	$\ell\,'\,x$	;
L02:	Hypothesis $\gg$	$\ell\,'\,y$	;
L03:	Hypothesis $\gg$	$\ell\,'\,u$	;
L04:	Hypothesis $\gg$	$\neg x$	;
L05:	Hypothesis $\gg$	$x \subseteq y$	;
L06:	Hypothesis $\gg$	$u \in W(y)\,'\,\overline{W}(x,y)$	;
L07:	Define $\gg$	$z \equiv \varepsilon z\colon u = W(y)\,'\,\overline{W}(x,y)\,'\,z$	;
L08:	z-ElimInEll $\trianglerighteq$ L6 $\gg$	$\ell\,'\,z$	;
L09:	z-ElimIn $\trianglerighteq$ L6 $\gg$	$u = W(y)\,'\,\overline{W}(x,y)\,'\,z$	;
L10:	Define $\gg$	$w{\equiv}\mathrm{if}(y'z{\in}x, y'z, y'\varepsilon v\colon y'v{\in}x)$	;
L11:	z-w3 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L8 $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\gg$	$W(y)\,'\,\overline{W}(x,y)\,'\,z \sim w$	;
L12:	z-RefEqual' $\trianglerighteq$ L11 $\gg$	$W(y)\,'\,\overline{W}(x,y)\,'\,z = w$	;
L13:	StrictEquivKX $\trianglerighteq$ L11 $\gg$	$\ell\,'(W(y)\,'\,\overline{W}(x,y)\,'\,z)$	;
L14:	StrictEquivKY $\trianglerighteq$ L11 $\gg$	$\ell\,'\,w$	;
L15:	z-TransEqual $\trianglerighteq$ L3 $\trianglerighteq$ L13 $\trianglerighteq$ L14 $\trianglerighteq$ L9 $\trianglerighteq$ L12 $\gg$	$u = w$	;
L16:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L8 $\gg$	$\ell\,'(y\,'\,z)$	;
L17:	z-TypeIn $\trianglerighteq$ L16 $\trianglerighteq$ L1 $\gg$	$!y\,'\,z \in x$	;
L18:	Block $\gg$	Begin	;
L19:	Hypothesis $\gg$	$y\,'\,z \in x$	;
L20:	Logic $\trianglerighteq$ L19 $\gg$	$y\,'\,z \equiv w$	;
L21:	Replace' $\triangleright$ L20 $\triangleright$ L19 $\gg$	$w \in x$	;
L22:	Block $\gg$	End	;
L23:	Block $\gg$	Begin	;
L24:	Hypothesis $\gg$	$\neg y\,'\,z \in x$	;
L25:	z-w9 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\gg$	$\exists v\colon y\,'\,v \in x$	;
L26:	Define $\gg$	$v \equiv \varepsilon v\colon y\,'\,v \in x$	;
L27:	ElimExists $\trianglerighteq$ L25 $\gg$	$y\,'\,v \in x$	;
L28:	Logic $\trianglerighteq$ L24 $\gg$	$w \equiv y\,'\,v$	;
L29:	Replace' $\triangleright$ L28 $\triangleright$ L27 $\gg$	$w \in x$	;
L30:	Block $\gg$	End	;
L31:	TND $\triangleright$ L17 $\triangleright$ L21 $\triangleright$ L29 $\gg$	$w \in x$	;
L32:	z-ext1$\trianglerighteq$L3$\trianglerighteq$L14$\trianglerighteq$L1$\trianglerighteq$L15$\trianglerighteq$L31$\gg$	$u \in x$	]*

[The Map proof of z-w6 reads

L01:	Hypothesis $\gg$	$\ell\,'x$	;
L02:	Hypothesis $\gg$	$\ell\,'y$	;
L03:	Hypothesis $\gg$	$\neg x$	;
L04:	Hypothesis $\gg$	$x \subseteq y$	;
L05:	z-TypePower' $\trianglerighteq$ L2 $\gg$	$\ell\,'W(y)$	;
L06:	z-TypePower2 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\,'\overline{W}(x,y)$	;
L07:	TypeApply $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$\ell\,'(W(y)\,'\,\overline{W}(x,y))$	;
L08:	Block $\gg$	Begin	;
L09:	Hypothesis $\gg$	$\ell\,'u$	;
L10:	z-TypeIn $\trianglerighteq$ L9 $\trianglerighteq$ L1 $\gg$	$!u \in x$	;
L11:	z-TypeIn $\trianglerighteq$ L9 $\trianglerighteq$ L7 $\gg$	$!u \in W(y)\,'\,\overline{W}(x,y)$	;
L12:	z-w4$\trianglerighteq$L1$\trianglerighteq$L2$\trianglerighteq$L9$\trianglerighteq$L3$\trianglerighteq$L4$\gg$	$u \in x \rightarrow u \in W(y)\,'\,\overline{W}(x,y)$	;
L13:	z-w5$\trianglerighteq$L1$\trianglerighteq$L2$\trianglerighteq$L9$\trianglerighteq$L3$\trianglerighteq$L4$\gg$	$u \in W(y)\,'\,\overline{W}(x,y) \rightarrow u \in x$	;
L14:	CDeduction$\triangleright$L10$\triangleright$L11$\triangleright$L12 $\gg$	$u \in x \Rightarrow u \in W(y)\,'\,\overline{W}(x,y)$	;
L15:	CDeduction$\triangleright$L11$\triangleright$L10$\triangleright$L13 $\gg$	$u \in W(y)\,'\,\overline{W}(x,y) \Rightarrow u \in x$	;
L16:	Logic $\trianglerighteq$ L14 $\trianglerighteq$ L15 $\gg$	$u \in x \Leftrightarrow u \in W(y)\,'\,\overline{W}(x,y)$	;
L17:	Block $\gg$	End	;
L18:	z-ext3$\trianglerighteq$L1$\trianglerighteq$L7$\trianglerighteq$(Gen$\triangleright$L16)$\gg$	$x = W(y)\,'\,\overline{W}(x,y)$	]*

[The Map proof of z-w1 reads

L01:	Hypothesis $\gg$	$\ell\,'x$	;
L02:	Hypothesis $\gg$	$\ell\,'y$	;
L03:	Hypothesis $\gg$	$x \subseteq y$	;
L04:	SubtypeLB $\trianglerighteq$ L1 $\gg$	$!x$	;
L05:	Reduction $\gg$	$\neg W(y)$	;
L06:	z-TypePower' $\trianglerighteq$ L2 $\gg$	$\ell\,'W(y)$	;
L07:	Block $\gg$	Begin	;
L08:	Hypothesis $\gg$	x	;
L09:	TypeT $\gg$	$\ell\,'\mathsf{T}$	;
L10:	Logic $\trianglerighteq$ L8 $\gg$	$x = W(y)\,'\,\mathsf{T}$	;
L11:	z-IntroIn $\trianglerighteq$ L1 $\trianglerighteq$ L6 $\trianglerighteq$ L9 $\trianglerighteq$ L5 $\trianglerighteq$ L10 $\gg$	$x \in W(y)$	;
L12:	Block $\gg$	End	;
L13:	Block $\gg$	Begin	;
L14:	Hypothesis $\gg$	$\neg x$	;
L15:	z-w6 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\trianglerighteq$ L14 $\trianglerighteq$ L3 $\gg$	$x = W(y)\,'\,\overline{W}(x,y)$	;
L16:	z-TypePower2 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\,'\overline{W}(x,y)$	;
L17:	z-IntroIn $\trianglerighteq$ L1 $\trianglerighteq$ L6 $\trianglerighteq$ L16 $\trianglerighteq$ L5 $\trianglerighteq$ L15 $\gg$	$x \in W(y)$	;
L18:	Block $\gg$	End	;
L19:	TND $\triangleright$ L4 $\triangleright$ L11 $\triangleright$ L17 $\gg$	$x \in W(y)$	]*

[The Map proof of z-w2 reads

L01:	Hypothesis $\gg$	$\ell\,'\,x$	;
L02:	Hypothesis $\gg$	$\ell\,'\,y$	;
L03:	z-TypePower' $\trianglerighteq$ L2 $\gg$	$\ell\,'\,W(y)$	;
L04:	z-TypeIn $\trianglerighteq$ L1 $\trianglerighteq$ L3 $\gg$	$!x \in W(y)$	;
L05:	Block $\gg$	Begin	;
L06:	Algebra $\gg$	$x \subseteq y$	;
L07:	Logic $\gg$	$x \subseteq y \mathrel{\tilde{\wedge}} \mathsf{T}$	;
L08:	Rev $\triangleright$ z-w1 $\gg$	$x \subseteq y \mathrel{\tilde{\wedge}} x \in W(y)$	;
L09:	Logic $\trianglerighteq$ L4 $\gg$	$x \in W(y) \mathrel{\dot{\wedge}} x \subseteq y$	;
L10:	Block $\gg$	End	]*

[The Map proof of z-w reads

L01:	Block $\gg$	Begin	;
L02:	Hypothesis $\gg$	$\ell\,'\,x$	;
L03:	Hypothesis $\gg$	$\ell\,'\,y$	;
L04:	z-ExtVarY $\gg$	$\mathrm{ZfcExt}'(\lambda u.u)$	;
L05:	z-ExtVarX $\trianglerighteq$ L3 $\gg$	$\mathrm{ZfcExt}'(\lambda u.y)$	;
L06:	z-ExtPower' $\trianglerighteq$ L4 $\trianglerighteq$ L5 $\gg$	$\mathrm{ZfcExt}(\lambda u.\forall z\colon (z \in x \Rightarrow z \in y))$	;
L07:	z-TypePower' $\trianglerighteq$ L3 $\gg$	$\ell\,'\,W(y)$	;
L08:	z-Subset $\triangleright$ L2 $\triangleright$ L7 $\triangleright$ L6 $\gg$	$x \in \{u{\in}W(y) \mid \forall z\colon (z{\in}u \Rightarrow z{\in}y)\} \Leftrightarrow$ $x \in W(y) \mathrel{\dot{\wedge}} \forall z\colon (z \in x \Rightarrow z \in y)$	;
L09:	z-w2 $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$\forall z\colon (z \in x \Rightarrow z \in y) \equiv$ $x \in W(y) \mathrel{\dot{\wedge}} \forall z\colon (z \in x \Rightarrow z \in y)$	;
L10:	Replace' $\triangleright$ L9 $\triangleright$ L8 $\gg$	$x \in \mathcal{P}y \Leftrightarrow \forall z\colon (z \in x \Rightarrow z \in y)$	;
L11:	Block $\gg$	End	;
L12:	Gen2 $\triangleright$ L10 $\gg$	$\forall x \forall y\colon (x \in \mathcal{P}y \Leftrightarrow \forall z\colon (z \in x \Rightarrow z \in y))$	]*

[The Map proof of z-ExtPower reads

L01:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(a)$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\ell\,'\,x$	;
L04:	Hypothesis $\gg$	$\ell\,'\,y$	;
L05:	z-SubtypeExtL' $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\gg$	$\ell\,'(a\,'\,y)$	;
L06:	z-TypePower $\trianglerighteq$ L5 $\gg$	$\ell\,'\,\mathcal{P}a\,'\,y$	;
L07:	ElimAll2 $\trianglerighteq$ z-w $\trianglerighteq$ L3 $\trianglerighteq$ L5 $\gg$	$x \in \mathcal{P}a'y \Leftrightarrow \forall z\colon (z{\in}x{\Rightarrow}z{\in}a'y)$	;
L08:	z-ExtVarX $\trianglerighteq$ L3 $\gg$	$\mathrm{ZfcExt}'(\lambda y.x)$	;
L09:	z-ExtPower' $\trianglerighteq$ L8 $\trianglerighteq$ L1 $\gg$	$\mathrm{ZfcExt}(\lambda y.\forall z\colon (z{\in}x{\Rightarrow}z{\in}a'y))$	;
L10:	Block $\gg$	End	;
L11:	z-IntroZfcExtIff' $\triangleright$ L6 $\triangleright$ L7 $\triangleright$ L9 $\gg$	$\mathrm{ZfcExt}'(\lambda y.\mathcal{P}a\,'\,y)$	]*

A.64 Lemmas about $[\,\omega\,]$

$[\ x \cup y$	$\doteq$	$\bigcup\{x, y\}$	$]$
$[\ \{x\}$	$\doteq$	$\{x, x\}$	$]$
$[\ x^+$	$\doteq$	$x \cup \{x\}$	$]$
$[\ \omega$	$\doteq$	$\lambda x.\mathrm{if}(x, \emptyset, (\omega\,'(x\,'\,\mathsf{N}))^+)$	$]$

[Map lemma

z-TypeU$_{265}$:	$\ell x, y{:}\, \ell\,\text{'}(x \cup y)$	;
z-TypeUnit$_{265}$:	$\ell x{:}\, \ell\,\text{'}\{x\}$	;
z-TypeSuc$_{265}$:	$\ell x{:}\, \ell\,\text{'}(x^{+})$	;
z-TypeInfty'$_{265}$:	$\forall v{:}\, (\mathsf{T} \sim_{\mathsf{T}} v \dot{\Rightarrow} \omega\,\text{'}\,\mathsf{T} \sim \omega\,\text{'}\,v)$	;
z-TypeInfty''$_{266}$:	$\neg u \to \ell\,\text{'}\,u \to \forall y{:} \forall v{:}\, (u\text{'}y \sim_{\mathsf{T}} v \dot{\Rightarrow} \omega\,\text{'}(u\text{'}y) \sim \omega\,\text{'}\,v) \to$ $\forall v{:}\, (u \sim_{\mathsf{T}} v \dot{\Rightarrow} \omega\,\text{'}\,u \sim \omega\,\text{'}\,v)$	;
z-TypeInfty$_{266}$:	$\ell\,\text{'}\,\omega$	;
z-ExtU$_{266}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(b) \to \mathrm{ZfcExt}'(\lambda y.a\,\text{'}\,y \cup b\,\text{'}\,y)$	;
z-ExtUnit$_{267}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(\lambda y.\{a\,\text{'}\,y\})$	;
z-ExtSuc$_{267}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(\lambda y.(a\,\text{'}\,y)^{+})$	;
z-ExtInfty$_{267}$:	$\mathrm{ZfcExt}'(\lambda y.\omega)$	;
z-i$_{267}$:	$\emptyset \in \omega \;\dot{\wedge}\; \forall x{:}\, (x \in \omega \Rightarrow x^{+} \in \omega)$	]*

[The Map proof of z-TypeU reads

L1:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L2:	Hypothesis $\gg$	$\ell\,\text{'}\,y$	;
L3:	z-TypePair $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\,\text{'}\{x, y\}$	;
L4:	z-TypeUnion $\trianglerighteq$ L3 $\gg$	$\ell\,\text{'}\bigcup\{x, y\}$	;
L5:	Repetition $\triangleright$ L4 $\gg$	$\ell\,\text{'}(x \cup y)$	]*

[The Map proof of z-TypeUnit reads

L1:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L2:	z-TypePair $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}\{x, x\}$	;
L3:	Repetition $\triangleright$ L2 $\gg$	$\ell\,\text{'}\{x\}$	]*

[The Map proof of z-TypeSuc reads

L1:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L2:	z-TypeUnit $\trianglerighteq$ L1 $\gg$	$\ell\,\text{'}\{x\}$	;
L3:	z-TypeU $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\ell\,\text{'}(x \cup \{x\})$	;
L4:	Repetition $\triangleright$ L3 $\gg$	$\ell\,\text{'}(x^{+})$	]*

[The Map proof of z-TypeInfty' reads

L1:	Block $\gg$	Begin	;
L2:	Hypothesis $\gg$	$\ell\,\text{'}\,v$	;
L3:	TypeT $\gg$	$\ell\,\text{'}\,\mathsf{T}$	;
L4:	TypeEquivP $\trianglerighteq$ L3 $\trianglerighteq$ L3 $\trianglerighteq$ L2 $\gg$	$!\mathsf{T} \sim_{\mathsf{T}} v$	;
L5:	Logic $\gg$	$\mathsf{T} \sim_{\mathsf{T}} v \to \omega\,\text{'}\,\mathsf{T} \sim \omega\,\text{'}\,v$	;
L6:	Deduction'' $\triangleright$ L4 $\triangleright$ L5 $\gg$	$\mathsf{T} \sim_{\mathsf{T}} v \dot{\Rightarrow} \omega\,\text{'}\,\mathsf{T} \sim \omega\,\text{'}\,v$	;
L7:	Block $\gg$	End	;
L8:	Gen $\triangleright$ L6 $\gg$	$\forall v{:}\, (\mathsf{T} \sim_{\mathsf{T}} v \dot{\Rightarrow} \omega\,\text{'}\,\mathsf{T} \sim \omega\,\text{'}\,v)$	]*

[The Map proof of z-TypeInfty” reads

L01:	Hypothesis $\gg$	$\neg u$	;
L02:	Hypothesis $\gg$	$\ell\,'u$	;
L03:	Hypothesis $\gg$	$\forall y{:}\forall v{:}(u\,'y \sim_{\mathsf{T}} v \Rrightarrow \omega\,'(u\,'y) \sim \omega\,'v)$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\ell\,'v$	;
L06:	TypeT $\gg$	$\ell\,'\mathsf{T}$	;
L07:	TypeEquivP $\unrhd$ L6 $\unrhd$ L2 $\unrhd$ L5 $\gg$	$!u \sim_{\mathsf{T}} v$	;
L08:	Block $\gg$	Begin	;
L09:	Hypothesis $\gg$	$u \sim_{\mathsf{T}} v$	;
L10:	IntroInTwoT $\gg$	$\mathsf{T} \in_2 \mathsf{T}$	;
L11:	ElimEquivP $\unrhd$ L9 $\unrhd$ L10 $\gg$	$u\,'\mathsf{T} \sim_{\mathsf{T}} v\,'\mathsf{T}$	;
L12:	TypeApply $\unrhd$ L5 $\unrhd$ L6 $\gg$	$\ell\,'(v\,'\mathsf{T})$	;
L13:	ElimAll2 $\unrhd$ L3 $\unrhd$ L2 $\unrhd$ L12 $\gg$	$u\,'\mathsf{T} \sim_{\mathsf{T}} v\,'\mathsf{T} \Rrightarrow \omega\,'(u\,'\mathsf{T}) \sim \omega\,'(v\,'\mathsf{T})$	;
L14:	Logic $\unrhd$ L11 $\unrhd$ L13 $\gg$	$\omega\,'(u\,'\mathsf{T}) \sim \omega\,'(v\,'\mathsf{T})$	;
L15:	Gen $\rhd$ z-TypeSuc $\gg$	$!_1\lambda x.x^+$	;
L16:	Ext $\unrhd$ L15 $\unrhd$ L14 $\gg$	$(\omega\,'(u\,'\mathsf{T}))^+ \sim (\omega\,'(v\,'\mathsf{T}))^+$	;
L17:	Logic $\unrhd$ L1 $\unrhd$ L9 $\gg$	$\neg v$	;
L18:	Logic $\unrhd$ L1 $\unrhd$ L17 $\gg$	$\omega\,'u \sim \omega\,'v \equiv (\omega\,'(u\,'\mathsf{T}))^+ \sim (\omega\,'(v\,'\mathsf{T}))^+$	;
L19:	Transitivity $\rhd$ L18 $\rhd$ L16 $\gg$	$\omega\,'u \sim \omega\,'v$	;
L20:	Block $\gg$	End	;
L21:	Deduction” $\rhd$ L7 $\rhd$ L19 $\gg$	$u \sim_{\mathsf{T}} v \Rrightarrow \omega\,'u \sim \omega\,'v$	;
L22:	Block $\gg$	End	;
L23:	Gen $\rhd$ L21 $\gg$	$\forall v{:}(u \sim_{\mathsf{T}} v \Rrightarrow \omega\,'u \sim \omega\,'v)$	]*

[The Map proof of z-TypeInfty reads

L01:	TypeT $\gg$	$\ell\,'\mathsf{T}$	;
L02:	Transfinite” $\rhd$ z-TypeInfty’ $\rhd$ z-TypeInfty” $\gg$	$\forall u{:}\forall v{:}(u \sim_{\mathsf{T}} v \Rrightarrow \omega\,'u \sim \omega\,'v)$	;
L03:	Block $\gg$	Begin	;
L04:	Hypothesis $\gg$	$u \sim_{\mathsf{T}} v$	;
L05:	StrictEquivPX $\unrhd$ L4 $\gg$	$\ell\,'u$	;
L06:	StrictEquivPY $\unrhd$ L4 $\gg$	$\ell\,'v$	;
L07:	ElimAll2 $\unrhd$ L2 $\unrhd$ L5 $\unrhd$ L6 $\gg$	$u \sim_{\mathsf{T}} v \Rrightarrow \omega\,'u \sim \omega\,'v$	;
L08:	Logic $\unrhd$ L4 $\unrhd$ L7 $\gg$	$\omega\,'u \sim \omega\,'v$	;
L09:	Block $\gg$	End	;
L10:	IntroInEll $\rhd$ L1 $\rhd$ L08 $\gg$	$\omega \in_\ell \mathsf{T}$	;
L11:	IntroEll $\unrhd$ L10 $\gg$	$\ell\,'\omega$	]*

[The Map proof of z-ExtU reads

L1:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(a)$	;
L2:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(b)$	;
L3:	z-ExtPair $\unrhd$ L1 $\unrhd$ L2 $\gg$	$\mathrm{ZfcExt}'(\lambda y.\{a\,'y, b\,'y\}$	;
L4:	z-ExtUnion $\unrhd$ L3 $\gg$	$\mathrm{ZfcExt}'(\lambda y.a\,'y \cup b\,'y)$	]*

[The Map proof of z-ExtUnit reads
L1: Hypothesis $\gg$ $\mathrm{ZfcExt}'(a)$;
L2: z-ExtPair $\trianglerighteq$ L1 $\trianglerighteq$ L1 $\gg$ $\mathrm{ZfcExt}'(\lambda y.\{a\,'\,y, a\,'\,y\})$;
L3: Repetition $\triangleright$ L2 $\gg$ $\mathrm{ZfcExt}'(\lambda y.\{a\,'\,y\})$]*

[The Map proof of z-ExtSuc reads
L1: Hypothesis $\gg$ $\mathrm{ZfcExt}'(a)$;
L2: z-ExtUnit $\trianglerighteq$ L1 $\gg$ $\mathrm{ZfcExt}'(\lambda y.\{a\,'\,y\})$;
L3: z-ExtU $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$ $\mathrm{ZfcExt}'(\lambda y.a\,'\,y \cup \{a\,'\,y\})$;
L4: Repetition $\triangleright$ L3 $\gg$ $\mathrm{ZfcExt}'(\lambda y.(a\,'\,y)^{+})$]*

[The Map proof of z-ExtInfty reads
L1: z-ExtVarX $\trianglerighteq$ z-TypeInfty $\gg$ $\mathrm{ZfcExt}'(\lambda y.\omega)$]*

[The Map proof of z-i reads
L01: TypeT $\gg$ $\ell\,'\,\mathsf{T}$;
L02: z-TypeInfty $\gg$ $\ell\,'\,\omega$;
L03: Reduction $\gg$ $\neg\omega$;
L04: Reduction $\gg$ $\emptyset = \omega\,'\,\mathsf{T}$;
L05: z-IntroIn$\trianglerighteq$L1$\trianglerighteq$L2$\trianglerighteq$L1$\trianglerighteq$L3$\trianglerighteq$L4$\gg$ $\emptyset \in \omega$;
L06: Block $\gg$ Begin ;
L07: Hypothesis $\gg$ $\ell\,'\,x$;
L08: z-TypeIn $\trianglerighteq$ L7 $\trianglerighteq$ L2 $\gg$ $!x \in \omega$;
L09: z-TypeSuc $\trianglerighteq$ L7 $\gg$ $\ell\,'(x^{+})$;
L10: z-TypeIn $\trianglerighteq$ L9 $\trianglerighteq$ L2 $\gg$ $!x^{+} \in \omega$;
L11: Block $\gg$ Begin ;
L12: Hypothesis $\gg$ $x \in \omega$;
L13: Define $\gg$ $y \equiv \varepsilon y\colon x = \omega\,'\,y$;
L14: z-ElimInEll $\trianglerighteq$ L12 $\gg$ $\ell\,'\,y$;
L15: z-ElimIn $\trianglerighteq$ L12 $\gg$ $x = \omega\,'\,y$;
L16: TypeKa $\trianglerighteq$ L14 $\gg$ $\ell\,'(\mathsf{K}\,'\,y)$;
L17: TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L14 $\gg$ $\ell\,'(\omega\,'\,y)$;
L18: z-ExtVarY $\gg$ $\mathrm{ZfcExt}'(\lambda z.z)$;
L19: z-ExtSuc $\trianglerighteq$ L18 $\gg$ $\mathrm{ZfcExt}'(\lambda z.z^{+})$;
L20: z-ElimZfcExt'$\trianglerighteq$L7$\trianglerighteq$L17$\trianglerighteq$L19$\trianglerighteq$L15$\gg$ $x^{+} = (\omega\,'\,y)^{+}$;
L21: Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L20 $\gg$ $x^{+} = \omega\,'(\mathsf{K}\,'\,y)$;
L22: z-IntroIn$\trianglerighteq$L9$\trianglerighteq$L2$\trianglerighteq$L16$\trianglerighteq$L3$\trianglerighteq$L21$\gg$ $x^{+} \in \omega$;
L23: Block $\gg$ End ;
L24: CDeduction $\triangleright$ L8 $\triangleright$ L10 $\triangleright$ L22 $\gg$ $x \in \omega \Rightarrow x^{+} \in \omega$;
L25: Block $\gg$ End ;
L26: Gen $\triangleright$ L24 $\gg$ $\forall x\colon (x \in \omega \Rightarrow x^{+} \in \omega)$;
L27: Logic $\trianglerighteq$ L5 $\trianglerighteq$ L26 $\gg$ $\emptyset \in \omega \mathbin{\dot\wedge} \forall x\colon (x{\in}\omega{\Rightarrow}x^{+}{\in}\omega)$]*

A.65 Lemmas about [$\mathsf{C}x$]

[$\mathsf{C}x \doteq \varepsilon y\colon y \in x$]*

[Map lemma

z-TypeChoice$_{268}$:	$\ell x\colon \ell\,\text{'}\,\mathsf{C}x$	;
z-ExtChoice$_{268}$:	$\mathrm{ZfcExt}'(a) \to \mathrm{ZfcExt}'(\lambda y.\mathsf{C}a\,\text{'}\,y)$	;
z-a$_{268}$:	$\forall x \forall y\colon (x = y \Rightarrow \mathsf{C}x = \mathsf{C}y)$	;
z-c$_{269}$:	$\forall x\colon (x \neq \emptyset \Rightarrow \mathsf{C}x \in x)$	]*

[The Map proof of z-TypeChoice reads

L1:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad\ell\,\text{'}\,y$	;
L4:	z-TypeIn $\trianglerighteq$ L3 $\trianglerighteq$ L1 $\gg$	$\quad !y \in x$	;
L5:	Block $\gg$	End	;
L6:	TypeChoice $\trianglerighteq$ (Gen $\triangleright$ L4) $\gg$	$\ell\,\text{'}\,\varepsilon y\colon y \in x$	;
L7:	Repetition $\triangleright$ L6 $\gg$	$\ell\,\text{'}\,\mathsf{C}x$	]*

[The Map proof of z-ExtChoice reads

L1:	Hypothesis $\gg$	$\mathrm{ZfcExt}'(a)$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad\ell\,\text{'}\,z$	;
L4:	z-ExtVarX $\trianglerighteq$ L3 $\gg$	$\quad\mathrm{ZfcExt}'(\lambda y.z)$	;
L5:	z-ExtIn $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\gg$	$\quad\mathrm{ZfcExt}'(\lambda y.z \in a\,\text{'}\,y)$	;
L6:	Block $\gg$	End	;
L7:	z-ExtChoice $\triangleright$ L5 $\gg$	$\mathrm{ZfcExt}'(\lambda y.\varepsilon z\colon z \in a\,\text{'}\,y)$	;
L8:	Rename $\triangleright$ L7 $\gg$	$\mathrm{ZfcExt}'(\lambda y.\mathsf{C}a\,\text{'}\,y)$	]*

[The Map proof of z-a reads

L01:	Block $\gg$	Begin	;
L02:	Hypothesis $\gg$	$\quad\ell\,\text{'}\,x$	;
L03:	Hypothesis $\gg$	$\quad\ell\,\text{'}\,y$	;
L04:	z-TypeEqual $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$\quad !x = y$	;
L05:	z-TypeChoice $\trianglerighteq$ L2 $\gg$	$\quad\ell\,\text{'}\,\mathsf{C}x$	;
L06:	z-TypeChoice $\trianglerighteq$ L3 $\gg$	$\quad\ell\,\text{'}\,\mathsf{C}y$	;
L07:	z-TypeEqual $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$\quad !\mathsf{C}x = \mathsf{C}y$	;
L08:	Block $\gg$	$\quad$Begin	;
L09:	Hypothesis $\gg$	$\qquad x = y$	;
L10:	z-ExtVarY $\gg$	$\qquad\mathrm{ZfcExt}'(\lambda z.z)$	;
L11:	z-ExtChoice $\trianglerighteq$ L10 $\gg$	$\qquad\mathrm{ZfcExt}'(\lambda z.\mathsf{C}z)$	;
L12:	z-ElimZfcExt'$\trianglerighteq$L2$\trianglerighteq$L3$\trianglerighteq$L11$\trianglerighteq$L9$\gg$	$\qquad\mathsf{C}x = \mathsf{C}y$	;
L13:	Block $\gg$	$\quad$End	;
L14:	CDeduction $\triangleright$ L4 $\triangleright$ L7 $\triangleright$ L12 $\gg$	$\quad x = y \Rightarrow \mathsf{C}x = \mathsf{C}y$	;
L15:	Block $\gg$	End	;
L16:	Gen2 $\triangleright$ L14 $\gg$	$\forall x \forall y\colon (x = y \Rightarrow \mathsf{C}x = \mathsf{C}y)$	]*

[The Map proof of z-c reads

L01:	Block $\gg$	Begin	;
L02:	Hypothesis $\gg$	$\ell\,{'}\,x$	;
L03:	z-TypeEmpty $\gg$	$\ell\,{'}\,\emptyset$	;
L04:	z-TypeEqual $\trianglerighteq$ L2 $\trianglerighteq$ L3 $\gg$	$!x = \emptyset$	;
L05:	Logic $\trianglerighteq$ L4 $\gg$	$!x \neq \emptyset$	;
L06:	z-TypeChoice $\trianglerighteq$ L2 $\gg$	$\ell\,{'}\,\mathsf{C}x$	;
L07:	z-TypeIn $\trianglerighteq$ L6 $\trianglerighteq$ L2 $\gg$	$!\mathsf{C}x \in x$	;
L08:	Block $\gg$	Begin	;
L09:	Hypothesis $\gg$	$x \neq \emptyset$	;
L10:	Logic $\trianglerighteq$ L9 $\gg$	$\neg x$	;
L11:	z-Member' $\trianglerighteq$ L2 $\trianglerighteq$ L10 $\gg$	$\exists y\colon y \in x$	;
L12:	ElimExists $\trianglerighteq$ L11 $\gg$	$(\varepsilon y\colon y \in x) \in x$	;
L13:	Rename $\triangleright$ L12 $\gg$	$\mathsf{C}x \in x$	;
L14:	Block $\gg$	End	;
L15:	CDeduction $\triangleright$ L5 $\triangleright$ L7 $\triangleright$ L13 $\gg$	$x \neq \emptyset \Rightarrow \mathsf{C}x \in x$	;
L16:	Block $\gg$	End	;
L17:	Gen $\triangleright$ L15 $\gg$	$\forall x\colon (x \neq \emptyset \Rightarrow \mathsf{C}x \in x)$	]*

A.66 The axiom of replacement

[Map lemma

z-TypeR'$_{269}$:	$\langle x,u{\in}\mathcal{V};\mathcal{A}{\in}F\rangle \ell u\colon \mathrm{ZfcExt}(\lambda x.\mathcal{A}) \vdash$	
	$\ell x,u\colon !\mathcal{A}$	;
z-TypeR$_{269}$:	$\langle x,u{\in}\mathcal{V};\mathcal{A}{\in}F\rangle \ell u\colon \mathrm{ZfcExt}(\lambda x.\mathcal{A}) \vdash$	
	$\ell z\colon \ell\,{'}((\lambda x.\varepsilon u\colon \mathcal{A}) \circ z)$	;
z-TypeR1$_{270}$:	$\langle x,u{\in}\mathcal{V};\mathcal{A},z{\in}F\rangle \ell u\colon \mathrm{ZfcExt}(\lambda x.\mathcal{A}) \vdash \ell\,{'}\,z \vdash$	
	$\ell x\colon !(x \in z \mathbin{\dot\wedge} \exists u\colon \mathcal{A})$	;
z-TypeR2$_{270}$:	$\langle x,u{\in}\mathcal{V};\mathcal{A},y{\in}F\rangle \ell u\colon \mathrm{ZfcExt}(\lambda x.\mathcal{A}) \vdash \ell\,{'}\,y \vdash$	
	$\ell x\colon !\exists u\colon u \in y \mathbin{\dot\wedge} \mathcal{A}$	;
z-r'$_{270}$:	$\langle x,y,z,u{\in}\mathcal{V};\mathcal{A}{\in}F\rangle \mathrm{notfree}(y,z;\mathcal{A}) \Vdash \ell u\colon \mathrm{ZfcExt}(\lambda x.\mathcal{A}) \vdash$	
	$\ell x,z\colon x \in z \to \exists u\colon \mathcal{A} \to \exists u\colon u \in (\lambda x.\varepsilon u\colon \mathcal{A}) \circ z \mathbin{\dot\wedge} \mathcal{A}$	;
z-r$_{271}$:	$\langle x,y,z,u{\in}\mathcal{V};\mathcal{A}{\in}F\rangle \mathrm{notfree}(y,z;\mathcal{A}) \Vdash \ell u\colon \mathrm{ZfcExt}(\lambda x.\mathcal{A}) \vdash$	
	$\forall z \exists y \forall x\colon (x \in z \mathbin{\dot\wedge} \exists u\colon \mathcal{A} \Rightarrow \exists u\colon u \in y \mathbin{\dot\wedge} \mathcal{A})$	]*

[The Map proof of z-TypeR' reads

L01:	Premise $\gg$	$\ell u\colon \mathrm{ZfcExt}(\lambda x.\mathcal{A})$	;
L02:	Hypothesis $\gg$	$\ell\,{'}\,x$	;
L03:	Hypothesis $\gg$	$\ell\,{'}\,u$	;
L04:	L1 $\trianglerighteq$ L3 $\gg$	$\mathrm{ZfcExt}(\lambda x.\mathcal{A})$	;
L05:	z-SubtypeExtB $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\gg$	$!\mathcal{A}$	]*

[The Map proof of z-TypeR reads

```
L01:  Premise $\gg$                                          $\ell u{:}\,\mathrm{ZfcExt}(\lambda x.\mathcal{A})$      ;
L02:  Hypothesis $\gg$                                       $\ell\,'\,z$                                          ;
L03:  Block $\gg$                                            Begin                                                ;
L04:  Hypothesis $\gg$                                         $\ell\,'\,x$                                        ;
L05:  Block $\gg$                                              Begin                                              ;
L06:  Hypothesis $\gg$                                           $\ell\,'\,u$                                      ;
L07:  z-TypeR' $\triangleright$ L1 $\trianglerighteq$ L4 $\trianglerighteq$ L6 $\gg$          $!\mathcal{A}$                                  ;
L08:  Block $\gg$                                              End                                                ;
L09:  TypeChoice $\trianglerighteq$ (Gen $\triangleright$ L7) $\gg$          $\ell\,'\,\varepsilon u{:}\,\mathcal{A}$                 ;
L10:  Trans $\triangleright$ $\bar{\mathcal{R}}$ $\triangleright$ L9 $\gg$                  $\ell\,'((\lambda x.\varepsilon u{:}\,\mathcal{A})\,'\,x)$   ;
L11:  Block $\gg$                                            End                                                  ;
L12:  TypeAll $\trianglerighteq$ (Gen $\triangleright$ L10) $\gg$            $\ell_1\lambda x.\varepsilon u{:}\,\mathcal{A}$          ;
L13:  TypeCompose $\trianglerighteq$ L12 $\trianglerighteq$ L2 $\gg$         $\ell\,'((\lambda x.\varepsilon u{:}\,\mathcal{A})\circ z)$  ]*
```

[The Map proof of z-TypeR1 reads

```
L1:  Premise $\gg$                                  $\ell u{:}\,\mathrm{ZfcExt}(\lambda x.\mathcal{A})$        ;
L2:  Premise $\gg$                                  $\ell\,'\,z$                                            ;
L3:  Hypothesis $\gg$                               $\ell\,'\,x$                                            ;
L4:  z-TypeIn $\trianglerighteq$ L3 $\trianglerighteq$ L2 $\gg$          $!x \in z$                                ;
L5:  z-TypeR' $\triangleright$ L1 $\trianglerighteq$ L3 $\gg$           $\ell u{:}\,!\mathcal{A}$                     ;
L6:  TypeExists $\trianglerighteq$ (Gen $\triangleright$ L5) $\gg$       $!\exists u{:}\,\mathcal{A}$                 ;
L7:  Logic $\trianglerighteq$ L4 $\trianglerighteq$ L6 $\gg$             $!(x \in z \mathbin{\dot{\wedge}} \exists u{:}\,\mathcal{A})$   ]*
```

[The Map proof of z-TypeR2 reads

```
L01:  Premise $\gg$                                       $\ell u{:}\,\mathrm{ZfcExt}(\lambda x.\mathcal{A})$      ;
L02:  Premise $\gg$                                       $\ell\,'\,y$                                          ;
L03:  Hypothesis $\gg$                                    $\ell\,'\,x$                                          ;
L04:  Block $\gg$                                         Begin                                                ;
L05:  Hypothesis $\gg$                                      $\ell\,'\,u$                                        ;
L06:  z-TypeIn $\trianglerighteq$ L5 $\trianglerighteq$ L2 $\gg$                $!u \in y$                             ;
L07:  z-TypeR' $\triangleright$ L1 $\trianglerighteq$ L3 $\trianglerighteq$ L5 $\gg$      $!\mathcal{A}$                        ;
L08:  Logic $\trianglerighteq$ L6 $\trianglerighteq$ L7 $\gg$                   $!(u \in y \mathbin{\dot{\wedge}} \mathcal{A})$       ;
L09:  Block $\gg$                                         End                                                  ;
L10:  TypeExists $\trianglerighteq$ (Gen $\triangleright$ L8) $\gg$          $!\exists u{:}\, u \in y \mathbin{\dot{\wedge}} \mathcal{A}$   ]*
```

[The Map proof of z-r' reads

L01:	Premise $\gg$	$\ell u\colon \mathrm{ZfcExt}(\lambda x.\mathcal{A})$	;
L02:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L03:	Hypothesis $\gg$	$\ell\,\text{'}\,z$	;
L04:	Hypothesis $\gg$	$x \in z$	;
L05:	Hypothesis $\gg$	$\exists u\colon \mathcal{A}$	;
L06:	z-TypeR $\rhd$ L1 $\unrhd$ L3 $\gg$	$\ell\,\text{'}((\lambda x.\varepsilon u\colon \mathcal{A}) \circ z)$	;
L07:	z-TypeR2 $\rhd$ L1 $\rhd$ L6 $\unrhd$ L2 $\gg$	$!\exists u\colon u \in (\lambda x.\varepsilon u\colon \mathcal{A}) \circ z \mathbin{\dot\wedge} \mathcal{A}$	;
L08:	StrictExists $\unrhd$ L7 $\gg$	$!_1\,\lambda u.u \in (\lambda x.\varepsilon u\colon \mathcal{A})\circ z \mathbin{\dot\wedge} \mathcal{A}$	;
L09:	Define $\gg$	$u \equiv \varepsilon u\colon \mathcal{A}$	;
L10:	ElimExistsEll $\unrhd$ L5 $\gg$	$\ell\,\text{'}\,u$	;
L11:	ElimExists $\unrhd$ L5 $\gg$	$\mathcal{A}$	;
L12:	Define $\gg$	$v \equiv \varepsilon v\colon x = z\,\text{'}\,v$	;
L13:	z-ElimInEll $\unrhd$ L4 $\gg$	$\ell\,\text{'}\,v$	;
L14:	z-ElimIn $\unrhd$ L4 $\gg$	$x = z\,\text{'}\,v$	;
L15:	z-ExtChoice' $\rhd$ L1 $\gg$	$\mathrm{ZfcExt}'(\lambda x.\varepsilon u\colon \mathcal{A})$	;
L16:	TypeApply $\unrhd$ L3 $\unrhd$ L13 $\gg$	$\ell\,\text{'}(z\,\text{'}\,v)$	;
L17:	z-ElimZfcExt$\unrhd$L2$\unrhd$L16$\unrhd$L15$\unrhd$L14$\gg$	$u = (\lambda x.\varepsilon u\colon \mathcal{A})\,\text{'}(z\,\text{'}\,v)$	;
L18:	Trans $\rhd$ $\bar{\mathcal{R}}$ $\rhd$ L17 $\gg$	$u = ((\lambda x.\varepsilon u\colon \mathcal{A}) \circ z)\,\text{'}\,v$	;
L19:	Reduction $\gg$	$\neg(\lambda x.\varepsilon u\colon \mathcal{A}) \circ z$	;
L20:	z-IntroIn$\unrhd$L10$\unrhd$L6$\unrhd$L13$\unrhd$L19$\unrhd$L18$\gg$	$u \in (\lambda x.\varepsilon u\colon \mathcal{A}) \circ z$	;
L21:	Logic $\unrhd$ L11 $\unrhd$ L20 $\gg$	$u \in (\lambda x.\varepsilon u\colon \mathcal{A}) \circ z \mathbin{\dot\wedge} \mathcal{A}$	;
L22:	IntroExists $\unrhd$ L8 $\unrhd$ L10 $\unrhd$ L21 $\gg$	$\exists u\colon u \in (\lambda x.\varepsilon u\colon \mathcal{A}) \circ z \mathbin{\dot\wedge} \mathcal{A}$	]*

[The Map proof of z-r reads

L01:	Premise $\gg$	$\ell u\colon \mathrm{ZfcExt}(\lambda x.\mathcal{A})$	;
L02:	Block $\gg$	Begin	;
L03:	Hypothesis $\gg$	$\ell\,\text{'}\,z$	;
L04:	Block $\gg$	Begin	;
L05:	Hypothesis $\gg$	$\ell\,\text{'}\,y$	;
L06:	Block $\gg$	Begin	;
L07:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L08:	z-TypeR1 $\rhd$ L1 $\rhd$ L3 $\unrhd$ L7 $\gg$	$!(x \in z \mathbin{\dot\wedge} \exists u\colon \mathcal{A})$	;
L09:	z-TypeR2 $\rhd$ L1 $\rhd$ L5 $\unrhd$ L7 $\gg$	$!\exists u\colon u \in y \mathbin{\dot\wedge} \mathcal{A}$	;
L10:	Logic $\unrhd$ L8 $\unrhd$ L9 $\gg$	$!(x \in z \mathbin{\dot\wedge} \exists u\colon \mathcal{A} \Rightarrow \exists u\colon u \in y \mathbin{\dot\wedge} \mathcal{A})$	;
L11:	Block $\gg$	End	;
L12:	TypeAll $\unrhd$ (Gen $\rhd$ L10) $\gg$	$!\forall x\colon (x \in z \mathbin{\dot\wedge} \exists u\colon \mathcal{A} \Rightarrow \exists u\colon u \in y \mathbin{\dot\wedge} \mathcal{A})$	;
L13:	Block $\gg$	End	;
L14:	z-TypeR $\rhd$ L1 $\unrhd$ L3 $\gg$	$\ell\,\text{'}((\lambda x.\varepsilon u\colon \mathcal{A}) \circ z)$	;
L15:	Block $\gg$	Begin	;
L16:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L17:	z-TypeR1 $\rhd$ L1 $\rhd$ L3 $\unrhd$ L16 $\gg$	$!x \in z \mathbin{\dot\wedge} \exists u\colon \mathcal{A}$	;
L18:	z-TypeR2 $\rhd$ L1 $\rhd$ L14 $\unrhd$ L16 $\gg$	$!\exists u\colon u \in (\lambda x.\varepsilon u\colon \mathcal{A}) \circ z \mathbin{\dot\wedge} \mathcal{A}$	;
L19:	Block $\gg$	Begin	;
L20:	Hypothesis $\gg$	$x \in z \mathbin{\dot\wedge} \exists u\colon \mathcal{A}$	;
L21:	Logic $\unrhd$ L20 $\gg$	$x \in z$	;
L22:	Logic $\unrhd$ L20 $\gg$	$\exists u\colon \mathcal{A}$	;
L23:	z-r'$\rhd$L1$\unrhd$L16$\rhd$L3$\unrhd$L21$\unrhd$L22$\gg$	$\exists u\colon u {\in} (\lambda x.\varepsilon u\colon \mathcal{A}) {\circ} z {\dot\wedge} \mathcal{A}$	;
L24:	Block $\gg$	End	;
L25:	CDeduction$\rhd$L17$\rhd$L18$\rhd$L23$\gg$	$x \in z \mathbin{\dot\wedge} \exists u\colon \mathcal{A} \Rightarrow \exists u\colon u \in (\lambda x.\varepsilon u\colon \mathcal{A}) \circ z \mathbin{\dot\wedge} \mathcal{A}$	;
L26:	Block $\gg$	End	;
L27:	Gen $\rhd$ L25 $\gg$	$\forall x\colon (x \in z \mathbin{\dot\wedge} \exists u\colon \mathcal{A} \Rightarrow \exists u\colon u \in (\lambda x.\varepsilon u\colon \mathcal{A}) \circ z \mathbin{\dot\wedge} \mathcal{A})$	;
L28:	IntroExists $\unrhd$ (Gen $\rhd$ L12) $\rhd$ L14 $\rhd$ L27 $\gg$	$\exists y \forall x\colon (x \in z \mathbin{\dot\wedge} \exists u\colon \mathcal{A} \Rightarrow \exists u\colon u \in y \mathbin{\dot\wedge} \mathcal{A})$	;
L29:	Block $\gg$	End	;
L30:	Gen $\rhd$ L28 $\gg$	$\forall z \exists y \forall x\colon (x \in z \mathbin{\dot\wedge} \exists u\colon \mathcal{A} \Rightarrow \exists u\colon u \in y \mathbin{\dot\wedge} \mathcal{A})$	]*

A.67 The axiom of restriction

$$\left[D(x,y) \doteq \exists u\colon u \in x \mathbin{\dot\wedge} u \in y \begin{cases} D(x, y\,\text{'}\,\varepsilon v\colon y\,\text{'}\,v \in x) \\ y \end{cases} \right]^{*}.$$

[Map lemma

z-DT'$_{273}$:	$\ell x\colon \neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in \mathsf{T}$	;
z-DT$_{273}$:	$\ell x\colon D(x,\mathsf{T}) \equiv \mathsf{T}$	;
z-TypeD1$_{273}$:	$\ell x,y\colon !\exists u\colon u \in x \mathbin{\dot\wedge} u \in y$	;
z-TypeD2$_{273}$:	$\ell x,y\colon \forall v\colon !v\colon y\,'\,v \in x$	;
z-TypeD3$_{274}$:	$\ell x,y\colon \ell\,'\,\varepsilon v\colon y\,'\,v \in x$	;
z-DE$_{274}$:	$\ell x,y\colon \exists u\colon u \in x \mathbin{\dot\wedge} u \in y \to \exists v\colon y\,'\,v \in x$	;
z-TypeD$_{275}$:	$\ell x,y\colon \ell\,'\,D(x,y)$	;
z-D1$_{276}$:	$\ell x,y\colon y \in x \mathrel{\dot\Rightarrow} D(x,y) \in x$	;
z-D2$_{277}$:	$\ell x,y\colon \neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in D(x,y)$	;
z-d$_{278}$:	$\forall x\colon (x \neq \emptyset \Rightarrow \exists z\colon z \in x \mathbin{\dot\wedge} \neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in z)$	]*

[The Map proof of z-DT' reads

L1:	Hypothesis $\gg$	$\ell\,'\,x$	;
L2:	Block $\gg$	Begin	;
L3:	Hypothesis $\gg$	$\quad\ell\,'\,u$	;
L4:	z-TypeIn $\trianglerighteq$ L3 $\trianglerighteq$ L1 $\gg$	$\quad !u \in x$	;
L5:	Logic $\trianglerighteq$ L4 $\gg$	$\quad\neg(u \in x \mathbin{\dot\wedge} u \in \mathsf{T})$	;
L6:	Block $\gg$	End	;
L7:	Gen $\triangleright$ L5 $\gg$	$\forall u\colon \neg(u \in x \mathbin{\dot\wedge} u \in \mathsf{T})$	;
L8:	Trans $\triangleright$ AllNot $\triangleright$ L7 $\gg$	$\neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in \mathsf{T}$	]*

[The Map proof of z-DT reads

L1:	Hypothesis $\gg$	$\ell\,'\,x$	;
L2:	z-DT' $\trianglerighteq$ L1 $\gg$	$\neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in \mathsf{T}$	;
L3:	Logic $\trianglerighteq$ L2 $\gg$	$D(x,\mathsf{T}) \equiv \mathsf{T}$	]*

[The Map proof of z-TypeD1 reads

L1:	Hypothesis $\gg$	$\ell\,'\,x$	;
L2:	Hypothesis $\gg$	$\ell\,'\,y$	;
L3:	Block $\gg$	Begin	;
L4:	Hypothesis $\gg$	$\quad\ell\,'\,u$	;
L5:	z-TypeIn $\trianglerighteq$ L4 $\trianglerighteq$ L1 $\gg$	$\quad !u \in x$	;
L6:	z-TypeIn $\trianglerighteq$ L4 $\trianglerighteq$ L2 $\gg$	$\quad !u \in y$	;
L7:	Logic $\trianglerighteq$ L5 $\trianglerighteq$ L6 $\gg$	$\quad !(u \in x \mathbin{\dot\wedge} u \in y)$	;
L8:	Block $\gg$	End	;
L9:	TypeExists $\trianglerighteq$ (Gen $\triangleright$ L7) $\gg$	$!\exists u\colon u \in x \mathbin{\dot\wedge} u \in y$	]*

[The Map proof of z-TypeD2 reads

L1:	Hypothesis $\gg$	$\ell\,'\,x$	;
L2:	Hypothesis $\gg$	$\ell\,'\,y$	;
L3:	Block $\gg$	Begin	;
L4:	Hypothesis $\gg$	$\quad\ell\,'\,v$	;
L5:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L4 $\gg$	$\quad\ell\,'(y\,'\,v)$	;
L6:	z-TypeIn $\trianglerighteq$ L5 $\trianglerighteq$ Lx $\gg$	$\quad !y\,'\,v \in x$	;
L7:	Block $\gg$	End	;
L8:	Gen $\triangleright$ L6 $\gg$	$\forall v\colon !y\,'\,v \in x$	]*

[The Map proof of z-TypeD3 reads

L1:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L2:	Hypothesis $\gg$	$\ell\,\text{'}\,y$	;
L3:	TypeChoice $\trianglerighteq$ (z-TypeD2 $\trianglerighteq$ L1 $\trianglerighteq$ L2) $\gg$	$\ell\,\text{'}\,\varepsilon v\colon y\,\text{'}\,v \in x$	]*

[The Map proof of z-DE reads

L01:	Hypothesis $\gg$	$\ell\,\text{'}\,x$	;
L02:	Hypothesis $\gg$	$\ell\,\text{'}\,y$	;
L03:	Hypothesis $\gg$	$\exists u\colon u \in x \mathrel{\dot{\wedge}} u \in y$	;
L04:	Define $\gg$	$u \equiv \varepsilon u\colon u \in x \mathrel{\dot{\wedge}} u \in y$	;
L05:	ElimExistsEll $\trianglerighteq$ L3 $\gg$	$\ell\,\text{'}\,u$	;
L06:	ElimExists $\trianglerighteq$ L3 $\gg$	$u \in x \mathrel{\dot{\wedge}} u \in y$	;
L07:	Logic $\trianglerighteq$ L6 $\gg$	$u \in x$	;
L08:	Logic $\trianglerighteq$ L6 $\gg$	$u \in y$	;
L09:	Define $\gg$	$v \equiv \varepsilon v\colon u = y\,\text{'}\,v$	;
L10:	z-ElimInEll $\trianglerighteq$ L8 $\gg$	$\ell\,\text{'}\,v$	;
L11:	z-ElimIn $\trianglerighteq$ L8 $\gg$	$u = y\,\text{'}\,v$	;
L12:	TypeApply $\trianglerighteq$ L2 $\trianglerighteq$ L10 $\gg$	$\ell\,\text{'}(y\,\text{'}\,v)$	;
L13:	z-ComEqual $\trianglerighteq$ L5 $\trianglerighteq$ L12 $\trianglerighteq$ L9 $\gg$	$y\,\text{'}\,v = u$	;
L14:	z-ext1 $\trianglerighteq$ L12 $\trianglerighteq$ L5 $\trianglerighteq$ L1 $\trianglerighteq$ L13 $\trianglerighteq$ L7 $\gg$	$y\,\text{'}\,v \in x$	;
L15:	z-TypeD2 $\trianglerighteq$ L1 $\trianglerighteq$ L2 $\gg$	$\forall v\colon !y\,\text{'}\,v \in x$	;
L16:	IntroExists $\trianglerighteq$ L15 $\trianglerighteq$ L10 $\trianglerighteq$ L14 $\gg$	$\exists v\colon y\,\text{'}\,v \in x$	]*

```
[ The Map proof of z-TypeD reads
  L01: Hypothesis ≫                          $\ell \mathrel{'} x$                                                    ;
  L02: TypeT ≫                               $\ell \mathrel{'} \mathsf{T}$                                           ;
  L03: Replace' ▷ z-DT ▷ L2 ≫                $\ell \mathrel{'} D(x, \mathsf{T})$                                     ;
  L04: Block ≫                               Begin                                                                   ;
  L05: Hypothesis ≫                            $\neg y$                                                              ;
  L06: Hypothesis ≫                            $\ell \mathrel{'} y$                                                  ;
  L07: Hypothesis ≫                            $\forall z \colon \ell \mathrel{'} D(x, y \mathrel{'} z)$             ;
  L08: z-TypeD1 ⊵ L1 ⊵ L6 ≫                    $!\exists u \colon u \in x \mathbin{\dot\wedge} u \in y$              ;
  L09: Block ≫                                 Begin                                                                 ;
  L10: Hypothesis ≫                              $\exists u \colon u \in x \mathbin{\dot\wedge} u \in y$             ;
  L11: Logic ⊵ L10 ≫                             $D(x,y) \equiv D(x, y \mathrel{'} \varepsilon v \colon y \mathrel{'} v \in x)$ ;
  L12: z-TypeD3 ⊵ L1 ⊵ L6 ≫                      $\ell \mathrel{'} \varepsilon v \colon y \mathrel{'} v \in x$       ;
  L13: ElimAll ⊵ L7 ⊵ L12 ≫                      $\ell \mathrel{'} D(x, y \mathrel{'} \varepsilon v \colon y \mathrel{'} v \in x)$ ;
  L14: Replace' ▷ L11 ▷ L13 ≫                    $\ell \mathrel{'} D(x,y)$                                           ;
  L15: Block ≫                                 End                                                                   ;
  L16: Block ≫                                 Begin                                                                 ;
  L17: Hypothesis ≫                              $\neg\exists u \colon u \in x \mathbin{\dot\wedge} u \in y$         ;
  L18: Logic ⊵ L17 ≫                             $D(x,y) \equiv y$                                                   ;
  L19: Replace' ▷ L18 ▷ L6 ≫                     $\ell \mathrel{'} D(x,y)$                                           ;
  L20: Block ≫                                 End                                                                   ;
  L21: TND ▷ L8 ▷ L14 ▷ L19 ≫                  $\ell \mathrel{'} D(x,y)$                                             ;
  L22: Block ≫                               End                                                                     ;
  L23: Transfinite' ▷ L3 ▷ L21 ≫             $\forall y \colon \ell \mathrel{'} D(x,y)$                              ;
  L24: ElimAll ⊵ L23 ≫                       $\ell \mathrel{'} D(x,y)$                                               ]*
```

[The Map proof of z-D1 reads
```
L01: Hypothesis ≫                       $\ell\,'\,x$                                                  ;
L02: TypeT ≫                            $\ell\,'\,\mathsf{T}$                                         ;
L03: z-TypeIn ⊵ L2 ⊵ L1 ≫               $!\mathsf{T} \in x$                                           ;
L04: Logic ⊵ L3 ≫                       $\mathsf{T} \in x \dot{\Rightarrow} \mathsf{T} \in x$         ;
L05: z-DT ⊵ L1 ≫                        $D(x,\mathsf{T}) \equiv \mathsf{T}$                           ;
L06: Replace ▷ L5 ▷ L4 ≫                $\mathsf{T} \in x \dot{\Rightarrow} D(x,\mathsf{T}) \in x$    ;
L07: Block ≫                            Begin                                                         ;
L08: Hypothesis ≫                         $\neg y$                                                    ;
L09: Hypothesis ≫                         $\ell\,'\,y$                                                ;
L10: Hypothesis ≫                         $\forall z\colon (y\,'\,z \in x \dot{\Rightarrow} D(x,y\,'\,z) \in x)$ ;
L11: z-TypeIn ⊵ L9 ⊵ L1 ≫                 $!y \in x$                                                  ;
L12: Block ≫                              Begin                                                       ;
L13: Hypothesis ≫                           $y \in x$                                                 ;
L14: z-TypeD1 ⊵ L1 ⊵ L9 ≫                   $!\exists u\colon u \in x \dot{\wedge} u \in y$           ;
L15: Block ≫                                Begin                                                     ;
L16: Hypothesis ≫                             $\exists u\colon u \in x \dot{\wedge} u \in y$          ;
L17: z-DE ⊵ L1 ⊵ L9 ⊵ L16 ≫                   $\exists v\colon y\,'\,v \in x$                         ;
L18: Define ≫                                 $v \equiv \varepsilon v\colon y\,'\,v \in x$            ;
L19: ElimExistsEll ⊵ L17 ≫                    $\ell\,'\,v$                                            ;
L20: ElimExists ⊵ L17 ≫                       $y\,'\,v \in x$                                         ;
L21: ElimAll ⊵ L10 ⊵ L19 ≫                    $y\,'\,v \in x \dot{\Rightarrow} D(x,y\,'\,v) \in x$    ;
L22: Logic ⊵ L20 ⊵ L21 ≫                      $D(x,y\,'\,v) \in x$                                    ;
L23: Logic ⊵ L16 ≫                            $D(x,y) \equiv D(x,y\,'\,v)$                            ;
L24: Replace' ⊵ L23 ⊵ L22 ≫                   $D(x,y) \in x$                                          ;
L25: Block ≫                                End                                                       ;
L26: Block ≫                                Begin                                                     ;
L27: Hypothesis ≫                             $\neg\exists u\colon u \in x \dot{\wedge} u \in y$      ;
L28: Logic ⊵ L27 ≫                            $D(x,y) \equiv y$                                       ;
L29: Replace ▷ L28 ▷ L13 ≫                    $D(x,y) \in x$                                          ;
L30: Block ≫                                End                                                       ;
L31: TND ▷ L14 ▷ L24 ▷ L29 ≫                $D(x,y) \in x$                                            ;
L32: Block ≫                              End                                                         ;
L33: Deduction" ▷ L11 ▷ L31 ≫             $y \in x \dot{\Rightarrow} D(x,y) \in x$                    ;
L34: Block ≫                            End                                                           ;
L35: Transfinite' ▷ L6 ▷ L33 ≫          $\forall y\colon (y \in x \dot{\Rightarrow} D(x,y) \in x)$    ;
L36: ElimAll ⊵ L35 ≫                    $\ell y\colon y \in x \dot{\Rightarrow} D(x,y) \in x$         ]*
```

[The Map proof of z-D2 reads

L01:	Hypothesis $\gg$	$\ell\,'\,x$	;
L02:	z-DT' $\trianglerighteq$ L1 $\gg$	$\neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in \mathsf{T}$	;
L03:	z-DT $\trianglerighteq$ L1 $\gg$	$D(x,\mathsf{T}) \equiv \mathsf{T}$	;
L04:	Replace' $\triangleright$ L3 $\triangleright$ L2 $\gg$	$\neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in D(x,y)$	;
L05:	Block $\gg$	Begin	;
L06:	Hypothesis $\gg$	$\quad \neg y$	;
L07:	Hypothesis $\gg$	$\quad \ell\,'\,y$	;
L08:	Hypothesis $\gg$	$\quad \forall z\colon \neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in D(x, y\,'\,z)$	;
L09:	z-TypeD1 $\trianglerighteq$ L1 $\trianglerighteq$ L7 $\gg$	$\quad !\exists u\colon u \in x \mathbin{\dot\wedge} u \in y$	;
L10:	Block $\gg$	$\quad$ Begin	;
L11:	Hypothesis $\gg$	$\quad\quad \exists u\colon u \in x \mathbin{\dot\wedge} u \in y$	;
L12:	z-DE $\trianglerighteq$ L1 $\trianglerighteq$ L7 $\trianglerighteq$ L11 $\gg$	$\quad\quad \exists v\colon y\,'\,v \in x$	;
L13:	Define $\gg$	$\quad\quad v \equiv \varepsilon v\colon y\,'\,v \in x$	;
L14:	ElimExistsEll $\trianglerighteq$ L12 $\gg$	$\quad\quad \ell\,'\,v$	;
L15:	ElimAll $\trianglerighteq$ L8 $\trianglerighteq$ L14 $\gg$	$\quad\quad \neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in D(x, y\,'\,v)$	;
L16:	Logic $\trianglerighteq$ L11 $\gg$	$\quad\quad D(x,y) \equiv D(x, y\,'\,v)$	;
L17:	Replace' $\triangleright$ L16 $\triangleright$ L15 $\gg$	$\quad\quad \neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in D(x,y)$	;
L18:	Block $\gg$	$\quad$ End	;
L19:	Block $\gg$	$\quad$ Begin	;
L20:	Hypothesis $\gg$	$\quad\quad \neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in y$	;
L21:	Logic $\trianglerighteq$ L20 $\gg$	$\quad\quad D(x,y) \equiv y$	;
L22:	Replace' $\triangleright$ L21 $\triangleright$ L20 $\gg$	$\quad\quad \neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in D(x,y)$	;
L23:	Block $\gg$	$\quad$ End	;
L24:	TND $\triangleright$ L9 $\triangleright$ L17 $\triangleright$ L22 $\gg$	$\quad \neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in D(x,y)$	;
L25:	Block $\gg$	End	;
L26:	Transfinite' $\triangleright$ L2 $\triangleright$ L24 $\gg$	$\forall y\colon \neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in D(x,y)$	;
L27:	ElimAll $\trianglerighteq$ L26 $\gg$	$\ell y\colon \neg\exists u\colon u \in x \mathbin{\dot\wedge} u \in D(x,y)$	]*

[The Map proof of z-d reads

```
L01:  Block ≫                          Begin                                                         ;
L02:  Hypothesis ≫                       $\ell$ ' $x$                                                  ;
L03:  z-TypeEmpty ≫                      $\ell$ ' $\emptyset$                                          ;
L04:  z-TypeEqual ⊵ L2 ⊵ L3 ≫            $!x = \emptyset$                                              ;
L05:  Logic ⊵ L4 ≫                       $!x \neq \emptyset$                                           ;
L06:  Block ≫                            Begin                                                       ;
L07:  Hypothesis ≫                         $\ell$ ' $z$                                                ;
L08:  z-TypeIn ⊵ L7 ⊵ L1 ≫                 $!z \in x$                                                  ;
L09:  z-TypeD1 ⊵ L1 ⊵ L7 ≫                 $!\exists u\colon u \in x \dot\wedge u \in z$               ;
L10:  Logic ⊵ L8 ⊵ L9 ≫                    $!(z \in x \dot\wedge \neg\exists u\colon u{\in}x \dot\wedge u{\in}z)$ ;
L11:  Block ≫                            End                                                         ;
L12:  TypeExists ⊵ (Gen ▷ L10) ≫         $!\exists z\colon z \in x \dot\wedge \neg\exists u\colon u{\in}x \dot\wedge u{\in}z$ ;
L13:  Block ≫                            Begin                                                       ;
L14:  Hypothesis ≫                         $x \neq \emptyset$                                          ;
L15:  Logic ⊵ L14 ≫                        $\neg x$                                                    ;
L16:  Define ≫                             $y \equiv x$ ' $x$                                          ;
L17:  z-Member ⊵ L2 ⊵ L2 ⊵ L15 ≫           $y \in x$                                                   ;
L18:  TypeApply ⊵ L2 ⊵ L2 ≫                $\ell$ ' $y$                                                ;
L19:  z-TypeD ⊵ L2 ⊵ L18 ≫                 $\ell$ ' $D(x,y)$                                           ;
L20:  z-D1 ⊵ L2 ⊵ L18 ⊵ L17 ≫              $D(x,y) \in x$                                              ;
L21:  z-D2 ⊵ L2 ⊵ L18 ≫                    $\neg\exists u\colon u \in x \dot\wedge u \in D(x,y)$       ;
L22:  Logic ⊵ L20 ⊵ L21 ≫                  $D(x,y) \in x \dot\wedge$
                                           $\neg\exists u\colon u \in x \dot\wedge u \in D(x,y)$       ;
L23:  IntroExists ▷
      (Gen ▷ L10) ▷ L19 ▷ L22 ≫            $\exists z\colon z{\in}x\dot\wedge\neg\exists u\colon u{\in}x\dot\wedge u{\in}z$ ;
L24:  Block ≫                            End                                                         ;
L25:  CDeduction▷L5▷L12▷L23≫             $x{\neq}\emptyset{\Rightarrow}\exists z\colon z{\in}x\dot\wedge\neg\exists u\colon u{\in}x\dot\wedge u{\in}z$ ;
L26:  Block ≫                          End                                                           ;
L27:  Gen ▷ L25 ≫                      $\forall x\colon (x \neq \emptyset \Rightarrow$
                                       $\exists z\colon z \in x \dot\wedge \neg\exists u\colon u \in x \dot\wedge u \in z)$  ]*
```

Appendix B

Index

Appendix C

Summary

C.1 Brackets

All formulas are enclosed in brackets. Formulas that are intended for mechanical proof checkers are decorated with an asterisk.

C.2 Priority

Below, $\left[\,\boxed{\mathcal{A} \overset{\smile}{>} \mathcal{B}}\,\right]$ means that $[\,\mathcal{A}\,]$ has greater priority than $[\,\mathcal{B}\,]$. $\left[\,\boxed{\mathcal{A} \overset{\smile}{=} \mathcal{B}}\,\right]$ means that $[\,\mathcal{A}\,]$ has the same priority as $[\,\mathcal{B}\,]$. As an example, $[\,x \vee y \overset{\smile}{>} x \Rightarrow y \overset{\smile}{=} x \Leftrightarrow y\,]$ means that $[\,x \vee y\,]$ has greater priority than $[\,x \Rightarrow y\,]$ which in turn has the same priority as $[\,x \Leftrightarrow y\,]$ (which infers that $[\,x \vee y\,]$ has greater priority than [$x \Leftrightarrow y$]).

$$
\begin{array}{lll}
[& x \mathbin{'} y & \breve{>} \\
 & x \circ y & \breve{>} \\
 & x^{+} \mathrel{\breve{=}} x^{-} & \breve{>} \\
 & x :: y & \breve{>} \\
 & x \downarrow y & \breve{>} \\
 & x \mathbin{@} y & \breve{>} \\
 & x \simeq y \mathrel{\breve{=}} x =_p y \mathrel{\breve{=}} x \sim y \mathrel{\breve{=}} x \sim_p y \mathrel{\breve{=}} x \in_\ell y \mathrel{\breve{=}} & \\
 & x \in_2 y \mathrel{\breve{=}} x \subseteq_2 y & \breve{>} \\
 & \approx x \mathrel{\breve{=}} \neg x \mathrel{\breve{=}} !x \mathrel{\breve{=}} ¡x \mathrel{\breve{=}} ?x & \breve{>} \\
 & x \wedge y \mathrel{\breve{=}} x \mathbin{\tilde{\wedge}} y & \breve{>} \\
 & x \vee y \mathrel{\breve{=}} x \mathbin{\tilde{\vee}} y & \breve{>} \\
 & \forall x \colon y \mathrel{\breve{=}} \exists x \colon y \mathrel{\breve{=}} \varepsilon x \colon y \mathrel{\breve{=}} \mathsf{E} x \colon y & \breve{>} \\
 & x \Rightarrow y \mathrel{\breve{=}} x \Leftarrow y \mathrel{\breve{=}} x \Leftrightarrow y \mathrel{\breve{=}} x \mathrel{\tilde{\Rightarrow}} y \mathrel{\breve{=}} x \mathrel{\tilde{\Leftarrow}} y & \breve{>} \\
 & x \left\{ \begin{array}{l} y \\ z \end{array} \right. & \breve{>} \\
 & \lambda x . y & \breve{>} \\
 & x \equiv y \mathrel{\breve{=}} x \preceq y & \breve{>} \\
 & x \rightarrow y & \breve{>} \\
 & \langle x \in y \rangle z & \breve{>} \\
 & x \vdash y \mathrel{\breve{=}} x \nvdash y \mathrel{\breve{=}} x \Vdash y & \breve{>} \\
 & x \rhd y \mathrel{\breve{=}} x \unrhd y & \breve{>} \\
 & x \gg y & \breve{>} \\
 & x \colon y & \breve{>} \\
 & x ; y & \breve{>} \\
 & x / y & \breve{>} \\
 & x | y & \breve{>} \\
 & x ::= y & \breve{>} \\
 & x \text{ rule } y \mathrel{\breve{=}} x \text{ lemma } y \mathrel{\breve{=}} x \text{ anti lemma } y \mathrel{\breve{=}} & \\
 & x \text{ proof of } y \text{ reads } z &]^{*}
\end{array}
$$

C.3 Associativity

Below, [$\boxed{\mathcal{A} \overset{\sim}{\rightarrow} \mathcal{B}}$] means that the ambiguous statement [$\mathcal{A}$] should be interpretted like [$\mathcal{B}$]. As an example, [$x \mathbin{'} y \mathbin{'} z$] could mean either [$(x \mathbin{'} y) \mathbin{'} z$] or [$x \mathbin{'} (y \mathbin{'} z)$] but [$x \mathbin{'} y \mathbin{'} z \overset{\sim}{\rightarrow} (x \mathbin{'} y) \mathbin{'} z$] states that the former choice is the intended one or, in other words, [$x \mathbin{'} y$] is left associative.

Constructs with the same priority have the same associativity so e.g. [$x \mathbin{\tilde{\wedge}} y$]

inherits left associativity from [$x \wedge y$].

[$x\,'\,y\,'\,z$	$\breve{\rightarrow}$	$(x\,'\,y)\,'\,z$	]*
[$x \circ y \circ z$	$\breve{\rightarrow}$	$(x \circ y) \circ z$	]*
[$x :: y :: z$	$\breve{\rightarrow}$	$x :: (y :: z)$	]*
[$x \downarrow y \downarrow z$	$\breve{\rightarrow}$	$(x \downarrow y) \downarrow z$	]*
[$x \wedge y \wedge z$	$\breve{\rightarrow}$	$(x \wedge y) \wedge z$	]*
[$x \vee y \vee z$	$\breve{\rightarrow}$	$(x \vee y) \vee z$	]*
[$x \Rightarrow y \Rightarrow z$	$\breve{\rightarrow}$	$x \Rightarrow (y \Rightarrow z)$	]*
[$x \rightarrow y \rightarrow z$	$\breve{\rightarrow}$	$x \rightarrow (y \rightarrow z)$	]*
[$x \vdash y \vdash z$	$\breve{\rightarrow}$	$x \vdash (y \vdash z)$	]*
[$x \rhd y \rhd z$	$\breve{\rightarrow}$	$(x \rhd y) \rhd z$	]*
[$x \gg y \gg z$	$\breve{\rightarrow}$	$(x \gg y) \gg z$	]*
[$x\colon y\colon z$	$\breve{\rightarrow}$	$x\colon (y\colon z)$	]*
[$x; y; z$	$\breve{\rightarrow}$	$(x; y); z$	]*

C.4 Fundamental constructs

[N]	the ur-element
[$\lambda x.\mathcal{A}$]	the function that maps [x] to [$\mathcal{A}$]
[$x\,'\,y$]	x applied to y
[$\mathrm{if}(x, y, z)$]	equals [y] is [x] is the ur-element, equals [z] if [x] is a function, and equals [$\perp$] if [x] equals [$\perp$].
[$\mathsf{E}x$]	equals [T] if [$x\,'\,y \equiv \mathsf{T}$] for some [y], and equals [$\perp$] otherwise.
[εx]	equals [$\perp$] if [$x\,'\,y \equiv \perp$] for any classical [y], else equals a classical [y] for which [$x\,'\,y$] if such a one exists, else equals some, unspecified classical [y].

C.5 Definitions

The following list of definitions merely includes those concepts that are needed for stating the axioms and inference rules in Section C.10.

[$\perp$	$\doteq (\lambda x.x\,'\,x)\,'(\lambda x.x\,'\,x)$	]*
[Y	$\doteq \lambda f.(\lambda x.f\,'(x\,'\,x))\,'(\lambda x.f\,'(x\,'\,x))$	]*
[T	$\doteq \mathsf{N}$	]*
[F	$\doteq \lambda x.\mathsf{N}$	]*
[$\approx x$	$\doteq \mathrm{if}(x, \mathsf{T}, \mathsf{F})$	]*
[$x\left\{\begin{array}{l} y \\ z \end{array}\right.$	$\doteq \mathrm{if}(x, y, z)$	]*
[$x \wedge y$	$\doteq x\left\{\begin{array}{l} \mathrm{if}(y, \mathsf{T}, \mathsf{F}) \\ \mathrm{if}(y, \mathsf{F}, \mathsf{F}) \end{array}\right.$	]*
[$x \vee y$	$\doteq x\left\{\begin{array}{l} \mathrm{if}(y, \mathsf{T}, \mathsf{T}) \\ \mathrm{if}(y, \mathsf{T}, \mathsf{F}) \end{array}\right.$	]*

$$\left[\; x \Rightarrow y \qquad \doteq x \begin{cases} \mathrm{if}(y, \mathsf{T}, \mathsf{F}) \\ \mathrm{if}(y, \mathsf{T}, \mathsf{T}) \end{cases} \right]^*$$

$$\left[\; x \Leftarrow y \qquad \doteq x \begin{cases} \mathrm{if}(y, \mathsf{T}, \mathsf{T}) \\ \mathrm{if}(y, \mathsf{F}, \mathsf{T}) \end{cases} \right]^*$$

$$\left[\; x \Leftrightarrow y \qquad \doteq x \begin{cases} \mathrm{if}(y, \mathsf{T}, \mathsf{F}) \\ \mathrm{if}(y, \mathsf{F}, \mathsf{T}) \end{cases} \right]^*$$

$$\begin{array}{lll}
[\; x \mathrel{\tilde{\wedge}} y & \doteq \mathrm{if}(x, y, \mathsf{F}) &]^* \\
[\; x \mathrel{\tilde{\vee}} y & \doteq \mathrm{if}(x, \mathsf{T}, y) &]^* \\
[\; x \mathrel{\tilde{\Rightarrow}} y & \doteq \mathrm{if}(x, y, \mathsf{T}) &]^* \\
[\; x \mathrel{\tilde{\Leftarrow}} y & \doteq y \mathrel{\tilde{\Rightarrow}} x &]^* \\
[\; \neg x & \doteq \mathrm{if}(x, \mathsf{F}, \mathsf{T}) &]^* \\
[\; !x & \doteq \mathrm{if}(x, \mathsf{T}, \mathsf{T}) &]^* \\
[\; !_1 x & \doteq \forall y \colon !\,{}'(x\,{}'\,y) &]^*
\end{array}$$

$$\left[\; x \downarrow y \qquad \doteq x \begin{cases} \mathrm{if}(y, \mathsf{N}, \bot) \\ \mathrm{if}(y, \bot, \lambda z.x\,{}'\,z \downarrow y\,{}'\,z) \end{cases} \right]^*$$

$$\begin{array}{lll}
[\; x \preceq y & \doteq x \equiv x \downarrow y &]^* \\
[\; \mathsf{E}x \colon y & \doteq \mathsf{E}\lambda x.y &]^* \\
[\; x \circ y & \doteq \lambda z.x\,{}'(y\,{}'\,z) &]^* \\
[\; ? & \doteq \lambda x.\mathrm{if}(x, \mathsf{T}, \bot) &]^* \\
[\; \varepsilon x \colon y & \doteq \varepsilon \lambda x.y &]^* \\
[\; \exists x \colon y & \doteq \approx((\lambda x.y)\,{}'\,\varepsilon x \colon y) &]^* \\
[\; \forall x \colon y & \doteq \neg \exists x \colon \neg y &]^*
\end{array}$$

$$\left[\; x =_p y \qquad \doteq x \begin{cases} \mathrm{if}(y, \mathsf{T}, \mathsf{F}) \\ \mathrm{if}(y, \mathsf{F}, \forall u \colon \forall v \colon x\,{}'(p\,{}'\,u\,{}'\,v) =_p y\,{}'(p\,{}'\,u\,{}'\,v)) \end{cases} \right]^*$$

$$\begin{array}{lll}
[\; \mathsf{K} & \doteq \lambda x.\lambda y.x &]^* \\
[\; x \in_l p & \doteq \forall u \forall v \colon (u =_p v \Rightarrow x\,{}'\,u =_{\mathsf{K}} x\,{}'\,v) &]^* \\
[\; \bar{\ell} & \doteq \lambda f.\lambda x.\mathrm{if}(x, \mathsf{T}, (\forall y \colon f\,{}'(x\,{}'\,y)) \wedge \mathsf{E}p \colon f\,{}'\,p \wedge x \in_l p) &]^* \\
[\; \ell & \doteq \mathsf{Y}\,{}'\,\bar{\ell} &]^*
\end{array}$$

C.6 Shorthand conventions

Whenever a term $[\; \mathcal{A} \;]$ occurs in a position where a formula is expected, $[\; \mathcal{A} \;]$ is shorthand for $[\; \mathcal{A} \equiv \mathsf{T} \;]$.

$[\; \mathcal{A} \to (\mathcal{B} \equiv \mathcal{C}) \;]$ is shorthand for $[\; \mathcal{A} \mathrel{\tilde{\wedge}} \mathcal{B} \equiv \mathcal{A} \mathrel{\tilde{\wedge}} \mathcal{C} \;]$.

$[\; \forall x, y \colon z \;]$ is shorthand for $[\; \forall x \colon \forall y \colon z \;]$. Similar conventions are in effect for $[\; \exists \;]$, $[\; ! \;]$, $[\; !_1 \;]$, $[\; !_2 \;]$, $[\; \ell \;]$, $[\; \ell_1 \;]$, and $[\; \ell_2 \;]$.

$[\; \langle x; y \rangle z \;]$ is shorthand for $[\; \langle x \rangle \langle y \rangle z \;]$. $[\; \langle x, y {\in} z \rangle v \;]$ is shorthand for $[\; \langle x {\in} z \rangle \langle y {\in} z \rangle v \;]$.

C.7 Operations on Gödel-numbers

The following constructs can be used in side conditions.

[$\langle \mathcal{A} \mid x{:=}\mathcal{B} \rangle$] equals [$\mathcal{A}$] where all free occurrences of [x] are replaced by [$\mathcal{B}$]. Bound variables are renamed to avoid variable clashes. Renaming is done in some deterministic but unspecified way.

[$\mathcal{A} \simeq \mathcal{B}$] is true if [$\mathcal{A}$] and [$\mathcal{B}$] are identical except for renaming of bound variables.

[$\mathrm{notfree}(x_1, \cdots, x_m; a_1, \cdots, a_n)$] is true if [$x_1, \ldots, x_n$] are distinct variables none of which are free in any of [$a_1, \ldots, a_n$].

C.8 Syntax of MT

variables $\mathcal{V}$:	[$\mathcal{V}$	$::=$	$x \mid y \mid z \mid \cdots$	]
terms $\mathcal{T}$:	[$\mathcal{T}$	$::=$	$\mathcal{V} \mid \mathsf{N} \mid \lambda\mathcal{V}.\mathcal{T} \mid \mathcal{T}\,\text{'}\,\mathcal{T} \mid \mathrm{if}(\mathcal{T},\mathcal{T},\mathcal{T}) \mid \mathsf{E}\mathcal{T} \mid \varepsilon\mathcal{T}$	]
formulas $\mathcal{F}$:	[$\mathcal{F}$	$::=$	$\mathcal{T} \equiv \mathcal{T}$	]

C.9 Constructs for formulating rules

[$x \vdash y$] states that [y] is provable if [x] is provable.
[$x \Vdash y$] states that [y] is provable if [x] computes to [T].
[$\langle x{\in}\mathcal{V} \rangle y$] states that [y] holds for all variables [x].
[$\langle x{\in}\mathcal{T} \rangle y$] states that [y] holds for all terms [x].

C.10 Axioms and inference rules

[Map rule			
IfNil:	$\langle \mathcal{B},\mathcal{C}\in\mathcal{T}\rangle$	$\mathrm{if}(\mathsf{N},\mathcal{B},\mathcal{C})\equiv\mathcal{B}$	;
IfLambda:	$\langle x\in\mathcal{V};\mathcal{A},\mathcal{B},\mathcal{C}\in\mathcal{T}\rangle$	$\mathrm{if}(\lambda x.\mathcal{A},\mathcal{B},\mathcal{C})\equiv\mathcal{C}$	;
IfBottom:	$\langle \mathcal{B},\mathcal{C}\in\mathcal{T}\rangle$	$\mathrm{if}(\bot,\mathcal{B},\mathcal{C})\equiv\bot$	;
Trans:	$\langle \mathcal{A},\mathcal{B},\mathcal{C}\in\mathcal{T}\rangle$	$\mathcal{A}\equiv\mathcal{B}\vdash\mathcal{A}\equiv\mathcal{C}\vdash\mathcal{B}\equiv\mathcal{C}$	;
SubLambda:	$\langle x\in\mathcal{V};\mathcal{A},\mathcal{B}\in\mathcal{T}\rangle$	$\mathcal{A}\equiv\mathcal{B}\vdash\lambda x.\mathcal{A}\equiv\lambda x.\mathcal{B}$	;
SubApply:	$\langle \mathcal{A},\mathcal{B},\mathcal{C}\in\mathcal{T}\rangle$	$\mathcal{A}\equiv\mathcal{B}\vdash\mathcal{C}\,'\,\mathcal{A}\equiv\mathcal{C}\,'\,\mathcal{B}$	;
ApplyNil:	$\langle \mathcal{B}\in\mathcal{T}\rangle$	$\mathsf{N}\,'\,\mathcal{B}\equiv\mathsf{N}$	;
ApplyLambda:	$\langle x\in\mathcal{V};\mathcal{A},\mathcal{B},\mathcal{C}\in\mathcal{T}\rangle$	$\mathcal{C}\simeq\langle\mathcal{A}\mid x{:=}\mathcal{B}\rangle\Vdash$ $(\lambda x.\mathcal{A})\,'\,\mathcal{B}\equiv\mathcal{C}$	;
ApplyBottom:	$\langle \mathcal{B}\in\mathcal{T}\rangle$	$\bot\,'\,\mathcal{B}\equiv\bot$	;
QND:	$\langle \mathcal{A},\mathcal{B},\mathcal{C}\in\mathcal{T}\rangle$	$\mathcal{A}\,'\,\mathsf{N}\equiv\mathcal{B}\,'\,\mathsf{N}\vdash$ $\mathcal{A}\,'\,\Lambda\mathcal{C}\equiv\mathcal{B}\,'\,\Lambda\mathcal{C}\vdash$ $\mathcal{A}\,'\,\bot\equiv\mathcal{B}\,'\,\bot\vdash$ $\mathcal{A}\,'\,\mathcal{C}\equiv\mathcal{B}\,'\,\mathcal{C}$	;
Extensionality:	$\langle u,v\in\mathcal{V};\mathcal{A},\mathcal{B},\mathcal{C}\in\mathcal{T}\rangle$	$\mathrm{notfree}(u,v;\mathcal{A},\mathcal{B},\mathcal{C})\Vdash$ $\approx(\mathcal{A}\,'\,u)\equiv\approx(\mathcal{B}\,'\,u)\vdash$ $\mathcal{A}\,'\,u\,'\,v\equiv\mathcal{A}\,'(\mathcal{C}\,'\,u\,'\,v)\vdash$ $\mathcal{B}\,'\,u\,'\,v\equiv\mathcal{B}\,'(\mathcal{C}\,'\,u\,'\,v)\vdash$ $\mathcal{A}\,'\,u\equiv\mathcal{B}\,'\,u$	;
Monotonicity:	$\langle \mathcal{A},\mathcal{B},\mathcal{C}\in\mathcal{T}\rangle$	$\mathcal{A}\preceq\mathcal{B}\vdash\mathcal{C}\,'\,\mathcal{A}\preceq\mathcal{C}\,'\,\mathcal{B}$	;
Minimality:	$\langle a,b\in\mathcal{T}\rangle$	$a\,'\,b\preceq b\vdash\mathsf{Y}\,'\,a\preceq b$	;
Existence:		$\mathsf{ET}\equiv\mathsf{T}$	;
NonExistence:		$\mathsf{E}\bot\equiv\bot$	;
ImpliedExistence:		$\mathsf{E}(x\circ y)\to\mathsf{E}x$	;
TruthExistence:		$\mathsf{E}x\equiv\mathsf{E}(?\circ x)$	;
ElimAll:		$\forall x\colon a\,'\,x\to\ell\,'\,b\to a\,'\,b$	;
AckermannChoice:		$\varepsilon x\colon a\,'\,x\equiv\varepsilon x\colon\ell\,'\,x\wedge a\,'\,x$	;
StrictTypeChoice:		$\ell\,'\,\varepsilon x\colon a\,'\,x\equiv\,!_1 a$	;
StrictTypeAll:		$!\forall x\colon a\,'\,x\equiv\,!_1 a$	]*

Appendix D

Bibliography

[1] Alfred V. Aho and Jeffrey D. Ullman. *Principles of Compiler Design.* Addison-Wesley, 1979.

[2] H.P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*, volume 103 of *Studies in Logic and The Foundation of Mathematics.* North-Holland, 1984.

[3] C. Berline and K. Grue. A κ-denotational semantics for Map Theory in ZFC+SI. *Theoretical Computer Science*, 179(1–2):137–202, June 1997.

[4] U. Felgner. Choice functions on sets and classes. In *Sets and Classes: On the works by Paul Bernays*, pages 217–255. North-Holland, 1976.

[5] K. Grue. Map theory. *Theoretical Computer Science*, 102(1):1–133, July 1992.

[6] K. Grue. *Mathematics and Computation (lecture notes)*, volume 1–3. DIKU, University of Copenhagen, Denmark, 5. edition, 2000.

[7] D. Hilbert and P. Bernays. *Grundlagen der Mathematic*, volume 2. Springer-Verlag, 1939.

[8] K. Kunen. *Set Theory, An Introduction to Independence Proofs*, volume 102 of *Studies in Logic and The Foundation of Mathematics.* North-Holland, 1980.

[9] E. Mendelson. *Introduction to Mathematical Logic.* Wadsworth and Brooks, 3. edition, 1987.