On the proof theory of Coquand's calculus of constructions\(^1\)

Jonathan P. Seldin

Department of Mathematics, Concordia University, Montréal, Qué., Canada

Received 13 January 1994; revised 17 November 1995

Communicated by A. Nerode

Abstract

The calculus of constructions is formulated as a natural deduction system in which deductions follow the constructions of the terms to which types are assigned. Strong normalization is proved for deductions. This strong normalization result implies the consistency of the underlying system, but it is still possible to make contradictory assumptions. A number of assumption sets useful in implementations are proved consistent, including certain sets of assumptions whose types are negations, negations of certain equations, arithmetic, classical logic, classical arithmetic, and the existence of power sets. All results are given with complete proofs.

The calculus of constructions of Thierry Coquand [5, 7–11] is a system of typed \(\lambda\)-calculus in which the second-order polymorphic typed \(\lambda\)-calculus can be interpreted. Furthermore, using the formulas-as-types notion [19], it is possible to interpret constructive higher-order logic in it. It differs from the type theory of Martin-Löf [27–29] in two important ways: it is impredicative, and it has only a small number of primitive axioms and rules. Coquand proved [5] the normalization theorem and later [7, 11] the strong normalization theorem for it. He also showed [6] that any of several natural looking extensions of the calculus is inconsistent. (More recently, Luo [22, 23] has shown that the calculus of constructions can be consistently extended in an interesting way by adding an infinite hierarchy of type universes, but this extension will not concern us here.)

\(^{1}\) E-mail: seldin@alcor.concordia.ca.

\(^{1}\) This research was sponsored in part by the US Air Force Systems Command, Rome Air Development Center, Griffiss AFB, New York 13441-5700 under Contract No. F16602-85-C-0098 (while the author was working for Odyssey Research Associates), in part by grant EQ41840 of the program Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (F.C.A.R.) of the Québec Ministry of Education, and in part by grant OGP0023391 from the Natural Sciences and Engineering Research Council of Canada.
In [37, Chs. 4 and 5], I gave a proof of the strong normalization theorem for deductions (rather than for terms) of a version of the calculus of constructions. The version is a natural deduction system in the style of Gentzen in which the deductions follow the formation of the terms to which types are being assigned. The normalization theorem for terms follows from that for deductions. I also proved consistent the result of adding certain assumptions (in the form of premises for mathematical theories). More recent versions of these results are stated without proof in [38], while some new results along these lines are given in [39]. Here, these results will be given with full proofs.

We shall begin in Section 1 with the formulation of the system. In Section 2 we shall take up its basic metatheory; because of the formulation, some results which are trivial for the formulation of Coquand require nontrivial proofs. In Section 3 we shall give the proof of the strong normalization theorem for deductions; its basic consequences will be given in Section 4. In Section 5 we shall compare the formulation of Section 1 with other formulations closer to those of Coquand and Huet. In Section 6 we shall consider representing logic with equality. In Section 7 we shall discuss the addition of assumptions for mathematics and logic and define some weak sets of assumptions that are provably consistent. In Section 8 we look at negations of equations and conditions for them to be consistent. In Section 9 it is shown that assumptions sufficient for arithmetic are consistent, and these results are applied to other inductively generated free algebras, in particular to lists. In Section 10, it is shown that an assumption sufficient for classical logic is consistent by itself and together with the assumptions needed for arithmetic. Finally, in Section 11, we look at sets; Huet [20, Ch. 12; 21] has given an interpretation of the language of sets which is natural but in which the power set of a set is never a set, and here assumptions which allow many power sets to be sets are shown to be consistent.

1. Natural deduction formulation of the calculus

We begin with terms which are built up from variables \( w, x, y, z, w_1, \ldots \) and the constants Prop and Type in the following way:

1. Variables and constants are terms.
2. If \( X \) and \( Y \) are terms, then so are \( (XY), (\lambda x : X . Y), \) and \( (\forall x : X)Y. \)

In \( (\lambda x : X . Y) \) and \( (\forall x : X)Y \), the variable \( x \) is bound to the left of the colon and in \( Y \), but not in \( X \) (where in practice it does not occur). \( [X/x]Y \) will be the result of substituting \( X \) for the free occurrences of \( x \) in \( Y \), bound variables being changed as necessary in order to avoid capture. We shall follow the usual practice of identifying terms that differ only by alphabetical changes of bound variables.

Remark 1. If we represent the type \( (\forall x : X)Y \) by \( GX(\lambda x . Y) \), then we have a system of generalized-type assignment in the sense of Curry. See [18, Sections 16C–E].
Remark 2. In a typed system, it is the intention that \([X/x]Y\) be meaningful only if \(X\) and \(x\) have the same type. However, when we define type assignment by a set of rules which depends on substitution being previously defined, as we are doing here, we cannot make this condition part of the definition of substitution.

Remark 3. When a system like this is implemented, there are complications that are not considered here. Coquand and Huet use in their publications a system for representing these terms due to de Bruijn [15] in which terms which differ only by alphabetic changes of bound variables are really identical. For another such system for representing terms, see [34].

From now on, capital letters will denote terms. \(FV(X)\) will be the set of free variables in \(X\).

A term of the form \((\lambda x : A.M)N\) is called a redex and \([N/x]M\) is called its contractum. With these definitions, reduction and conversion are defined in the way usual for \(\lambda\)-calculus, and we shall denote them by \(\triangleright\) and \(\Rightarrow^\ast\) (as in [18]). Note that although this corresponds to \(\beta\)-reduction rather than \(\eta\)-reduction, it is not identical to \(\beta\)-reduction (because the syntax involves giving explicitly the types of variables bound by \(\lambda\)). We will not be considering here a reduction corresponding to \(\eta\)-reduction.

Remark 4. Suppose we do consider a reduction corresponding to \(\eta\)-reduction, and consider the term \(\lambda x : y. (\lambda x : z.x)x\). It would be natural to think of this as an \(\eta\)-redex with contractum \(\lambda x : z.x\). But the term has a subterm \((\lambda x : z.x)x\), which, by the above definition, is a redex with contractum \(x\), and contracting this redex in the original term leads to \(\lambda x : y.x\). Now \(\lambda x : z.x\) and \(\lambda x : y.x\) are both in normal form, and they are clearly distinct. Thus, adding these "\(\eta\)-redexes" would lead to a failure of the Church–Rosser Theorem. This example appears in [31, p. 71].

The Church–Rosser Theorem holds for this reduction, as is proved in [25] and [14, Ch. 2].

The formulas of all the formulations we consider will be of the form \(M : A\), where \(M\) and \(A\) are terms. The letters \(E, F, G, E_1\), etc. will be used for formulas.

Type assumptions are sequences of the form \(x_0 : A_0, \ldots, x_n : A_n\), where \(x_0, \ldots, x_n\) are variables and \(A_0, \ldots, A_n\) are terms. Capital Greek letters will denote type assumptions.

The two constants \(\text{Prop}\) and \(\text{Type}\) are called kinds. The lower case Greek letters \(\kappa\) and \(\kappa'\) will be used for variables that range over kinds. Thus, \(\kappa\) and \(\kappa'\) are each either \(\text{Prop}\) or \(\text{Type}\).

Definition 1. The system \(\text{TOCO}\) is a natural deduction system the steps of whose deductions are formulas. It has one axiom, namely

\[(\text{PT})\quad \text{Prop} : \text{Type}\]
The rules are as follows:

\((\text{KK'F})\) If \(x\) does not occur free in \(A\) or in any undischarged assumption,

\[
\begin{array}{c}
A : \kappa \\
B : \kappa'
\end{array} \quad \frac{[x:A]}{ \forall x : A \, B : \kappa'},
\]

\((\text{Eq'k})\)

\[
\frac{A : \kappa \quad A =_{e} B}{B : \kappa},
\]

\((\forall v)\)

\[
\frac{M : (\forall x : A) \, B \quad N : A}{MN : [N/x] \, B},
\]

\((\forall \text{ki})\) If \(x\) does not occur free in \(A\) or in any undischarged assumption,

\[
\begin{array}{c}
M : B \\
A : \kappa
\end{array} \quad \frac{[x:A]}{\lambda x : A \, . \, M : (\forall x : A) \, B},
\]

\((\text{Eq''})\)

\[
\frac{M : A \quad A =_{e} B}{M : B}.
\]

If \(\Gamma\) is a sequence of assumptions, then \(\Gamma \vdash M : A\) holds in TOC0 if there is a deduction whose last formula is \(M : A\) and in which every undischarged assumption occurs in \(\Gamma\).

In [37–39], \((\text{KK'F})\) is called \((\text{KK'Formation})\).

In writing out deductions, we shall omit the second premise of rules \((\text{Eq'k})\) and \((\text{Eq''})\), since it can be easily inferred from the other premise and the conclusion.

2. The basic metatheory

TOC0 differs from other typed lambda-calculi in that the types are determined by the deductive rules. The intention is, however, that types will be terms \(A\) such that \(A : \kappa\) can be proved for some kind \(\kappa\). As we shall see, this idea will have to be slightly generalized, but it will do as a first approximation. As it happens, the rules of TOC0 strictly limit the terms that can be proved to have Type as a type. To state the required result, we need a definition:

**Definition 2.** Terms which convert to the form

\[(\forall x_1 : A_1)(\forall x_2 : A_2)\ldots(\forall x_n : A_n)\text{Prop}\]
are called contexts. Terms which convert to the form

$$(\forall x_1 : A_1)(\forall x_2 : A_2)\ldots(\forall x_n : A_n)\text{Type}$$

are called supercontexts. In the presence of an assumption of the form

$$x : (\forall y_1 : B_1)(\forall y_2 : B_2)\ldots(\forall y_m : B_m)\text{Type},$$

and deductions of $M_1 : B_1, M_2 : B_2, \ldots, M_m : B_m$, terms which convert to the form

$$(\forall x_1 : A_1)(\forall x_2 : A_2)\ldots(\forall x_n : A_n)(xM_1M_2\ldots M_m)$$

are called generalized contexts (with tail variable $x$). The given form of each of these kinds of term is called its standard form of the term, $n$ is called its index, and $A_1, A_2, \ldots, A_n$ are called its prefix types. A context function [respectively supercontext function, generalized context function] of degree $k$ is a term $A$ such that for terms $M_1, M_2, \ldots, M_k$ of appropriate types, $AM_1M_2\ldots M_k$ is a context [respectively supercontext, generalized context].

If $A'$ is the standard form of a context [respectively supercontext, generalized context], then

$$\lambda x_1 : A_1 . \lambda x_2 : A_2 . \ldots . \lambda x_k : A_k . A'$$

is a context [respectively supercontext, generalized context] function of degree $k$. Once we have proved the normalization theorem, we shall see that the normal form of every context function either has this form or else is a generalized context function of the form $xM_1M_2\ldots M_j$, where $x$ is a variable which is also a generalized context function of degree $j + k$.

It follows from the Church–Rosser Theorem that two standard forms can be the standard form of the same context, generalized context, or supercontext if and only if they have the same index, the prefix types are convertible, and, in the case of a generalized context, the tail variable is the same, the number of its arguments is the same, the corresponding arguments are convertible. It also follows that no one of these three types of term converts to any other.

Noting that Type is a supercontext, we can state the result we want as follows.

**Theorem 1.** If, for any set of assumptions $\Gamma$,

$$\Gamma \vdash A : B,$$

where $B$ is a supercontext of index $k$, then $A$ is a context function or a generalized context function of degree $k$.

**Proof.** By induction on the length of the deduction of $\Gamma \vdash A : B$. For the basis, note that if $A : B$ is an instance of the axiom, then $B$ is Type, a supercontext of index 0,
and $A$ is $\text{Prop}$, which is a context (a context function of degree 0), while if $A : B$ is an assumption, then $A$ is a variable which, under Definition 2, is a generalized context function of degree $k$. For the induction step, note that if the type in the conclusion of any rule except $(\text{MC'} F)$ is a context, then so is the type in the appropriate premise (the left premise of every rule except $(\text{MC'} F)$, where it is the right premise), so by the induction hypothesis the term in this premise satisfies the theorem, and hence so does the term in the conclusion of the rule. (Note that if the rule is $(\text{JCIC'} F)$ or $(\text{Eq'c})$, then $B$ is $\text{Type}$, which is a supercontext of index 0, and in this case $A$ is a context or a generalized context.) □

Corollary 1.1. If, for any set of assumptions $\Gamma$,

$$\Gamma \vdash A : \text{Type},$$

then $A$ is a context or a generalized context.

Now consider the conditions for the discharge of assumptions. In order to discharge a sequence of assumptions

$$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$$

in reverse order, we need the following conditions for each $i$, $1 \leq i \leq n$:

1. $x_i$ does not occur free in $A_1, \ldots, A_i$ (but it may occur free in $A_{i+1}, \ldots, A_n$), and
2. $x_1 : A_1, \ldots, x_{i-1} : A_{i-1} \vdash A_i : \kappa$.

Since we will also want supercontexts among the types of assumptions, this leads to the following definition.

Definition 3. A sequence of assumptions

$$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$$

is a well-formed environment provided that the following conditions hold for each $i$, $1 \leq i \leq n$:

1. $x_i$ does not occur free in $A_1, \ldots, A_i$ (but it may occur free in $A_{i+1}, \ldots, A_n$), and
2. if $A_i$ is not a supercontext, then $x_1 : A_1, \ldots, x_{i-1} : A_{i-1} \vdash A_i : \kappa$.

To get a partial converse to Corollary 1.1, note that to a standard form of any context or generalized context with prefix types $A_1, A_2, \ldots, A_n$, there corresponds a sequence of assumptions

$$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n.$$

Definition 4. A standard form of a context or generalized context is said to be well-formed if and only if the corresponding sequence of assumptions is a well-formed environment and none of the prefix types is a supercontext.
Theorem 2. If $A$ is a context or generalized context with a well formed standard form, then
\[ \Gamma \vdash A : \text{Type}, \]
where $\Gamma$ is void if $A$ is a context and consists of the assumptions needed to assign types to the tail variable of $A$ and its arguments if $A$ is a generalized context.

Proof. Suppose the well formed standard form of $A$ is
\[ (\forall x_1 : A_1)(\forall x_2 : A_2)\ldots(\forall x_n : A_n)B. \]
If $A$ is a context, then $B \equiv \text{Prop}$, and so $B : \text{Type}$ is an instance of the axiom. If $A$ is a generalized context and $\Gamma$ is as in the theorem, then $\Gamma \vdash B : \text{Type}$. In either case,
\[ \Gamma, x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n \vdash B : \text{Type}. \]
The theorem follows by repeated inferences by ($\kappa F$), which are valid by the assumptions of the theorem. \hfill \Box

Theorem 3. (a) If $\Gamma$ is a well-formed environment and if
\[ \Gamma \vdash M : A, \]
then $M$ reduces to a term in which there is no occurrence of $\text{Type}$.
(b) If $\Gamma$ is a well formed environment and if
\[ \Gamma \vdash M : A, \]
by a deduction from which it can be determined that there is an occurrence of $\text{Type}$ in every term to which $A$ reduces, then $A$ is a supercontext.

Proof. (a) By induction on the deduction of
\[ \Gamma \vdash M : A. \]
In the cases for rules ($\text{Eq'} \kappa$), the conclusion follows via the Church–Rosser Theorem and the fact that no reduction can introduce an occurrence of $\text{Type}$ into a term. The remaining cases are easy.
(b) By induction on the deduction of
\[ \Gamma \vdash M : A. \]
The only difficult case is that for rule ($\forall e$); in this case, suppose that the last inference is
\[ M : (\forall x : B)C \quad N : B \quad \frac{\text{[N/x]C}}{MN : [N/x]C}. \]
If it can be determined from the deduction that there is an occurrence of $\text{Type}$ in every term to which $[N/x]C$ reduces, then since by (a) $N$ reduces to a term in which there is
no occurrence of \textit{Type}, it follows that there is an occurrence of \textit{Type} in every term to which \(C\) reduces. Hence, there is an occurrence of \textit{Type} in every term to which \((\forall x : B)C\) reduces. Thus, by the induction hypothesis (on the left premise), \((\forall x : B)C\) is a supercontext. It follows that \(C\), and hence also \([N/x]C\), is a supercontext. \(\square\)

An occurrence of a subterm \(A\) of a term \(M\) is said to be the \textit{type of a bound variable} if \(A\) is the indicated part of a subterm of the form \(\lambda x : A \cdot N\) or \((\forall x : A)B\).

If \(A\) is a generalized context with standard form

\[(\forall x_1 : A_1)(\forall x_2 : A_2) \ldots (\forall x_n : A_n)(xM_1M_2 \ldots M_m),\]

then \(xM_1M_2 \ldots M_m\) is called a \textit{generalized Prop-term}.

\textbf{Theorem 4.} Let \(\Gamma\) be a well-formed environment, and suppose

\[
\Gamma \vdash M : A,
\]

by a deduction from which it is not possible to determine that \(A\) is a supercontext. Then \(M =_{\star} N\) for some term \(N\) in which every occurrence of \textit{Prop} or a generalized \textit{Prop-term} is inside the type of a bound variable.

\textbf{Remark.} Saying that it is not possible to determine from a deduction that \(A\) is a supercontext means that \(A\) is not a type in a branch of a deduction that occurs above or below a step which would require a subterm of \(A\) to be \textit{Type} in such a way that \(A\) itself would be a supercontext. In other words, it means that as far as the given deduction is concerned, there is nothing in any of the inference steps that would prevent \(A\) being replaced by some term which is known not to be a supercontext. This property is thus decidable.

\textbf{Proof.} By induction on the deduction of \(\Gamma \vdash M : A\). \(\square\)

\textbf{Corollary 4.1.} If \(\Gamma\) is a well-formed environment, and if

\[
\Gamma \vdash M : A,
\]

by a deduction from which it is not possible to determine that \(A\) is a supercontext, then \(M\) is neither a context function nor a generalized context function.

\textbf{Corollary 4.2.} If \(\Gamma\) is a well-formed environment, and if

\[
\Gamma \vdash M : \kappa \quad \text{and} \quad \Gamma \vdash M : \kappa',
\]

then \(\kappa \equiv \kappa'\).

\textbf{Proof.} Otherwise, we have both \(\Gamma \vdash M : \text{Prop}\) and \(\Gamma \vdash M : \text{Type}\), from which we get by Theorem 1 that \(M\) is a context function or generalized context function and from Corollary 4.1 that this is not the case. \(\square\)
TOC0 has the property that deductions follow the construction of the terms to which they assign types. Curry called this the subject construction property; see [12, Corollary 9B1; 13, Theorem 14D1; 18, Note 14.18, Note 15.12, Remark 16.37]. This, in turn, means that it is reasonable to expect that there is a natural transformation on deductions corresponding to the reduction of a term. It turns out that there is such a transformation, but it depends on the subject-reduction theorem [18, Theorem 16.41], which, in turn, depends on the replacement lemma [18, Lemma 16.39]. The latter result is substantial enough to be called a theorem here.

**Theorem 5.** Let \( \Gamma_1 \) be any well formed environment, and let \( \mathcal{D} \) be a deduction of

\[
\Gamma_1 \vdash M : A.
\]

Let \( V : C \) be any statement in \( \mathcal{D} \), let \( \mathcal{D}_1 \) be that part of \( \mathcal{D} \) ending in \( V : C \), let \( \mathcal{D}_2 \) be the rest of \( \mathcal{D} \), and let \( x_1 : B_1, x_2 : B_2, \ldots, x_n : B_n \) be the assumptions in \( \mathcal{D}_1 \) that are discharged in \( \mathcal{D}_2 \) and let \( y_1 : C_1, y_2 : C_2, \ldots, y_m : C_m \) be the assumptions in \( \mathcal{D}_2 \) discharged in \( \mathcal{D}_2 \). Let \( W \) be a term such that \( W =_* V \), and suppose that \( \Gamma_2 \) is a well-formed environment in which \( x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m \) do not occur free. Suppose that \( \mathcal{D}_3 \) is a deduction of

\[
\Gamma_2, x_1 : B_1, x_2 : B_2, \ldots, x_n : B_n \vdash W : C.
\]

Then replacing \( \mathcal{D}_1 \) by \( \mathcal{D}_3 \) in \( \mathcal{D} \) results in a deduction \( \mathcal{D}_4 \) of

\[
\Gamma_1, \Gamma_2 \vdash M^* : A,
\]

where \( M^* \) is obtained from \( M \) by replacing appropriate occurrences of \( V \) by \( W \).

**Remark.** It is difficult to describe exactly the replacements which are required to obtain \( M^* \) from \( M \), but it is possible to read the replacement process from the proof. It is worth noting that the part of \( \mathcal{D}_4 \) which is not included in \( \mathcal{D}_3 \) has exactly the same inference rules in the same relative positions as \( \mathcal{D}_2 \) except perhaps for some inferences by (Eq' \( \kappa \)) or (Eq'').

**Proof.** By induction on the structure of \( \mathcal{D}_2 \).

**Basis:** There are two cases.

Case 1: \( \mathcal{D}_2 \) consists of the single statement \( V : C \). Then \( M \) is \( V \), \( M^* \) is \( W \), and \( \mathcal{D}_4 \) is just \( \mathcal{D}_3 \).

Case 2: \( \mathcal{D}_2 \) consists only of the axiom (PT). Then the replacement is vacuous, \( M^* \equiv M \equiv \text{Prop} \), and \( \mathcal{D}_4 \) consists only of the axiom (PT).

**Induction step:** We have the following cases depending on the last inference in \( \mathcal{D}_2 \).
Case 1: The last inference of $D_2$ is by $(\&x \;F)$. Then $A$ is $\kappa'$, $M$ is $(\forall x : B)E$, and $D$ is

$$ \frac{1}{\frac{[x : B]}{D_5}} \frac{B : \kappa}{\frac{E : \kappa'}{D_6(x)}} \frac{(\forall x : B)E : \kappa'}{(\&x \;F) - 1} $$

where the occurrence of $V : C$ is either in $D_5$ or in $D_6(x)$. By the induction hypothesis, the replacement of $D_1$ by $D_3$ in $D_5$ and $D_6(x)$ leads to deduction $D_7$ and $D_8(x)$ of, respectively,

$$ \Gamma_1, \Gamma_2 \vdash B^* : \kappa $$

and

$$ \Gamma_1, \Gamma_2, x : B \vdash E^* : \kappa' $$

for appropriate $B^*$ and $E^*$. Now $x$, because it is discharged in $D_2$, is either one of the $x_i$ or else it is one of the $y_j$, and hence it does not occur free in $\Gamma_2$. (It clearly does not occur in $\Gamma_1$.) Since $V =_* W$, $B^* =_* B$, and so $D_4$ is as follows:

$$ \frac{1}{\frac{[x : B]}{D_7}} \frac{B^* : \kappa}{\frac{E^* : \kappa'}{D_8(x)}} \frac{(\forall x : B)E^* : \kappa'}{(\&x \;F) - 1} $$

Case 2: The last inference in $D$ is by $(Eq'x)$. Then $A$ is $\kappa$ and $D$ is

$$ \frac{N : \kappa}{M : \kappa} \quad (Eq' \kappa) $$

where $N =_* M$. By the induction hypothesis, the replacement of $D_1$ by $D_3$ in $D_5$ leads to a deduction $D_6$ of

$$ \Gamma_1, \Gamma_2 \vdash N^* : \kappa $$

for an appropriate $N^*$. Since $N^* =_* N =_* M$, we can take $M^* \equiv M$, and then $D_4$ is obtained from $D_6$ by an inference by $(Eq' \kappa)$.

Case 3: The last inference of $D$ is by $(\forall e)$. Then $M$ is $M_1M_2$, $A$ is $[M_2/x]A'$, and $D$ is

$$ \frac{M_1 : (\forall x : B)A'}{M_1M_2} \quad \frac{M_2 : B}{M_2} \quad (\forall e) $$

$$ \frac{M_1M_2 : [M_2/x]A'}{D_5} \quad \frac{D_6}{} \quad \frac{D_7}{} \quad \frac{D_8}{} $$
By the induction hypothesis, the replacement of $D_1$ by $D_3$ in $D_5$ and $D_6$ leads to deductions $D_7$ and $D_8$ of

$$
\Gamma_1, \Gamma_2 \vdash M_1^* : (\forall x : B)A'
$$

and

$$
\Gamma_1, \Gamma_2 \vdash M_2^* : B
$$

respectively, for appropriate $M_1^*$ and $M_2^*$. Furthermore, $M_2^* =_\varepsilon M_2$. Hence, $D_4$ is

$$
\frac{M_1^* : (\forall x : B)A' \quad M_2^* : B}{M_1^* M_2^* : [M_2^*/x]A'} \quad (\forall e)
$$

By hypothesis, $x$ does not occur free in $\varepsilon, r_2$. Then $58$ is as follows:

$$
\begin{array}{c}
[\xi : B] \\
D_5(x) \\
N : E \\
\lambda x : B \cdot N : (\forall x : B)E.
\end{array}
$$

By the induction hypothesis, the replacement of $D_1$ by $D_3$ in $D_5(x)$ and $D_6$ leads to deductions $D_7$ and $D_8$ of

$$
\Gamma_1, \Gamma_2, x : B \vdash N^* : E
$$

and

$$
\Gamma_1, \Gamma_2 \vdash B^* : \kappa
$$

respectively, for appropriate $N^*$ and $B^*$, where $B^* =_\varepsilon B$. By hypothesis, $x$ does not occur free in $\Gamma_1, \Gamma_2$. Then $D_4$ is as follows:

$$
\frac{[\xi : B] \quad D_8}{D_7(x)}
\frac{N^* : E \quad B^* : \kappa}{\lambda x : B \cdot N^* : (\forall x : B)E.} \quad (\forall \kappa - 1)
$$

Case 4: The last inference of $D$ is by $(\forall \kappa i)$. Then $A$ is $(\forall x : B)E$, $M$ is $\lambda x : B \cdot N$, and $D$ is

$$
\begin{array}{c}
1 \\
[x : B] \\
D_5(x) \\
N : E \\
\lambda x : B \cdot N : (\forall x : B)E.
\end{array}
$$

Case 5: The last inference of $D$ is by $(\text{Eq''})$. Then $D$ is

$$
\frac{M : B}{M : A} \quad (\text{Eq''})
$$
where \( A =_* B \). By the induction hypothesis, the replacement of \( \mathcal{D}_1 \) in \( \mathcal{D}_3 \) in \( \mathcal{D}_5 \) leads to a deduction \( \mathcal{D}_6 \) of

\[
\Gamma_1, \Gamma_2 =_* M^*: B
\]

for appropriate \( M^* \), and \( \mathcal{D}_4 \) is obtained by adding an inference by \((\text{Eq}^\prime)\) at the end.

It is now possible to prove the subject-reduction theorem.

**Theorem 6.** Let \( \Gamma \) be a well formed environment. If

\[
(1) \quad \Gamma \vdash M : A
\]

and \( M \supset N \), then

\[
(2) \quad \Gamma \vdash N : A.
\]

**Proof.** By Theorem 5, it is sufficient to prove the theorem for the case in which \( M \) is a redex and \( N \) its contractum. Thus, suppose \( M \equiv (\lambda x : B . M_1)M_2 \) and \( N \equiv [M_2/x]M_1 \). By the subject-construction property, (1) is the conclusion of an inference by \((\forall e)\) whose premises are

\[
(3) \quad \Gamma \vdash \lambda x : B . M_1 : (\forall x : B)E
\]

and

\[
(4) \quad \Gamma \vdash M_2 : B,
\]

where \( A \equiv [M_2/x]E \). (It might be that (1) is the conclusion of an inference by \((\text{Eq}^\prime)\), but then we need only delete that inference to get to the case given above. Note that the transitivity of \( =_* \) makes it possible to eliminate successive inferences by either \((\text{Eq}^\prime)\) or \((\text{Eq}^\prime)\).) By the subject-reduction property again, (3) is the conclusion of an inference by \((\text{Eq}^\prime)\) whose premise is

\[
\lambda x : B . M_1 : (\forall x : B')E',
\]

where \( B' =_* B \) and \( E' =_* E \), and this, in turn, must be the conclusion of an inference by \((\forall ki)\) whose premise is

\[
(5) \quad \Gamma, x : B' \vdash M_1 : E',
\]

where \( x \) does not occur free in \( \Gamma \). Then (2) follows by (4) and (5) and Theorem 5.

\( \square \)

(Cf. [18, Theorems 15.17, 16.41].) This theorem is closely related to the reduction steps on deductions:

**Deduction reductions.** A deduction of the form
reduces to

\[ \frac{\mathcal{D}_3}{\mathcal{D}_4} \]

\[ \frac{N : C}{(\forall x : A.M)N} \]

where \( \mathcal{D}_4 \) is obtained from \( \mathcal{D}_4 \) by replacing appropriate occurrences of \( (\lambda x : A.M)N \) by \( [N/x]M \). The formula \( \lambda x : A.M : (\forall x : A)B \) is called the cut formula of the reduction step. A reduction is a (possibly empty) sequence of replacements using these reduction steps.

A special case of a deduction reduction is a context reduction step or c-reduction step, in which the type of the cut formula is a context, a supercontext, or a generalized context. A context reduction or c-reduction is a reduction in which each reduction step is a c-reduction step. A deduction will be said to be context normalized or c-normalized if it contains no cut formulas whose types are contexts or supercontexts (i.e., if no c-reduction steps are possible). It turns out to be easy to prove that every deduction can be reduced to a c-normalized deduction using the notion of the complexity of a term, and that this partial normalization result is important in proving the full normalization theorem.

**Remark 5.** What is here called the ‘complexity’ of a term was called the ‘degree’ of a term in [37], but here the word ‘degree’ refers to the number of arguments of a context function [respectively supercontext function, generalized context function] needs to become a context [respectively supercontext, generalized context], and so it is better to change the term used for this notion.

**Definition 5.** Let \( A \) be a term such that there is a step \( M : A \) in a deduction in TOC0. Then the complexity of \( A \) relative to the deduction is as follows:

(a) if \( A \) is not a context, a supercontext, or a generalized context, then the complexity of \( A \) is 0;
(b) the complexity of \( \text{Prop} \), \( \text{Type} \), and of a generalized context of the form \( \lambda M_1 M_2 \ldots M_n \) is 1;
(c) the complexity of \((\forall x : A)B\) is one more than the maximum of the complexities of \(A\) and \(B\); and
(d) if \(A =_* B\), then the complexity of \(A\) is equal to the complexity of \(B\).

Since only contexts, supercontexts, and generalized contexts have nonzero complexity, the degree of a term relative to a deduction is well-defined.

**Remark 6.** In general, it is not decidable whether a term is a context, a supercontext, or a generalized context. However, in this definition, what is needed is that the term does not need to be a context or supercontext or generalized context for the deduction to be valid, and given a deduction this is decidable.

**Theorem 7.** Every deduction in TOC0 with conclusion \(M : A\) can be reduced to a c-normalized deduction with the same undischarged assumptions and with conclusion \(N : A\) where \(M \triangleright N\).

**Proof.** Let the complexity of a cut formula be the complexity of its type with respect to the deduction. Let the index of a deduction be the pair \(\langle c, n \rangle\), where \(c\) is the maximum complexity of any cut formula and \(n\) is the number of cut formulas in the deduction with complexity \(d\). Let the pairs be ordered by specifying that \(\langle c, n \rangle < \langle c', n' \rangle\) if either \(c < c'\) or \(c = c'\) and \(n < n'\). At each step, choose a cut formula of complexity \(c\) such that there is no cut formula of complexity \(c\) either above the cut formula or above the minor (right) premise for the inference by \((\forall e)\) for which the cut formula is the major (left) premise. An examination of the deduction that results from applying this c-reduction step shows that there is one less cut formula with complexity \(c\), so that if, in the original deduction, \(n > 1\), then the index of the resulting deduction is \(\langle c, n - 1 \rangle\); otherwise, the index of the resulting deduction is \(\langle d, m \rangle\) for some \(m\) and for some \(d < c\). It follows that the index of the resulting deduction is less than the index of the original deduction. Since this is true for any c-reduction step satisfying the above conditions, the process must terminate. But if a c-reduction step is possible, then it is always possible to find a c-reduction step satisfying these conditions. Hence, every deduction can be c-reduced to a c-normalized deduction. In this reduction, the term in the conclusion is reduced. \(\square\)

**Definition 6.** The term \(N\) of Theorem 7 will be called a c-normal form of \(M\).

In terms of this definition, Theorem 7 says that every term to which a type is assigned by TOC0 has a c-normal form. This fact makes it possible to prove for terms whose type is \(\text{Prop}\) some of the properties we have for terms whose type is \(\text{Type}\). We start with some lemmas:

**Lemma 1.** Let \(\mathcal{D}\) be a c-normal deduction of \(\Gamma \vdash A : \text{Prop}\).
where \( \Gamma \) is a well-formed environment. Then either \( A =_{*}(\forall x : B)C \) for terms \( B \) and \( C \) and a variable \( x \) which does not occur free in \( \Gamma \), or \( A =_{x}M_{1}M_{2} \ldots M_{p} \) for a variable \( x \), a natural number \( p \) (which may be 0), and some terms \( M_{1}, M_{2}, \ldots, M_{p} \), and furthermore it can be decided constructively which of these alternatives holds.

**Proof.** Consider the last inference in \( \mathcal{D} \) which is not be (Eq") or (Eq'P). This inference cannot be by (\( \forall xi \)) since the type of the conclusion is an atomic constant, so the only possible rules are (kP F) and (Ve). Which of these rules it is can be determined constructively by an inspection of the deduction.

If the inference is by (kP F), then the second alternative holds.

If the inference is by (Ve), then consider the left branch of the deduction. As we proceed upwards from the last inference, the only rules we can find are by (Ve) or (Eq'P); for otherwise we would come to an inference by (\( \forall xi \)) whose conclusion is a context, contradicting the assumption that the deduction is c-normal. The formula at the top of this branch cannot be discharged, and so it must be in \( \Gamma \). It follows that this formula must have the form \( x : B \) where \( B =_{*}(\forall x_{1} : C_{1})(\forall x_{2} : C_{2}) \ldots (\forall x_{p} : C_{p}) \text{Prop} \) for some natural number \( p \) (which may be 0), and so we must have the first alternative of the theorem. \( \square \)

**Definition 7.** If \( \mathcal{D} \) is a deduction as in Lemma 1, then it is called **compound** if the first case of the lemma holds and **simple** if the second case holds. If \( A \) is a term such that \( A : \text{Prop} \) is the conclusion of such a deduction \( \mathcal{D} \), then \( A \) will be called **simple** or **compound** according to whether \( \mathcal{D} \) is simple or compound.

**Lemma 2.** If there is a deduction of

\[ \Gamma \vdash A : \text{Prop}, \]

then there is a c-normal deduction of it.

**Proof.** Let \( \mathcal{D} \) be the given deduction. By Theorem 7 there is a c-normal deduction of

\[ \Gamma \vdash B : \text{Prop}, \]

where \( A \triangleright B \). By adding one inference by (Eq'P) at the end, we get the desired c-normal deduction of

\[ \Gamma \vdash A : \text{Prop}. \] \( \square \)

By Lemma 2 and Definition 7, every type in \text{Prop} (with respect to a given well-formed environment) is either simple or compound, and it is possible to decide constructively which it is. Furthermore, the compound types are formed by repeated use of the operation \( \forall \) from the simple types and \text{Prop}. Note that the contexts are formed in more or less the same way.
Lemma 3. If \( \mathcal{D} \) is a deduction of
\[
\Gamma \vdash (\forall x : A)B : \text{Prop},
\]
where \( x \) does not occur free in \( \Gamma \) or in \( A \) and where \( \Gamma \) is a well formed-environment, then there is a deduction \( \mathcal{D}' \) of
\[
\Gamma, x : A \vdash B : \text{Prop}.
\]
Furthermore, the c-normal deduction to which \( \mathcal{D}' \) reduces has fewer inferences by rules other than (Eq") and (Eq'k) than the c-normal deduction to which \( \mathcal{D} \) reduces.

Proof. This follows from Lemmas 1 and 2. \( \square \)

Theorem 8. If
\[
\Gamma \vdash M : A,
\]
where \( \Gamma \) is a well-formed environment, then exactly one of the following holds:
1. \( \Gamma \vdash A : \text{Prop} \),
2. \( \Gamma \vdash A : T \), where \( T \) is a supercontext, or
3. \( A \) is a supercontext.

Proof. By induction on the length of the deduction \( \mathcal{D} \) with the conclusion \( M : A \). The only difficult case is that in which the last inference of \( \mathcal{D} \) is by rule (\( \forall e \)). In this case, \( M \equiv PN \), \( A \equiv [N/x]C \), and \( \mathcal{D} \) has the form
\[
\begin{array}{c}
\mathcal{D}_1 \\
P : (\forall x : B)C \\
\hline
PN : [N/x]C.
\end{array}
\]

By the induction hypothesis, exactly one of the following holds:
1. \( \Gamma \vdash (\forall x : B)C : \text{Prop} \),
2. \( \Gamma \vdash (\forall x : B)C : T \), where \( T \) is a supercontext, or
3. \( (\forall x : B)C \) is a supercontext.

This gives us three subcases.

Subcase 1: By Lemma 3, there is a deduction of
\[
x : B \\
\mathcal{D}'(x) \\
C : \text{Prop}.
\]

Then we get case 1 of the theorem as follows:
\[
\begin{array}{c}
\mathcal{D}_2 \\
N : B \\
\mathcal{D}'(N) \\
[N/x]C : \text{Prop}.
\end{array}
\]
Subcase 2: The last non-(Eq') inference in the deduction of
\[ \Gamma \vdash (\forall x : B)C : T \]
must be (kTF). Then
\[ T = \ast \text{ Type} \]
and the deduction is
\[ \frac{\text{D}_3}{\frac{\text{D}_4(x)}{B : \kappa \quad C : \text{Type} \quad (\forall x : B)C : \text{Type}. \quad (\kappa \text{TF} - 1).}}{1 \quad [x : B] \quad [N/x]C : \text{Type}.} \]

We get case 2 of the theorem as follows:

\[ \frac{\text{D}_2}{\frac{\text{D}_4(N)}{N : B \quad [N/x]C : \text{Type}.}} \]

Subcase 3: If \((\forall x : B)C\) is a supercontext, then so is \(C\), and hence, so also is \([N/x]C\).  

Lemmas 1 and 2 give us a structure on the types in Prop. It is interesting to note that the other types have exactly the same structure. By Theorem 8, every type is in Prop or Type or is a supercontext. It is clear from the definition that supercontexts have this structure, and Corollary 1.1 tells us that the same is true for contexts. What all of this means is that types are built up from Type, Prop, and the simple types by the operation of forming \((\forall x : A)B\).

Corollary 1.1 and Theorems 2 and 8 allow us to classify all formulas which can be deduced from well-formed environments.

Definition 8. A formula \(M : A\) is called:
(a) a context function if \(A\) is a supercontext;
(b) a context if \(A = \ast \text{ Type}\);
(c) a proposition function if \(A\) is a context;
(d) a proposition if \(A = \ast \text{ Prop}\); and
(e) a proof if \(A\) is neither a context nor a supercontext.

A deduction whose undischarged assumptions form a well-formed environment is classified according to its last formula.

This classification shows the connection between TOCO and the formulas-as-types isomorphism.
We would like to extend this classification to the terms \( M \) (at least relative to a given well-formed environment). In other words, we modify Definition 8 as follows.

**Definition 9.** A term \( M \) is called:

(a) a \( \Gamma \)-context function if there is a supercontext \( A \) such that \( \Gamma \vdash M : A \);

(b) a \( \Gamma \)-context if \( \Gamma \vdash M : Type \);

(c) a \( \Gamma \)-proposition function if there is a context \( A \) such that \( \Gamma \vdash M : A \);

(d) a \( \Gamma \)-proposition if \( \Gamma \vdash M : Prop \); and

(e) a \( \Gamma \)-proof if there is a term \( A \) which is neither a context nor a supercontext such that \( \Gamma \vdash M : A \).

We have already proved (Corollary 4.1) that no term is both a \( \Gamma \)-context function and a \( \Gamma \)-proposition function or both a \( \Gamma \)-context function and a \( \Gamma \)-proof. To complete the proof that this classification is exclusive, we need the following result.

**Theorem 9.** If \( \Gamma \) is a well-formed environment, and if

\[
\Gamma \vdash M : A \quad \text{and} \quad \Gamma \vdash M' : B,
\]

are both derivable, where \( M \) and \( M' \) differ only by changes of bound variables, then \( A = B \).

**Proof.** By induction on the lengths of the two deductions, \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \), respectively.

**Case 1:** The last inference in \( \mathcal{D}_1 \) is by \((Eq''\), Assume that the left premise is \( M : A' \). By the induction hypothesis, \( A' = B \). But \( A = A' \), and so \( A = B \).

**Case 2:** The last inference in \( \mathcal{D}_2 \) is by \((Eq''\). Symmetric to Case 1.

**Case 3:** The last inference in neither \( \mathcal{D}_1 \) nor \( \mathcal{D}_2 \) is by \((Eq''\).

**Subcase 3.1:** \( \mathcal{D}_1 \) consists of the axiom. Then \( M \) is \( Prop \) and \( A \) is \( Type \). Then either \( \mathcal{D}_2 \) is also the axiom, in which case \( B \) is \( Type \) and we are finished, or else the last inference in \( \mathcal{D}_2 \) is by rule \((Eq'\), in which case \( k \) is \( Type \) by Corollary 4.1.

**Subcase 3.2:** The last inference of \( \mathcal{D}_1 \) is by \((kk'F\). Then \( B \) is \( k' \) by Corollary 4.1.

**Subcase 3.3:** The last inference of \( \mathcal{D}_1 \) is by \((Eq'\). Then by Corollary 4.1, \( B \) is \( k \).

**Subcase 3.4:** The last inference of \( \mathcal{D}_1 \) is by \((\forall e\). Then the last inference of \( \mathcal{D}_2 \) is either \((\forall e)\) or \((Eq'k\). If it is \((Eq'k\), then the theorem follows by Corollary 4.2. Otherwise, \( M \) is \( NP \), \( M' \) is \( N'P' \) (where \( N' \) and \( P' \) differ from \( N \) and \( P \) only by changes in bound variables), \( A \) is \([P/x]A' \), \( B \) is \([P/x]B' \), \( \mathcal{D}_1 \) is

\[
\begin{align*}
\frac{N : (\forall x : C)A'}{NP : [P/x]A'} \quad \frac{P : C}{(\forall e)}
\end{align*}
\]

and \( \mathcal{D}_2 \) is

\[
\begin{align*}
\frac{N : (\forall x : D)B'}{NP : [P/x]B'} \quad \frac{P : B}{(\forall e)}
\end{align*}
\]
By the induction hypothesis, $C =^* D$ and $(\forall x : C)A' =^* (\forall x : D)B'$. It follows that $A' =^* B'$, and hence $A =^* B$.

**Subcase 3.5:** The last inference in $\mathcal{D}_1$ is by $(\forall x_i)$. Then the last inference in $\mathcal{D}_2$ is by $(\forall x_i)$, $M$ is $\lambda x : C . N$, $M'$ is $\lambda x : C . N'$ where $N$ and $N'$ differ by changes in bound variables, $A$ is $(\forall x : C)A'$, and $B'$ is $(\forall x : C)B'$. (There is no loss of generality in assuming that the indicated bound variable is $x$ in both $M$ and $M'$ because if the bound variables are different a minor modification of $\mathcal{D}_2$ will make them the same.) Furthermore, $\mathcal{D}_1$ is

$$
\begin{array}{c}
1 \\
[x : C] \\
\mathcal{D}_{11} \\
N : A' \\
\frac{C : \kappa}{\lambda x : C . N : (\forall x : C)A'} (\forall x_i - 1)
\end{array}
$$

and $\mathcal{D}_2$ is

$$
\begin{array}{c}
1 \\
[x : C] \\
\mathcal{D}_{21} \\
N' : B' \\
\frac{C : \kappa'}{\lambda x : C . N' : (\forall x : C)B'} (\forall x_i' - 1)
\end{array}
$$

By the induction hypothesis, $A' =^* B'$, and it clearly follows that $A =^* B$. □

**Corollary 9.1.** For any well-formed environment $\Gamma$, no term is both a $\Gamma$-proposition function and a $\Gamma$-proof.

**Proof.** Suppose $M$ is both a $\Gamma$-proposition function and a $\Gamma$-proof. Then there is a $\Gamma$-proposition $B$ and a $\Gamma$-context $C$ such that

$$\Gamma \vdash M : B \text{ and } \Gamma \vdash M : C.$$ 

Hence,

$$\Gamma \vdash B : \text{Prop} \text{ and } \Gamma \vdash C : \text{Type}.$$ 

By the theorem, $B =^* C$. Hence, by the Church–Rosser Theorem, there is a term $D$ to which both $B$ and $C$ reduce which can be proved on the basis of $\Gamma$ to be in both $\text{Prop}$ and $\text{Type}$, contradicting Corollary 4.2. □

Theorem 7 gives us the following characterization of $\Gamma$-proposition functions.

**Theorem 10.** If $\Gamma$ is a well-formed environment, and if $A$ is a $\Gamma$-proposition function which is not a proposition, then either each c-normal form of $A$ has the form $\lambda x : B . C$, in which case the type assigned to $A$ by $\Gamma$ converts to $(\forall x : B)F$, where $F$ is a context, or each c-normal form of $A$ has the form $xM_1 \ldots M_n$. 

Proof. By hypothesis, there is a c-normal deduction of
\[ \Gamma \vdash D : (\forall x : B)E, \]
where \( A \supset D \), which is a c-normal form of it, and \( E \) is a context. Except for \((\text{Eq}''')\), which makes no difference, the last inference in this c-normal deduction must be \((\forall \xi i)\) or \((\forall e)\). If it is \((\forall \xi i)\), we are done. If it is \((\forall e)\), then proceed up the left branch to the first formula which is not the conclusion of an inference by \((\forall e)\). Since the deduction is c-normal and since \( \Gamma \) is a context, this formula is not the conclusion of an inference by \((\forall \xi i)\). Hence, it is an assumption, and \( D \) has the form \( xM_1 \ldots M_n \), as desired. (That all c-normal forms of \( A \) are of the same kind follows by the Church–Rosser Theorem.) \( \square \)

By iterating the theorem, and, if necessary, replacing terms \( M \) by \( \lambda y_i : B_i . M y_i \), where \( y_i \) is not free in \( M \), we can prove the following corollary:

**Corollary 10.1.** Under the hypotheses of the theorem, if
\[ \Gamma \vdash A : (\forall x_1 : B_1) \cdots (\forall x_n : B_n) \text{Prop}, \]
then either \( A =* \lambda x_1 : B_1 . \ldots \lambda x_n : B_n . A' \), where \( A' \) is a \( \Gamma \)-context, or else every c-normal form of \( A \) has the form \( xM_1 \ldots M_n \).

3. The strong normalization theorem

It might appear that to prove the normalization theorem it is sufficient to combine Theorem 7 with a similar result for reduction steps whose cut formulas are not propositions. But this fails to work, for on the one hand, such a reduction step may require that a type of arbitrary complexity be substituted for a variable that is part of an assumption, and on the other hand, a reduction step whose cut formula is a proof may introduce a new cut formula which is a proposition and whose type is a context of arbitrarily high degree.

On the other hand, Theorem 7 is of help in proving normalization, for it shows (via Lemma 3) that the types which are proved to be in \( \text{Prop} \) can be formed from the simple types and \( \text{Prop} \) by \( \forall \) in much the same way that the types of the second-order polymorphic typed \( \lambda \)-calculus are formed from type variables by the type constructors. This turns out to make it possible to adapt a proof of normalization for the second-order polymorphic typed \( \lambda \)-calculus to TOCO. The proof we have chosen to adapt is a version of Girard’s proof [17] of strong normalization, the version being due to Stenlund [41, Section 5.6]. However, the proof needs to be modified in much the way that the proof of [24] is modified in [26].

**Convention.** Let \( \mathcal{D} \) be a deduction whose conclusion is \( M : A \), where \( A =* (\forall x_1 : A_1) \cdots (\forall x_n : A_n) B \), and for \( i = 1, \ldots, n \), let \( \mathcal{D}_i \) be a deduction with conclusion \( M_i : A'_i \),
where 

\[ A'_i \equiv [M_1/x_1, \ldots, M_{i-1}/x_{i-1}]A_i. \]

Then

\[ \mathcal{D} \]

\[
M : A \\
\{ \mathcal{D}_1, \ldots, \mathcal{D}_n \}
\]

will denote the deduction

\[
\frac{\mathcal{D}_1}{MM_1 : [M_1/x_1](\forall x_2 : A_2) \cdots (\forall x_n : A_n)B} \quad (Eq'')
\]

\[
\vdots
\]

\[
\frac{MM_1 \ldots M_n : B'}{M_1 : A_1} \quad (\forall e)
\]

where \[ B' \equiv [M_1/x_1, \ldots, M_{n-1}/x_{n-1}]B \] and \[ B'' \equiv [M_1/x_1, \ldots, M_n/x_n]B. \] (If \( n = 0 \), then it will denote \( \mathcal{D} \) itself.)

**Definition 10.** If \( \mathcal{D} \) is a deduction whose conclusion is \( M : A \), then \( A \) is called the **type** of \( \mathcal{D} \).

**Definition 11.** A deduction \( \mathcal{D} \) is said to be strongly normal (SN) if every reduction starting with \( \mathcal{D} \) terminates in a normal deduction.

Our aim is to prove that every deduction is SN.

**Remark 7.** In the proof, we will be making important use of the classifications in Definition 8. We will also be discussing a number of deductions at the same time. It will be important that each formula in each deduction be classified the same way in any other deduction under consideration. For this purpose we will need to know that the well-formed environments of different deductions are all consistent in that none of them have assumptions assigning different types to the same variable. To ensure this consistency, we will assume that we are starting with a generalized well-formed environment \( I_0 \) that is an infinite set rather than a finite sequence of assumptions. All well-formed environments actually considered will draw their assumptions from \( I_0 \), and no variable will be assigned more than one type in \( I_0 \). Furthermore, we shall assume that any finite subset of \( I_0 \) can be extended to a larger finite subset of \( I_0 \) whose elements can be ordered in such a way that it is a well-formed environment. For any deduction under consideration, we shall assume that its discharged assumptions belong to \( I_0 \); such a deduction will be called \( I_0 \)-acceptable. A term which is the type of a
\( \Gamma_0 \)-acceptable deduction will be called a \( \Gamma_0 \)-type. We shall assume that any term is a \( \Gamma_0 \)-type which can be built up from \textit{Prop}, \textit{Type}, and the simple types and simple generalized contexts obtainable from assumptions in \( \Gamma_0 \). (This assumption is easy to satisfy; if we start with a candidate for \( \Gamma_0 \) for which it is not true, we extend it with new assumptions (for new variables), and we keep doing this until there are enough assumptions.) A \( \Gamma_0 \)-proposition variable of type \( A \), where \( A \) is a context, is a variable \( x : A \) is in \( \Gamma_0 \). And finally, a \( \Gamma_0 \)-term of type \( A \) is a term \( M \) such that \( M : A \) is provable from assumptions in \( \Gamma_0 \).

**Definition 12.** A set \( S \) of \( \Gamma_0 \)-acceptable deductions is a \textit{grounded type set (ground)} if the following three conditions are satisfied:

(a) Every deduction in \( S \) is SN;

(b) If \( \mathcal{D}_1(N) \) is a part of a deduction obtained from a deduction

\[
\begin{align*}
x : A \\
\mathcal{D}_1(x) \\
M : B
\end{align*}
\]

by substituting \( N \) for \( x \), if \( \mathcal{D}_3 \) is SN, and if

\[
\begin{align*}
\mathcal{D}_3 \quad N : C \\
\frac{\text{(Eq'')}}{N : A} \\
\mathcal{D}_1(N) \\
[N/x]M : [N/x]B \\
\{\mathcal{D}_1', \ldots, \mathcal{D}_n'\}
\end{align*}
\]

is in \( S \), then

\[
\begin{align*}
1 \\
\mathcal{D}_1(x) \quad \mathcal{D}_2 \\
\frac{M : B}{\lambda x : A . M : (\forall x : A)B} \quad \text{(\( \forall k \text{ i - 1} \))} \\
\frac{\lambda x : A . M : (\forall x : C)B}{(\lambda x : A . M)N : [N/x]B} \quad \text{(Eq'')} \\
\mathcal{D}_3 \\
N : C \quad (\forall e) \\
\{\mathcal{D}_1', \ldots, \mathcal{D}_n'\}
\end{align*}
\]

is also in \( S \); and

(c) If \( \mathcal{D}_1, \ldots, \mathcal{D}_n \) are SN, and if

\[
\begin{align*}
x : A \\
\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}
\end{align*}
\]

is a \( \Gamma_0 \)-acceptable deduction, then it is in \( S \).
A ground in which all of the deductions have a given type \( A \) will be called a **ground of type** \( A \).

**Examples.** The set of all SN \( \Gamma_0 \)-acceptable deductions is a ground. This ground will be called SN. If \( A \) is a \( \Gamma_0 \)-type, then the set of all \( \Gamma_0 \)-acceptable deductions of type \( A \) is a ground of type \( A \); it is called SN(\( A \)).

**Definition 13.** A **proposition term** is a term \( A \) such that \( A : B \) is a proposition. A proposition term which is also a variable is a **proposition variable**. If \( A \) is a \( \Gamma_0 \)-type, then the set of all \( \Gamma_0 \)-acceptable deductions of type \( A \) is a ground of type \( A \); it is called SN(\( A \)).

**Definition 14.** The **rank** of a \( \Gamma_0 \)-type \( A \), \( \text{rk}(A) \), is defined as follows:

(a) if \( A \) is a simple type or a simple generalized context, \( \text{rk}(A) = 0 \);
(b) \( \text{rk}(\text{Prop}) = \text{rk}(\text{Type}) = 0 \);
(c) \( \text{rk}((\forall x : A)B) = \text{rk}(A) + \text{rk}(B) + 1 \); and
(d) if \( A \vdash B \), then \( \text{rk}(A) = \text{rk}(B) \).

**Definition 15.** Let \( M \) be a \( \Gamma_0 \)-term of type \( A \). By induction on \( \text{rk}(A) \), a **computability predicate of type** \( M \), denoted \( p[M] \) is defined as follows:

(a) if \( A \) is not a context, then \( p[M] \equiv M \);
(b) if \( A \vdash \text{Prop} \) or \( \text{Type} \), then \( p[M] \) is a ground of type \( M \); and
(c) if \( A \vdash ((\forall x_1 : A_1) \cdots (\forall x_n : A_n)\text{Prop}) \), then \( p[M] \) is a function whose arguments are computability predicates \( p[M_1], \ldots, p[M_n] \) of types \( M_1, \ldots, M_n \), where each \( M_i \) is a \( \Gamma_0 \)-term of type \( A_i \), and whose value is a ground of type \( MM_1 \cdots M_n \).

For the next definition, we need to proceed by a kind of induction on the structure of a term. For this induction, we need to note that if a term \( A \) is not a \( \Gamma_0 \)-proof, then it is a \( \Gamma_0 \)-proposition function, a \( \Gamma_0 \)-context function, or a supercontext. Thus, if \( A \) is not a \( \Gamma_0 \)-proof, then it converts to \( \text{Prop} \), \( \text{Type} \), a \( \Gamma_0 \)-simple type, a \( \Gamma_0 \)-simple generalized context, \((\forall x : B)C \) (where \( B \) is neither a supercontext nor a proof and where \( C \) is not a proof), or \( \lambda x : B . C \) (where \( B \) is neither a supercontext nor a proof and where \( C \) is neither a supercontext nor a proof). Here \( B \) and \( C \) are essentially simpler than \( A \); furthermore, if \( A \) converts to a simple type \( xM_1 \cdots M_n \), then each \( M_i \) is essentially simpler than \( A \). This justifies the following definition by induction on the “structure of \( A \)”. 
Definition 16. Let $A(x_1, \ldots, x_n)$ be a term all of whose free variables which are not assigned to supercontexts in $I_0$ occur in the list $x_1, \ldots, x_n$. Let $A_1, \ldots, A_n$ be $I_0$-terms of the types of $x_1, \ldots, x_n$, respectively. Let $p[A_1], \ldots, p[A_n]$ be an assignment of computability functions to the terms $A_1, \ldots, A_n$. Relative to this assignment we shall define by induction on the structure of $A(x_1, \ldots, x_n)$ a computability object

$$C[A(x_1, \ldots, x_n)](p[A_1], \ldots, p[A_n]),$$

which will contain deductions of type $A(A_1, \ldots, A_n)$ if $A(x_1, \ldots, x_n)$ is a $I_0$-type. To simplify the notation, we let $x$ be the sequence $x_1, \ldots, x_n$, $A$ the sequence $A_1, \ldots, A_n$, and $p[A]$ be the sequence $p[A_1], \ldots, p[A_n]$.

(a) if $A(x)$ is a $I_0$-proof, then $C[A(x)](p[A])$ is the term $A(A)$ itself;

(b) if $A(x) = \ast Prop, Type$, or a $I_0$-simple generalized context, then $C[A(x)](p[A]) = SN(A(A))$;

(c) if $A(x) = \ast \prod x_1 \cdots x_n \cdot M_1(x) \cdots M_m(x)$ and is neither a $I_0$-proof nor a $I_0$-simple generalized context, then $C[A(x)](p[A])$ is

$$p[A_1](C[M_1(x)](p[A]), \ldots, C[M_m(x)](p[A]));$$

(d) if $A(x) = \ast (\forall x : B(x))C(x, x)$, where $B(x)$ is not a context, then $C[A(x)](p[A])$ is the set of all $I_0$-acceptable deductions

$$\mathcal{D}$$

$$M : A(A)$$

such that if

$$\mathcal{D}^\prime$$

$$N : B(A)$$

is in $C[B(x)](p[A])$, then

$$\frac{M : A(A)}{M : (\forall x : B(A))C(x, A)} (Eq^{''})$$

$$\mathcal{D}^\prime$$

$$N : B(A) (\forall e)$$

$$MN : C(N, A),$$

is in $C[C(N, x)](p[A])$;

(e) if $A(x) = \ast (\forall x : B(x))C(x, x)$ where $B(x)$ is a context, then $C[A(x)](p[A])$ is the set of all $I_0$-acceptable deductions

$$\mathcal{D}$$

$$M : A(A)$$

such that if

$$\mathcal{D}^\prime$$

$$E : B(A)$$

is in $C[B(x)](p[A])$. 


is in $\mathbb{C}[B(x)](p[A])$ and if $p[E]$ is any computability predicate assigned to $E$, then

$$
\frac{M : A(A)}{\mathcal{D}}
\frac{M : (\forall x : B(A))C(x,A)}{(Eq'')} \frac{E : B(A)}{\mathcal{D}'}
\frac{\mathcal{E}}{E : (\forall x : \mathcal{E}(A,E,A),)
\mathcal{E}(x) = C_{A(x)}(p[A]),}
\mathcal{E}(x) = C_{C(x,x)}(p[A],p[A]).
$$

Remark 8. Clause (c) of this definition makes sense only if $\mathbb{C}[A(x)](p[A])$ is a computability predicate. This will be proved below (Lemma 6).

Remark 9. In clause (d), note that since $B(x)$ is not a context and since $N : B(A)$, $C(x,x)$ must have the same structure (with respect to the construction of types) as $C(x,x)$. The division into cases between (d) and (e) is precisely the distinction between terms which can, after substitution, change the structure of the type in an essential way, and dealing with this possible change is one of the main difficulties of the proof. Furthermore, in clauses (d) and (e) of this definition, we are assuming that $x$ does not occur free in $A$. Since $x$ does not occur in $B(A)$, this is immediate for those $A_i$ which actually occur in $B(A)$, and for those which do not occur in $C(x,A)$, there is clearly no problem. For those $A_i$ which occur in $C(x,A)$ but not in $B(A)$, since we automatically change bound variables to avoid clashes when we carry out a substitution, the fact that the bound variable is $x$ implies that it does not occur free in these $A_i$.

Lemma 4. (a) If

$$
\begin{align*}
&\mathcal{D}_1, \ldots, \mathcal{D}_n \\
&x : B \\
&\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}
\end{align*}
$$

for $n \geq 0$ is a deduction of type $A(A)$, and if $\mathcal{D}_1, \ldots, \mathcal{D}_n$ are all SN, then

$$
\begin{align*}
&\mathcal{D}_1, \ldots, \mathcal{D}_n \\
&x : B \\
&\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}
\end{align*}
$$

is in $\mathbb{C}[A(x)](p[A]).$

(b) Every deduction in $\mathbb{C}[A(x)](p[A])$ is SN.

Remark 10. Cf. [18, Theorem A2.3, Lemma 1].

Proof. By induction on the structure of $A(x)$. Note that $A(x)$ is not a $\Gamma_0$-proof and does not convert to $\lambda x : B(x). C(x,x)$. 
Case 1: \( A(x) = \* \) Prop, Type, or a \( \Gamma_0 \)-simple generalized context. Since
\[
x : B
\]
\( \{ D_1, \ldots, D_n \} \)
is SN whenever \( D_1, \ldots, D_n \) are SN, (a) follows by Definition 16(b). Part (b) follows immediately by Definition 16(b).

Case 2: \( A(x) = x_1, M_1 \ldots M_m \) and is not a \( \Gamma_0 \)-generalized context. Part (a) holds by Definition 12(c) and Definitions 15 and 16(b). Part (b) holds by Definition 12(a) and Definitions 15 and 16(b).

Case 3: \( A(x) = (\forall x : B(x))C(x, x) \), where \( B(x) \) is not a context. To prove (a), let
\[
D
\]
\( M : A(A) \)
be a deduction in \( C[A(x)](p[A]) \) and let \( x : B(A) \) be an assumption in \( \Gamma_0 \) for which \( x \) does not occur free in \( D \). (We may assume without loss of generality that the bound variable \( x \) has been changed if necessary to assure that there is such an assumption in \( \Gamma_0 \).) By the induction hypothesis (a) (with \( n = 0 \)), \( x : B(A) \) is in \( C[B(x)](p[A]) \). Hence, by Definition 16(d),
\[
M : A(A)
\]
\[ M : (\forall x : B(A))C(x, A) \quad \text{(Eq'')} \]
\[ Mx : C(x, A) \quad \text{(ve)} \]
is in \( C[C(x,x)](p[A]) \). Hence, by the induction hypothesis (b), this deduction is SN. Hence, \( D \) is SN.

To prove (b), let
\[
y : E
\]
\( \{ D_1, \ldots, D_n \} \)
be a \( \Gamma_0 \)-acceptable deduction of type \( A(A) \) where \( D_1, \ldots, D_n \) are all SN, and let
\[
D
\]
\( N : B(A) \)
be in \( C[B(x)](p[A]) \). By the induction hypothesis (b), \( D \) is SN. Hence, by the induction hypothesis (a),
\[
y : E
\]
\( \{ D_1, \ldots, D_n, D \} \)
is in \( C[C(N,x)](p[A]) \). Hence, by Definition 16(d),
\[
y : E
\]
\( \{ D_1, \ldots, D_n \} \)
is in \( C[A(x)](p[A]) \).
Case 4: \( A(x) \land (\forall x : B(x))C(x, x) \), where \( B(x) \) is a context. To prove (a), let

\[
\mathcal{D} \\
M : A(A)
\]

be in \( C[A(x)](p[A]) \), and let \( x : B(A) \) be an assumption in \( I_0 \). By the induction hypothesis (a) (with \( n = 0 \)), \( x : B(A) \) is in \( C[B(x)](p[A]) \). By Definition 16(e),

\[
\frac{M : A(A)}{M : (\forall x : B(A))C(x, A) (\text{Eq}''') \quad x : B(A) (\forall e)}
\]

is in \( C[C(x, x)](p[x], p[A]) \) for all \( p[x] \). By the induction hypothesis (b), it is SN. Hence, \( \mathcal{D} \) is SN.

To prove (b), let

\[
y : E \\
\{ \mathcal{D}_1, \ldots, \mathcal{D}_n \}
\]

be an \( I_0 \)-acceptable deduction of type \( A(A) \) where \( \mathcal{D}_1, \ldots, \mathcal{D}_n \) are all SN, and let

\[
\mathcal{D} \\
F : B(A)
\]

be in \( C[B(x)](p[A]) \). By the induction hypothesis (b), \( \mathcal{D} \) is SN. Hence, by the induction hypothesis (a),

\[
\frac{y : E}{\{ \mathcal{D}_1, \ldots, \mathcal{D}_n, \mathcal{D} \}}
\]

is in \( C[C(x, x)](p[F], p[A]) \) for all \( p[F] \). Hence, by Definition 16(d),

\[
\frac{y : E}{\{ \mathcal{D}_1, \ldots, \mathcal{D}_n \}}
\]

is in \( C[A(x)](p[A]) \). \( \square \)

Lemma 5. If \( \mathcal{D}_1(N) \) is a part of a deduction obtained from a deduction

\[
x : E \\
\mathcal{D}_1(x) \\
M : B
\]
by substituting $N$ for $x$, if $\mathcal{D}_3$ is SN, and if

$\begin{align*}
N : C \\
\frac{N : E}{\mathcal{D}_1(N)} \\
[N/x]M : [N/x]B \\
\{\mathcal{D}_1', \ldots, \mathcal{D}_n'\}
\end{align*}$

is in $\mathcal{C}[A(x)](p[A])$, then

$\begin{align*}
1 \\
[x : E] \\
\mathcal{D}_1(x) \\
\mathcal{D}_2 \\
\frac{M : B}{E : \kappa} \\
\frac{\lambda x : E.M : (\forall x : E)B}{(\forall \kappa - 1)} \\
\frac{\lambda x : E.M : (\forall x : C)B}{(\text{Eq}''')} \\
\frac{N : C}{\mathcal{D}_3} \\
\frac{\mathcal{D}_1' = \mathcal{D}_1, \ldots, \mathcal{D}_n' = \mathcal{D}_n}{(\forall e)} \\
\end{align*}$

is also in $\mathcal{C}[A(x)](p[A])$.

**Remark 11.** Cf. [18, Theorem A2.3 Lemma 2].

**Proof.** By induction on the structure of $A(x)$. Again, $A(x)$ is not a $\Gamma_0$-proof and does not convert to $\lambda x : B(x). C(x, x)$.

**Case 1:** $A(x) = *_{\text{Prop, Type}}$, or a $\Gamma_0$-simple generalized context. The lemma follows from Definition 16(b) and the fact that (7) is SN whenever (6) is and the hypotheses of the lemma are satisfied.

**Case 2:** $A(x) = *_{x_1M_1 \ldots M_m}$ and is not a $\Gamma_0$-simple generalized context. The lemma holds by Definition 12(b) and 16(c).

**Case 3:** $A(x) = *_{(\forall x : B(x))C(x, x)}$, where $B(x)$ is not a context. By hypothesis, (6) is in $\mathcal{C}[A(x)](p[A])$. Let

$\mathcal{D}$

$P : B(A)$

be any deduction in $\mathcal{C}[B(x)](p[A])$. Then by Definition 16(d) we have

$\begin{align*}
\mathcal{D}_3 \\
N : C \\
\frac{N : E}{\mathcal{D}_1(N)} \\
[N/x]M : [N/x]B \\
\{\mathcal{D}_1', \ldots, \mathcal{D}_n', \mathcal{D}\}
\end{align*}$
is in $\mathbb{C}[C(P,x)](p[A])$. By the induction hypothesis,

\[
\begin{align*}
\frac{\lambda x : E \cdot M : (\forall x : E)B (\forall \chi_i - 1)}{\lambda x : E \cdot M : (\forall x : C)B (\text{Eq''})} & \quad \frac{\lambda x : E \cdot M : (\forall x : C)B (\text{Eq''})}{N : C (\forall e)} \\
(\lambda x : E \cdot M) N : [N/x]B & \ \\
\{ \mathcal{D}_1 ', \ldots , \mathcal{D}_n ', \mathcal{D} \}
\end{align*}
\]

is in $\mathbb{C}[C(P,x)](p[A])$. Hence, by Definition 16(d), (7) is in $\mathbb{C}[A(x)](p[A])$.

Case 4: $A(x) =^* (\forall x : B(x))C(x,x)$, where $B(x)$ is a context. By hypothesis, (6) is in $\mathbb{C}[A(x)](p[A])$. Let

$\mathcal{D}$

$F : B(A)$

be any deduction in $\mathbb{C}[B(x)](p[A])$, and let $p[F]$ be a computability function for $F$. Then by Definition 16(e) we have

\[
\begin{align*}
\frac{\mathcal{D}_3}{N : C (\text{Eq''})} & \quad \frac{\mathcal{D}_1 (N)}{N : E} \\
[N/x]M : [N/x]B & \ \\
\{ \mathcal{D}_1 ', \ldots , \mathcal{D}_n ', \mathcal{D} \}
\end{align*}
\]

is in $\mathbb{C}[C(x,x)](p[F], p[A])$. By the induction hypothesis,

\[
\begin{align*}
\frac{\lambda x : E \cdot M : (\forall x : E)B (\forall \chi_i - 1)}{\lambda x : E \cdot M : (\forall x : C)B (\text{Eq''})} & \quad \frac{\lambda x : E \cdot M : (\forall x : C)B (\text{Eq''})}{N : C (\forall e)} \\
(\lambda x : E \cdot M) N : [N/x]B & \ \\
\{ \mathcal{D}_1 ', \ldots , \mathcal{D}_n ', \mathcal{D} \}
\end{align*}
\]

is in $\mathbb{C}[C(x,x)](p[F], p[A])$. Hence, by Definition 16(e), (7) is in $\mathbb{C}[A(x)](p[A])$. □

Lemma 6. If $A(x)$ and $p[A]$ satisfy the hypothesis of Definition 16, then $\mathbb{C}[A(x)](p[A])$ is a ground for each term $A(A)$.

Proof. Lemmas 4 and 5. □
The following lemma makes sense because of Lemma 6.

**Lemma 7.** Let \( x \) be a variable which is not assigned a supercontext as a type by \( \Gamma_0 \), let \( A(x,y) \) be any \( \Gamma_0 \)-type, and let \( B(y) \) be a term which can be shown from \( \Gamma_0 \) to have the same type as \( x \), where \( y \) includes all variables except \( x \) which occur free and which are not assigned supercontexts as types by \( \Gamma_0 \). Let \( C \) be a sequence of terms of the same types as the variables in \( y \) and let \( p[C] \) be an assignment of computability predicates to the terms in \( C \). Then

\[
\mathcal{C}[A(x,y)][\mathcal{C}[B(C)](p[C]), p[C]) = \mathcal{C}[A(B(y),y)](p[C]).
\]

**Proof.** By induction first on the rank of the type of \( B(y) \) and second on the structure of \( A(x,y) \). For simplicity, let \( \mathcal{p}[B(C)] \) abbreviate \( \mathcal{C}[B(y)](p[C]) \). (This is a computability predicate by Lemma 6.)

*Case 1:* \( A(x,y) \) is a \( \Gamma_0 \)-proof. Then both sides are \( \mathcal{A}(B(C), C) \) by Definition 16(a).

In the remaining cases, we may assume that \( A(x,y) \) is not a \( \Gamma_0 \)-proof.

*Case 2:* \( x \) does not occur free in \( A(x,y) \). Then the lemma is trivial. This takes care of the cases in which \( A(x,y) \) converts to \( \text{Prop} \) or \( \text{Type} \).

*Case 3:* \( A(x,y) =* zM_1\ldots M_n \), a simple generalized context. Then \( z \) is assigned a supercontext as a type by \( \Gamma_0 \) and hence, by hypothesis, is distinct from \( x \). Then by Definition 16(b), each side consists of the set of all SN deductions of type \( \mathcal{A}(B(C), C) \).

*Case 4:* \( A(x,y) =* yM_1(x,y)\ldots M_n(x,y) \), where \( y \neq x \) is one of the variables in \( y \), and \( C \) is the term in \( C \) corresponding to \( y \). Then

\[
\mathcal{C}[A(x,y)](p[B(C)], p[C]) = p[C](\mathcal{C}[M_1(x,y)](p[B(C)], p[C]), \ldots, \mathcal{C}[M_n(x,y)](p[B(C)], p[C])),
\]

and since \( \mathcal{A}(B(y),y) =* yM_1(B(y),y)\ldots M_n(B(y),y) \),

\[
\mathcal{C}[A(B(y),y)](p[C]) = (p[C])(\mathcal{C}[M_1(B(y),y)](p[C]), \ldots, \mathcal{C}[M_n(B(y),y)](p[C])).
\]

The lemma follows by the induction hypothesis.

*Case 5:* \( A(x,y) =* xM_1(x,y)\ldots M_p(x,y) \). For simplicity, write this as \( xM(x,y) \). Then the type of \( x \) and \( B(y) \) is

\[
(\forall z_1 : E_1)\cdots(\forall z_p : E_p)G,
\]

where \( G \) is either \( \text{Prop} \) or a \( \Gamma_0 \)-simple context function, and so \( B(y) \) is a proposition function. By Definition 16(c),

\[
\mathcal{C}[A(x,y)](p[B(C)], p[C]) = p[B(C)](\mathcal{C}[M(x,y)](p[B(C)], p[C])).
\]

By the induction hypothesis, the right-hand side equals

\[
p[C](\mathcal{C}[M(C,y)](p[C])),
\]
which, by our abbreviation for \( p[B(C)] \), is
\[
C[B(y)][p[C]](C[M(B(y),y)][p[C]]).
\]
If \( p = 0 \), we are finished, since \( A(B(y),y) =* B(y) \) and \( M(B(y)) \) is void, so this is just
\[
C[A(B(y),y)][p[C]],
\]
as desired. If \( p > 0 \), then we have the following subcases according to Corollary 10.1:

**Subcase 5.1:** \( B(y) = \lambda z_1 : E_1 \ldots \lambda z_p : E_p \cdot F(z,y) \), where \( z \) is the sequence \( z_1, \ldots, z_p \). By Definition 16(f),
\[
C[B(y)][p[C]](C[M_B(y),y])(p[C])
\]
is
\[
C[B(y)][p[C], C[M_B(y),y])(p[C])].
\]
By the induction hypothesis on the type of \( B(y) \), this is
\[
C[B(y)M(B(y),y)][p[C]],
\]
and since \( A(B(y),y) =* B(y)M(B(y),y) \), we are done.

**Subcase 5.2:** \( B(y) = y;N(y) \), which we may as well abbreviate as \( y;N(y) \). Then \( A(R(y),y) =* y;N(y)M(B(y),y) \). Now by Definition 16(c),
\[
C[B(y)][p[C]](C[M_B(y),y])(p[C])
\]
is
\[
p[C_i][C[N(y)][p[C]](C[M_B(y),y])(p[C])],
\]
but this is the same thing as
\[
p[C_i][C[N(y)][p[C], C[M_B(y),y])(p[C])],
\]
and by Definition 16(c), this is
\[
C[A(B(y),y)][p[C]],
\]
as desired.

**Case 6:** \( A(x,y) =* (\forall z : E(x,y))F(z,x,y) \), where \( E(x,y) \) is not a context. By the induction hypothesis,
\[
C[E(x,y)][p[B(C)],p[C]] = C[E(B(y),y)][p[C]]
\]
and, for any term \( N(y) \) such that there is a \( \Gamma_0 \)-acceptable deduction ending in \( N(C) \): \( E(B(C)) \),
\[
C[F(z,x,y)][p[B(C)],p[C]] = C[F(z,B(y),y)][p[C]].
\]
By Definition 16(d), the lemma follows.
Case 7: $A(x, y) =* (\forall z : E(x, y))F(z, x, y)$, where $E(x, y)$ is a context. Similar to Case 4 using Definition 16(e). □

Notation. In the following lemma, $x$ will denote the sequence $x_1, \ldots, x_n$, $y$ the sequence $y_1, \ldots, y_m$, $N$ the sequence $N_1, \ldots, N_n$, $B$ the sequence $B_1, \ldots, B_m$, and $p[B]$ the sequence $p[B_1], \ldots, p[B_m]$. Furthermore, $A_{i+1}'$, for $i = 0, 1, \ldots, n - 1$, will denote $[N_1/x_1, \ldots, N_i/x_i]A_{i+1}$.

**Lemma 8.** Let

$x_1 : A_1(y), \ldots, x_n : A_n(y)$

$\mathcal{D}(x, y)$

$M(x, y) : A(x, y)$

be a $\Gamma_0$-acceptable deduction all of whose undischarged assumptions are among those shown, where $y$ consists of all variables which occur free in any type or term which are not assigned supercontexts as types by $\Gamma_0$. For all assignments of terms $B_1, \ldots, B_m$ to $y_1, \ldots, y_m$ (where for each $i = 1, 2, \ldots, m$, it can be proved from $\Gamma_0$ that $B_i$ is in the type assigned to $y_i$) and for all assignments of computability predicates $p[B_1], \ldots, p[B_m]$ to $B_1, \ldots, B_m$, if for $i = 1, 2, \ldots, n$, the $\Gamma_0$-acceptable deduction

$\mathcal{D}_i$

$N_i : A_i'(B)$

is in $\mathcal{C}[A_i(y)](p[B])$, then

$\mathcal{D}_1$ \hspace{1cm} $\mathcal{D}_n$

$N_1 : A_1'(B), \ldots, N_n : A_n'(B)$

$\mathcal{D}(N, B)$

$M(N, B) : A(N, B)$

is in $\mathcal{C}[A(N, y)](p[B])$.

**Remark 12.** Cf. [18, Theorem A2.3, Lemma 3(b)].

**Proof.** By induction on structure of $\mathcal{D}(x, y)$.

**Basis:**

Case 1: $\mathcal{D}(x, y)$ consists of the axiom (PT). Since this deduction is clearly SN, the lemma follows by Definition 16(b).

Case 2: $\mathcal{D}(x, y)$ consists of the assumption $x_i : A_i(y)$. The lemma is immediate.

**Induction step:** There are the following cases, according to the last inference in $\mathcal{D}(x, y)$.

Case 1: The last inference is by ($\kappa'F$). By Definition 16(b), it is sufficient to prove that (8) is SN. By the induction hypothesis and Definition 16(b), the deductions of both premises are SN. Hence, (8) is SN.
Case 2: The last inference is by \((Eq')\). Similar to Case 1.

Case 3: The last inference is by \((\forall e)\). Then \(M(x,y) \equiv M_1(x,y)M_2(x,y)\), \(A(x,y) \equiv E(M_2(x,y), x, y)\), and \(\mathcal{D}(x,y)\) is

\[
\begin{align*}
&x_1 : A_1(y), \ldots, x_n : A_n(y) \\
&\mathcal{D}'(x,y) \\
&\mathcal{D}''(x,y) \\
&M_1(x,y) : (\forall x : C(x,y))E(x,x,y) \\
&M_2(x,y) : C(x,y) \\
&M_1(x,y)M_2(x,y) : E(M_2(x,y), x, y).
\end{align*}
\]

Subcase 3.1: \(C(x,y)\) is not a context. By the induction hypothesis,

\[
\begin{align*}
&\mathcal{D}_1 \quad \mathcal{D}_n \\
&N_1 : A'_1(B), \ldots, N_n : A'_n(B) \\
&\mathcal{D}'(N,B) \\
&M_1(N,B) : (\forall x : C(N,B))E(x,N,B)
\end{align*}
\]

is in \(C[(\forall x : C(N,y))E(x,N,y)](p[B])\) and

\[
\begin{align*}
&\mathcal{D}_1 \quad \mathcal{D}_n \\
&N_1 : A'_1(B), \ldots, N_n : A'_n(B) \\
&\mathcal{D}''(N,B) \\
&M_2(N,B) : (N,B)
\end{align*}
\]

is in \(C[C(N,y)](p[B])\). Then by Definition 16(d), (8) is in

\[
C[E(M_2(N,y), N,y)](p[B]).
\]

Subcase 3.2: \(C(x,y)\) is a context. By the induction hypothesis,

\[
\begin{align*}
&\mathcal{D}_1 \quad \mathcal{D}_n \\
&N_1 : A'_1(B), \ldots, N_n : A'_n(B) \\
&\mathcal{D}'(N,B) \\
&M_1(N,B) : (\forall x : C(N,B))E(x,N,B)
\end{align*}
\]

is in \(C[(\forall x : C(N,y))E(x,N,y)](p[B])\) and

\[
\begin{align*}
&\mathcal{D}_1 \quad \mathcal{D}_n \\
&N_1 : A'_1(B), \ldots, N_n : A'_n(B) \\
&\mathcal{D}''(N,B) \\
&M_2(N,B) : (N,B)
\end{align*}
\]

is in \(C[C(N,y)](p[B])\). Then by Definition 16(e), for any computability predicate \(p[M_2(N,B)]\), (8) is in \(C[E(x,N,y)](p[M_2(N,B)], p[B])\). To complete the proof, it is sufficient to find a computability predicate \(p[M_2(N,y)]\) such that

\[
(9) \quad C[E(x,N,y)](p[M_2(N,B)], p[B]) = C[E(M_2(N,y), N,y)](p[B]).
\]
A suitable such function is the one such that

\[ p[M_2(N, B)] = C[M_2(N, y)][p(B)]. \]

That this is a computability predicate follows from Definition 15 and Lemma 6. That (9) holds follows from Lemma 7.

**Case 4:** The last inference is by \((\forall \kappa i)\). Then

\[ A(x, y) \equiv (\forall x : C(x, y))E(x, x, y), \]

\[ M(x, y) = \lambda x : C(x, y). M_1(x, x, y), \]

and \(\varnothing(x, y)\) is

\[
\begin{array}{c}
1 \\
[x : C(x, y)], x_1 : A_1(y), \ldots, x_n : A_n(y) & x_1 : A_1(y), \ldots, x_n : A_n(y) \\
\varnothing'(x, y, y) & \varnothing''(x, y) \\
M_1(x, x, y) : E(x, x, y) & C(x, y) : \kappa \\
\lambda x : C(x, y). M_1(x, x, y) : (\forall x : C(x, y))E(x, x, y) & (\forall \kappa i - 1)
\end{array}
\]

**Subcase 4.1:** \(C(x, y)\) is not a context. Then \(\kappa \equiv \text{Prop}\). By the induction hypothesis, for all deductions

\[ \varnothing'' \]

\[ P : C(N, B) \]

in \(C[C(N, y)][p(B)]\). Hence, by Lemmas 4(b) and 5,

\[
\begin{array}{c}
\varnothing'' \\
\varnothing' \\
\varnothing \\
\varnothing_1 \quad \varnothing_n \\
[\lambda x : C^*, N_1 : A_1^*, \ldots, N_n : A_n^*] \quad N_1 : A_1^*, \ldots, N_n : A_n^* \\
\varnothing^*(x) \quad \varnothing'^*(x) \\
M_1^*(x) : E^*(x) \quad C^* : \kappa \\
\lambda x : C^*. M_1^*(x) : (\forall x : C^*)E^*(x) \quad P : C^* \\
(\forall \kappa i - 1) \quad (\forall e)
\end{array}
\]

where \(A_i^* \equiv A_i'(B)\), \(X^* \equiv X(N, B)\), and \(X^*(Y) \equiv X(Y, N, B)\), is also in \(C[C(E(P, N, y)][p(B)]\). Since \(\varnothing''\) is arbitrary, this implies by Definition 16(e) that (8) is in \(C[C(A(N, y)][p(B)]\).

**Subcase 4.2:** \(C(x, y)\) is a context. Then \(\kappa \equiv \text{Type}\). By the induction hypothesis, for all deductions

\[ \varnothing'' \]

\[ F : C(N, B) \]
in $C[C(N,y)](p[B])$ and for all computability predicates $p[F]$,

$$
\frac{\mathcal{D}_1 \ldots \mathcal{D}_n}{\mathcal{D}'''}
\frac{F : C(N,B), N_1 : A'_1(B), \ldots, A'_n(B)}{M_1(F,N,B) : E(F,N,B)}
\frac{\mathcal{D}'(F,N,B)}{M_1(F,N,B) : E(F,N,B)}
\frac{\mathcal{D}''(x)}{M_1^*(x) : E^*(x)}
\frac{C^* : \lambda (\forall \kappa_1 - 1) \mathcal{D}'''(\forall e)}{\mathcal{D}'''}
\frac{\lambda x : C^* \cdot M_1^*(x) : (\forall x : C^*)E^*(x)}{F : C^*}
$$

is in $C[E(x,N,y)](p[F],p[B])$. Hence, by Lemmas 4(b) and 5,

$$
\frac{[x : C^*], N_1 : A_1^*, \ldots, N_n : A_n^*}{\mathcal{D}_1 \ldots \mathcal{D}_n}
\frac{N_1 : A_1^*, \ldots, N_n : A_n^*}{\mathcal{D}_1 \ldots \mathcal{D}_n}
\frac{\mathcal{D}'(x)}{M_1^*(x) : E^*(x)}
\frac{C^* : \lambda (\forall \kappa_1 - 1) \mathcal{D}'''(\forall e)}{\mathcal{D}'''}
\frac{\lambda x : C^* \cdot M_1^*(x) : E^*(x)}{F : C^*}
$$

where $A_i^*$, $X^*$, and $X^*(Y)$ are as in Subcase 4.1, is also in

$C[E(x,N,y)](p[F],p[B])$.

Since $\mathcal{D}'''$ and $p[F]$ are arbitrary, this implies by Definition 16(d) that (8) is in $C[A(N,y)](p[B])$.

Case 5: The last inference is by (Eq''). This is straightforward by Definition 16. □

**Theorem 11.** Every deduction in TOC0 is strongly normal.

**Proof.** In Lemma 8, let $\mathcal{D}_i$ consist of the assumption $x_i : A_i(y)$ and let $B_j$ be $y_j$. Then for any sequence $p[B]$, $\mathcal{D}(x,y)$ is in $C[A(x,y)](p[B])$, and so is SN. □

4. Basic consequences of the strong normalization theorem

Although we have proved the strong normalization theorem for deductions, this theorem is usually proved for terms. In [18, Corollary 15.21.1] it is shown that for ordinary type assignment to $\lambda$-terms, the normalization theorem for terms can be proved from the strong normalization theorem for deductions by using the subject-construction theorem, which says that a deduction follows the construction of the term to which the type is assigned. (Actually, the reference in the proof of [18, Corollary 15.21.1] is to the proof of the subject-reduction theorem, but the property of that proof which is actually used is the subject-construction theorem.) We do not have this theorem for TOC0 in a form that is easy to state. Nevertheless, there is a relationship between terms and deductions, and we can expect to use this relationship to obtain a normalization theorem for terms.
Theorem 12. If $\Gamma$ is a well-formed environment and if

$$\Gamma \vdash M : A,$$

then $M$ has a normal form.

Proof. By Theorem 11 there is a normal deduction $\mathcal{D}$ of

$$\Gamma \vdash N : A,$$

where $M \triangleright N$. The proof is by induction on the deduction $\mathcal{D}$.

Basis: If $\mathcal{D}$ consists of an assumption, then $N$ is a variable, and so it is in normal form. If $\mathcal{D}$ consists of the axiom (PT), then $N$ is $\text{Prop}$, which is in normal form.

Induction step: There are the following cases, depending on the last inference in $\mathcal{D}$.

Case 1: The last inference is by rule $(\kappa \kappa' \mathcal{F})$. Then $A$ is $\kappa'$, $N$ is $(\forall x : B)C$, and $\mathcal{D}$ is

\[
\begin{array}{c}
1 \\
\mathcal{D}_1 \\
\mathcal{D}_2(x) \\
B : \kappa \\
C : \kappa' \\
(\forall x : B)C : \kappa'.
\end{array}
\]

By the induction hypothesis, $B$ and $C$ have normal forms; hence, so does $A$.

Case 2: The last inference is by rule $(\kappa \ell' \kappa)$. Then by the induction hypothesis, $N$ converts to a term $B$ (to the left of the colon in the premise) which has a normal form.

Case 3: The last inference is by rule $(\forall e)$. Then $N \equiv PQ$, $A \equiv [Q/x]C$, and $\mathcal{D}$ is

\[
\begin{array}{c}
\mathcal{D}_1 \\
P : (\forall x : B)C \\
\mathcal{D}_2 \\
Q : B \\
PQ : [Q/x]C.
\end{array}
\]

By the induction hypothesis, $P$ and $Q$ have normal forms. Furthermore, since $\mathcal{D}$ is normal, there is no Deduction-reduction possible in it. It follows that at the top of the left branch of $\mathcal{D}$ (and hence of $\mathcal{D}_1$) is an undischarged assumption. It follows that $P = * yQ_1 \ldots Q_m$ for some variable $y$. It follows that $Q_1, \ldots, Q_m$ all have normal forms, and hence that $PQ = * yQ_1 \ldots Q_m$ does as well.

Case 4: The last inference is by rule $(\forall \kappa i)$. Then $A \equiv (\forall x : B)C$, $N \equiv \lambda x : B \cdot P$, and $\mathcal{D}$ is

\[
\begin{array}{c}
1 \\
\mathcal{D}_1(x) \\
\mathcal{D}_2 \\
P : C \\
\lambda x : B \cdot P, (\forall x : B)C.
\end{array}
\]

(\forall \kappa i - 1)
By the induction hypothesis, $B$ and $P$ have normal forms; hence, so does $N = \lambda x : B.P$.

Case 5: The last inference is by rule $(\text{Eq}''')$. Then $N$ is the term to the left of the colon in the premise, and so by the induction hypothesis it has a normal form. □

Note that we have not proved that every term is SN. If we try to replace the conclusion by "$N$ is SN" in the above proof, we can see that Case 2 breaks down, since not every term convertible to an SN term is itself SN. Indeed, if $A$ is SN, and if $x \not\in FV(A)$, then for any terms $B$ and $C$, $(\lambda x : B.A)C = * A$; now if $C$ has no normal form, then $(\lambda x : B.A)C$ is not SN. This shows that we cannot strengthen the theorem to prove that $N$ is SN. (Of course, to prove that $M$ is SN is somewhat more complicated; we will take this up below.)

It might appear that since only Case 2 breaks down, and since the conclusion in this case is not a proof, we might want to add the assumption that $N : A$ is a proof. This will exclude Case 2. But now we have trouble with Case 4: we can conclude that $P$ is SN, but not that $B$ is SN. Indeed, by the remarks of the previous paragraph, $B$ might not be SN.

Mitchell [30] defines a function $\text{Erase}$ for the second order polymorphic typed $\lambda$-calculus which deletes the types of the bound variables. When this function is modified for TOC0, it is defined as follows.

**Definition 17.** (a) $\text{Erase}(a) \equiv a$ if $a$ is a constant or a variable;
(b) $\text{Erase}(MN) \equiv \text{Erase}(M)\text{Erase}(N)$;
(c) $\text{Erase}(\lambda x : A.M) \equiv \lambda x .\text{Erase}(M)$; and
(d) $\text{Erase}((\forall x : A)B) \equiv (\forall x : \text{Erase}(A))\text{Erase}(B)$.

Note that except for clause (d), we are mapping terms of TOC0 to pure $\lambda$-terms. In fact, the range of the function $\text{Erase}$ is the set of TAG terms [37, Definition 2.18], or, equivalently, the set of terms assigned types by TAG$_A$ [18, Definition 16.31].

We can now prove that if $A$ is neither a context nor a supercontext in the theorem, then $\text{Erase}(N)$ is SN. To extend this result to $\text{Erase}(M)$, it is enough to note that deductions of proofs do follow the constructions of the terms except that additional inferences of formulas which are not proofs are added at various places on top. This will give us the following result.

**Corollary 12.1.** Under the hypotheses of Theorem 12, if $A$ is neither a context nor a supercontext, then $\text{Erase}(M)$ is strongly normal.

There are some further corollaries that follow immediately from Theorem 12. These corollaries are standard consequences of normalization theorems.

**Corollary 12.2.** For terms $M$ and $N$ such that

$$\Gamma \vdash M : A,$$
and
\[ \Gamma \vdash N : A, \]

where \( \Gamma \) is a well-formed environment, it is decidable whether or not \( M =_* N \).

**Corollary 12.3.** For a term \( M \) and a well-formed environment \( \Gamma \), it is decidable whether or not there is a term \( A \) such that
\[ \Gamma \vdash M : A. \]

We can also relate TOC0 to the second-order polymorphic typed \( \lambda \)-calculus. Let us take the latter in the form of the system TAP of [18, Definition 16.81, except that to say that a term \( M \) is in type \( \alpha \), I shall continue to write \( M : \alpha \). Interpret terms of TAP as terms of TOC0 as follows: first, divide the variables of TOC0 into two mutually disjoint classes, the first for interpreting term variables of TAP and the second for interpreting the type variables. Then, for a term or type \( A \) of TAP, we define \( A^* \), a term of TOC0, as follows:

(a) if \( x \) is a term variable, then \( x^* \) is a variable of the first class chosen so that if \( x \) and \( y \) are distinct, so are \( x^* \) and \( y^* \);

(b) if \( a \) is a type variable, then \( a^* \) is a variable of the second class chosen so that if \( a \) and \( b \) are distinct, so are \( a^* \) and \( b^* \);

(b') \( (\alpha \rightarrow \beta)^* \) is \( (\forall x : x^*)\beta^* \) for a (term-) variable \( x \) which does not occur free in \( \alpha^* \) or \( \beta^* \);

(c) \( (\Delta a \cdot \alpha)^* \) is \( (\forall a^* : \text{Prop})\alpha^* \);

(d) \( (MN)^* \) is \( M^*N^* \);

(e) \( (M\alpha)^* \) is \( M^*\alpha^* \);

(f) \( (\lambda x : \alpha . M)^* \) is \( \lambda a^* : \alpha^* . M^* \); and

(g) \( (\Lambda a . M)^* \) is \( \lambda a^* : \text{Prop} . M^* \).

It is easy to show that if \( \alpha \) is any type scheme of TAP, then \( \alpha^* \) is in normal form, and that if \( M \) is any term of TAP which is in normal form, then \( M^* \) is also in normal form. Note also that this interpretation takes any \( \beta^2 \)-contraction of TAP into a \( \beta \)-contraction of TOC0.

It is easy to show that TAP can be interpreted in TOC0 using this mapping; see [18, Theorem 16.66]. (The result is proved for a different system, TAGL, but that system is close enough to TOC0 that the proof can be easily adapted.) For the converse, we have the following theorem.

**Theorem 13.** Let \( \Gamma \) be a sequence
\[ x_1 : \alpha_1, x_2 : \alpha_2, \ldots, x_n : \alpha_n \]
of assumptions in TAP, and let \( \Gamma^* \) be
\[ x_1^* : \alpha_1^*, x_2^* : \alpha_2^*, \ldots, x_n^* : \alpha_n^* \]
Let $\alpha$ be any type scheme in TAP, let $a_1, \ldots, a_m$ include all of the type variables which occur free in $\alpha$, and let $\Gamma'$ be

$$a_1^* : \text{Prop}, \ldots, a_m^* : \text{Prop}.$$

If $\mathcal{D}$ is a normal deduction in TOC0 of

$$\Gamma'^*, \Gamma' : M^* : \alpha^*,$$

where $M$ is a term of TAP, then there is a normal deduction $\mathcal{D}'$ in TAP of

$$\Gamma : M : \alpha.$$

**Remark.** Note that this theorem does not say that TOC0 is no stronger than TAP. It only says that any result of TOC0 which consists of translations of TAP formulas could have been proved in TAP. Functions that cannot be defined in TAP can be defined in TOC0, but to do so requires formulas which are not translations of TAP formulas.

**Proof.** Note first that [18, Lemmas 16.67 and 16.68] hold for TOC0 as well as for TAGL; the proofs for TOC0 are obtained by a minor change in notation from those for TAGL.

The proof is by induction on the deduction $\mathcal{D}$. Note that by hypothesis, $\mathcal{D}$ does not consist of axiom (PT), and its last inference is not by any of rules (κκ'F) or (Eqκ). Furthermore, the last inference is not by rule (Eq''); for the types of the assumptions (both discharged and undischarged) and of the conclusion are all in normal form, and if the types of the premises of any rule except (∀e) and (Eq'') are in normal form, then so is the type of the conclusion. With regard to inferences in $\mathcal{D}$ by rule (∀e) the left branch above each such inference contains inferences only by the same rule and rule (Eq'') and at the top of the branch is an assumption (since $\mathcal{D}$ is normal); and it is not hard to see by beginning with the assumption that because the type of the left premise of each such inference by rule (∀e) is $\beta^*$ for some TAP type scheme $\beta$, so is the type of the conclusion. It follows that each of these types is in normal form, and so there is no inference by rule (Eq'') in the branch. There are the following remaining cases:

**Case 1:** $\mathcal{D}$ consists of an assumption. Then $M$ is $\alpha$, $\alpha$ is $\alpha$, and $\mathcal{D}'$ consists of the corresponding assumption in TAP.

**Case 2:** The last inference in $\mathcal{D}$ is by rule (∀e). Then since $\mathcal{D}$ is normal, the only inferences which occur in the left branch are by rules (∀e). Furthermore, $M^*$ is in normal form. Now it follows from this that $M^*$ has the form $xM_1 \ldots M_p$, where $x$ is assigned a type by the assumption at the top of the branch (which is not discharged). Hence, $x$ is one of the $x_i^*$. By the definition of the interpretation, it follows that each $M_i$ is either $N_j^*$ for some TAP term $N_j$, in which case the type assigned to it is $\gamma_j^*$ for some TAP type scheme $\gamma_j$, or else some $\beta_j^*$ for some TAP type scheme $\beta_j$, in which case the type assigned to it is Prop. By the induction hypothesis, there is a normal
deduction \( \mathcal{D}_j \) of \( \Gamma \vdash N_j : \gamma_j \) for each such \( N_j \), and then rules (\( \rightarrow e \)) and (\( \forall e \)) of TAP can be used to obtain \( \mathcal{D}' \) from the assumption \( x_i : \alpha_i \) and the deductions \( \mathcal{D}_j \).

Case 3: The last inference in \( \mathcal{D} \) is by rule (\( \forall Pi \)). Then \( \alpha^* \) is (\( \forall x : B \))\( C \) and \( M^* \) is \( \lambda x : B. \, N \). By the right premise, \( B \) is \( \beta^* \) for some TAP type scheme \( \beta \), and it follows that \( x \) is some \( y^* \), for a TAP term variable \( y \), and does not occur free in \( C \); furthermore, \( C \) is \( \gamma^* \) for some TAP type scheme \( \gamma \). In addition, \( N \) is \( P^* \) for some TAP term \( P \). It follows that if the last inference is removed from \( \mathcal{D} \), the result is a normal deduction \( \mathcal{D}_1 \) of

\[
\Gamma^*, y^* : \beta^*, \Gamma' \vdash P^* : \gamma^*.
\]

By the induction hypothesis, there is a normal deduction \( \mathcal{D}_1' \) of

\[
\Gamma, y : \beta \vdash P : \gamma
\]

in TAP, and \( \mathcal{D}' \) is obtained by an inference by rule (\( \rightarrow i \)).

Case 4: The last inference in \( \mathcal{D} \) is by rule (\( \forall Ti \)). Then \( \alpha^* \) is (\( \forall x : B \))\( C \) and \( M^* \) is \( \lambda x : B. \, N \). By the right premise, \( B \) is Prop. Hence, \( x \) is \( \beta^* \) for a TAP type variable \( a \), \( C \) is \( \beta^* \) for some TAP type scheme \( \beta \), and \( N \) is \( P^* \) for some TAP term \( P \). It follows that if the last inference is removed from \( \mathcal{D} \), the result is a normal deduction \( \mathcal{D}_1 \) of

\[
\Gamma^*, \Gamma', a^* : \text{Prop} \vdash P^* : \beta^*.
\]

By the induction hypothesis, there is a normal deduction \( \mathcal{D}_1' \) of

\[
\Gamma \vdash P : \beta
\]

in TAP. Since \( \alpha \) is \( \Delta a . \beta \), \( \mathcal{D}' \) follows by an inference by rule (\( \forall i \)). \( \square \)

**Corollary 13.1.** Under the hypotheses of the theorem, if \( N =_* M^* \), if \( A =_* \alpha^* \), and if

\[
\Gamma^*, \Gamma' \vdash N : A,
\]

in TOC0, then

\[
\Gamma \vdash M : \alpha
\]

in TAP.

5. Other formulations of the calculus

In this section we shall consider an alternative formulation of the theory of constructions. It is a variant of the form in which the theory was originally presented in Coquand [5], and is closer to the presentation in other papers by Coquand and Huet than is the system TOC0.
As we saw in a previous section, every rule which discharges an assumption of the form \( x : A \) has a premise not depending on this discharged assumption that is either \( A : \text{Prop} \) or \( A : \text{Type} \). If we wanted to, we could take these premises as justifications for the assumptions instead of premises for the rules; this is the approach adopted by Martin-Löf in his work (see his [27–29]). The main reason this is not done in TOC0 is that it would require that premise to be written above the assumption, and then the assumptions would not occur at the tops of branches, an inconvenience for the theory of a system such as TOC0. But for the form of the theory of constructions presented by Coquand, it is the most useful approach.

This form of the theory of constructions is what is known as a sequent calculus. A sequent is an expression of the form

\[
\Gamma \vdash E,
\]

where \( \Gamma \) is a (possibly empty) sequence of formulas and \( E \) is a formula. This particular sequent calculus is formulated in such a way that the only nonempty sequences that can occur to the left of the turnstile (the symbol ‘\( \vdash \)’) are well-formed environments. This will make unnecessary the premises which “justify” the discharged assumptions; for these assumptions will all occur to the left of the turnstile in the premises of the rules and will hence be part of well-formed environments, and so these premises will automatically hold. The fact that \( \Gamma \) is a well-formed environment will be equivalent to the derivability of the sequent

\[
\Gamma \vdash \text{Prop} : \text{Type}.
\]

The system will be called TOC2, following [32].

Note that until the equivalence of TOC0 and TOC2 is proved, it will be necessary to specify the system with respect to which an environment is well-formed. Until notice to the contrary is given, a well-formed environment will mean with respect to TOC2.

**Definition 18.** The system TOC2 is a sequent calculus; its sequents are of the form

\[
\Gamma \vdash E,
\]

where \( \Gamma \) is a sequence of TOC0 formulas and \( E \) is a TOC0 formula. The formulation has one axiom, namely

\[
(\text{PT}) \quad \vdash \text{Prop} : \text{Type}
\]

(which says that the empty type assumption is valid). The rules of the system are as follows.

**Validity 1.** If \( x \) does not occur free in \( A \) or \( \Gamma \),

\[
\Gamma \vdash A : \kappa
\]

\[
\Gamma, x : A \vdash \text{Prop} : \text{Type},
\]
Validity 2. If \( A \) is a supercontext and if \( x \) does not occur free in \( A \) or \( \Gamma \),
\[
\Gamma \vdash \text{Prop : Type}
\]
\[
\Gamma,x : A \vdash \text{Prop : Type},
\]

Variable.
\[
\Gamma,x : A, \Theta \vdash \text{Prop : Type}
\]
\[
\Gamma,x : A, \Theta \vdash x : A,
\]

Type formation. If \( x \) does not occur free in \( A \) or \( \Gamma \) and if \( A \) is not a supercontext,
\[
\Gamma,x : A \vdash B : \kappa
\]
\[
\Gamma \vdash (\forall x : A)B : \kappa,
\]

Abstraction. If \( x \) does not occur free in \( A \) or \( \Gamma \) and if \( A \) is not a supercontext,
\[
\Gamma,x : A \vdash M : B \quad \Gamma,x : A \vdash B : \kappa
\]
\[
\Gamma \vdash \lambda x : A . M : (\forall x : A)B,
\]

Application.
\[
\Gamma \vdash M : (\forall x : A)B \quad \Gamma \vdash N : A
\]
\[
\Gamma \vdash MN : [N/x]B.
\]

Conversion 1.
\[
\Gamma \vdash M : A \quad A =_{\ast} B
\]
\[
\Gamma \vdash M : B.
\]

Conversion 2.
\[
\Gamma \vdash A : \kappa \quad A =_{\ast} B
\]
\[
\Gamma \vdash B : \kappa
\]

Remark 13. TOC2 is similar to the presentation of Coquand and Huet [11], but differs in that it includes Validity 2 and its conversion rules are more general. To get a system equivalent to that of Coquand and Huet, it is necessary to delete Conversion 2 and Validity 2 and replace Conversion 1 by the rule
\[
\Gamma \vdash M : A \quad A =_{\ast} B \quad \Gamma \vdash B : \kappa
\]
\[
\Gamma \vdash M : B.
\]

For a discussion of the generalization of the conversion rules used here, see [38, Remark before Definition 1]. The need for Validity 2 results from our inclusion of assumptions whose types are supercontexts.

Remark 14. Pottinger [32] has also proposed replacing rules Validity 1, Validity 2, and Variable by the following three structural rules.

Hypothesis 1. If \( x \) does not occur free in \( A \) or \( \Gamma \),
\[
\Gamma \vdash A : \kappa
\]
\[
\Gamma,x : A \vdash x : A,
\]
Hypothesis 2. If $\Gamma$ is a well-formed environment, $x$ does not occur free in $A$ or $\Gamma$, and if $A$ is a supercontext,

$$\Gamma, x : A \vdash x : A,$$

Reiteration.

$$\Gamma \vdash E \quad \Gamma, F \vdash G$$

$$\Gamma, F \vdash E.$$

Pottinger calls this formulation TOC1. (These rules are sequent versions of the rules for Hypothesis and Reiteration of Fitch [16].) It is not difficult to prove that a sequent $\Gamma \vdash E$ is provable in TOC1 if and only if it is provable in TOC2 [32]. Note that in [32], both TOC1 and TOC2 have the more restricted conversion rules of Coquand and Huet, but the extension of his equivalence result to the versions given here poses no problem.

**Remark 15.** Pottinger's formulation of the rule Abstraction differs from that given here in that instead of the second premise he adds the proviso that $B$ not convert to $\text{Type}$. It turns out that the proviso is equivalent to the second premise given here (which also appears in [7]).

We shall now establish the equivalence of TOC2 and TOC0. For this proof, $\vdash_0$ will mean provable in TOC0 and $\vdash_2$ will mean provable in TOC2. We need some lemmas.

**Lemma 9.** If $\Gamma \vdash_2 E$ for any formula $E$, and if $\Gamma'$ is any initial segment of $\Gamma$ (possibly including $\Gamma$ itself), then each derivation of $\Gamma \vdash_2 E$ contains a subderivation of $\Gamma' \vdash_2 \text{Prop} : \text{Type}$.

**Proof.** By induction on the derivation of $\Gamma \vdash_2 E$.

**Basis:** If $\Gamma \vdash_2 E$ is the axiom (PT), then $\Gamma'$ is empty, and the result is trivial.

**Induction step:** We assume the property for each premise of a rule and prove it for the conclusion.

If the sequence to the left of $\vdash$ in the conclusion is an initial segment of that of at least one premise, this is trivial. This takes care of all rules except the validity rules. In these cases, $\Gamma$ is $\Gamma_1, A : \text{Prop}$, and $E$ is $\text{Prop} : \text{Type}$. If $\Gamma'$ is all of $\Gamma$, then the entire deduction is what we seek. Otherwise, $\Gamma'$ is an initial segment of $\Gamma_1$, and the result is trivial by the induction hypothesis. □

**Lemma 10.** If $\Gamma \vdash_2 \text{Prop} : \text{Type}$, then $\Gamma$ is a well-formed environment.

**Proof.** By induction on the pair $(n,m)$, where $n$ is the number of formulas in $\Gamma$ and $m$ is the length of the derivation of $\Gamma \vdash_2 \text{Prop} : \text{Type}$.

**Basis:** Trivial, since $\Gamma$ is empty.
**Induction step:** Assume the lemma for any initial subsequence of \( \Gamma \), and suppose that \( \Gamma \) is \( \Gamma', x : A \). By the induction hypothesis, \( \Gamma' \) is a well-formed environment. Now the only rules of which
\[
\Gamma', x : A \vdash \text{Prop} : \text{Type}
\]
can be the conclusion are the conversion and validity rules. If the rule is a conversion rule, then by Lemma 9 there is a subderivation of the derivation of the premise of the inference which is a derivation of
\[
\Gamma', x : A \vdash \text{Prop} : \text{Type}
\]
and so the conclusion follows by the induction hypothesis; if the rule is a validity rule, then it follows that \( x \) does not occur free in \( \Gamma' \) or in \( A \) and that if \( A \) is not a supercontext then
\[
\Gamma' \vdash A : \kappa.
\]
Since \( \Gamma' \) is a well-formed environment, this implies that \( \Gamma \) is as well. \( \Box \)

**Lemma 11.** If \( \Gamma \vdash E \), then \( \Gamma \) is a well-formed environment.

**Proof.** Lemmas 9 and 10. \( \Box \)

**Theorem 14.** There is a formula \( E \) such that \( \Gamma \vdash E \) if and only if \( \Gamma \) is a well-formed environment.

**Proof.** The “only if” part is Lemma 11. The “if” part is easy using the axiom and rules Validity.

We are now in a position to prove the equivalence between TOC0 and TOC2.

**Theorem 15.** If
\[
(12) \quad \Gamma \vdash E,
\]
then
\[
(13) \quad \Gamma \vdash_0 E.
\]

**Proof.** By induction on the derivation of (12).

- **Basis:** (12) is (PT) in TOC2. Then \( \Gamma \) is empty, \( E \) is \text{Prop} : \text{Type}, and (13) holds by axiom (PT) in TOC0.

- **Induction step:** The cases are by the last rule used in the derivation of (12).
  - **Case** Validity 1. Trivial.
  - **Case** Validity 2. Trivial.
Case Type formation. \( E \) is \( (\forall x : A)B : \kappa \), where \( x \) does not occur free in \( A \) or \( \Gamma \), and the premise is
\[
\Gamma, x : A \vdash B : \kappa.
\]
By the induction hypothesis,
\[
\Gamma, x : A \vdash_0 B : \kappa.
\]
Furthermore, by Theorem 14, \( \Gamma, x : A \) is a well-formed environment (with respect to TOC2), and \( A \) is not a supercontext. This means that the derivation of (12) includes a subderivation of
\[
\Gamma \vdash_2 A : \kappa'.
\]
Hence, again by the induction hypothesis,
\[
\Gamma \vdash_0 A : \kappa'.
\]
Hence, (13) follows by \((\kappa' \kappa F)\).

Case Variable. Trivial by the conventions of natural deduction systems.

Case Abstraction. Similar to Case Type formation, using \((\forall \kappa i)\).

Case Application. \( E \) is \( MN : [N/x]B \), and the premises are
\[
\Gamma \vdash_2 M : C \quad \text{and} \quad \Gamma \vdash_2 N : A,
\]
where \( C =_\ast (\forall x : A)B \). By the induction hypothesis
\[
\Gamma \vdash_0 M : C \quad \text{and} \quad \Gamma \vdash_0 N : A.
\]
(13) then follows by rules \((\text{Eq}''\prime)\) and \((\forall e)\).

Case Conversion 1. Trivial by rule \((\text{Eq}''\prime)\).

Case Conversion 2. Trivial by rule \((\text{Eq}'\kappa)\). \( \square \)

For the converse we have:

**Theorem 16.** If \( \Gamma \) is a well-formed environment, and if (13) holds, then (12) holds.

**Proof.** By induction on the sum of the length (number of formulas in) of the proof of (13) plus the subsidiary proofs that show that \( \Gamma \) is a well-formed environment. We have the following cases:

Case (PT): If (13) is \((\text{PT})\) in TOC0, then (12) follows by \((\text{PT})\) in TOC2.

Case Assumption. If (13) consists only of the assumption \( x : A \), then \( \Gamma \) consists only of \( x : A \). Since \( \Gamma \) is a well-formed environment, \( A \) is a supercontext or \( \vdash_0 A : \kappa \).

If \( A \) is a supercontext, we have (12) as follows:

\[
\begin{align*}
\Gamma \vdash_2 & \text{Prop : Type} \\
\Gamma \vdash_2 & \text{Prop : Type} \\
\text{Validity 2} \\
\text{Variable}
\end{align*}
\]
If $\Gamma \vdash_0 A : \kappa$, then by the induction hypothesis, $\Gamma \vdash_2 A : \kappa$, and we get (12) by Validity 1 and Variable.

The remaining cases are by the last rule in the deduction of (13).

Case ($\forall \kappa'$F). (13) is

\[ \Gamma \vdash_0 ((\forall x : A))B : \kappa', \]

where $x$ does not occur free in $A$ or in $\Gamma$. The premises are

\[ \Gamma \vdash_0 A : \kappa \quad \text{and} \quad \Gamma, x : A \vdash_0 B : \kappa'. \]

Hence, $\Gamma, x : A$ is a well-formed environment (with respect to TOC0), and so by the induction hypothesis

\[ \Gamma, x : A \vdash_2 B : \kappa'. \]

Hence, (12) follows by Validity.

Case ($\forall e$): (13) is

\[ \Gamma \vdash_0 MN : [N/x]B, \]

where the premises are

\[ \Gamma \vdash_0 M : (\forall x : A)B \quad \text{and} \quad \Gamma \vdash_0 N : A. \]

By the induction hypothesis,

\[ \Gamma \vdash_2 M : (\forall x : A)B \quad \text{and} \quad \Gamma \vdash_2 N : A. \]

Hence, (12) follows by rule Application.

Case ($\forall \kappa i$): (13) is

\[ \Gamma \vdash_0 \lambda x : A.M : (\forall x : A)B, \]

where the premises are

\[ \Gamma, x : A \vdash_0 M : B \quad \text{and} \quad \Gamma \vdash_0 A : \kappa, \]

where $x$ does not occur free in $A$ or in $\Gamma$. It follows that $\Gamma, x : A$ is a well-formed environment with respect to TOC0, and so by the induction hypothesis,

\[ \Gamma, x : A \vdash_2 M : B. \]

Hence, (12) follows by rule Abstraction.

Cases (Eq") and (Eq'k): Trivial by the corresponding rules in TOC2. □

Theorem 17. A necessary and sufficient condition that (12) hold is that $\Gamma$ be a well-formed environment (with respect to TOC0) and that (13) hold.
Proof. Theorems 15 and 16. □

Corollary 17.1. An environment $\Gamma$ is well-formed with respect to TOC0 if and only if it is well-formed with respect to TOC2.

For this reason, we shall no longer specify the system with respect to which an environment is well-formed.

6. Representing logic with equality

We now turn to a representation of logic with equality. The definitions are all standard for second- and higher-order logic, but since they will be used extensively in the rest of the paper it is worth giving them here for definiteness.

Definition 19. The term $F$ is defined as follows:

$$F \equiv \lambda u : \text{Prop}. \lambda v : \text{Prop}. (\forall x : u)v.$$

We use either $A \rightarrow B$ or $A \supset B$ as an abbreviation for $FAB$, depending on the context. In particular, we often use $A \supset B$ when both $A$ and $B$ are in Prop.

It is easy to show that $\rightarrow$ satisfies the rules ($\rightarrow e$)

$$M : A \rightarrow B \quad N : A \quad \frac{}{MN : B}$$

and ($\rightarrow i$)

$$[x : A] \left[ \begin{array}{c} M : B \end{array} \right] A : k \quad \frac{\lambda x : A. M : A \rightarrow B}{\lambda x : A. M : A \rightarrow B}.$$

This means, of course, that $\supset$ satisfies rules ($\supset e$)

$$M : A \supset B \quad N : A \quad \frac{}{MN : B}$$

and ($\supset i$)

$$[x : A] \left[ \begin{array}{c} M : B \quad A : \text{Prop} \end{array} \right] \quad \frac{\lambda x : A. M : A \supset B}{\lambda x : A. M : A \supset B}.$$

Definition 20. The conjunction proposition operator and its associated pairing and projection operators are defined as follows:

(a) $\Lambda \equiv \lambda u : \text{Prop}. \lambda v : \text{Prop}. (\forall w : \text{Prop})((u \rightarrow v \rightarrow w) \rightarrow w)$;

(b) $D \equiv \lambda u : \text{Prop}. \lambda v : \text{Prop}. \lambda x : u. \lambda y : v. \lambda w : \text{Prop}. \lambda z : u \rightarrow v \rightarrow w. zxy$;
(c) \(\text{fst} \equiv \lambda u : \text{Prop}. \lambda v : \text{Prop}. \lambda x : \Lambda uv.xu(\lambda y : u. \lambda z : v. y)\); and
(d) \(\text{snd} \equiv \lambda u : \text{Prop}. \lambda v : \text{Prop}. \lambda x : \Lambda uv.xu(\lambda y : u. \lambda z : v. z)\).

We use \(A \land B\) and \(A \times B\) as abbreviations for \(\Lambda AB\).

It is not at all difficult to prove from these definitions that if \(A : \text{Prop}\) and \(B : \text{Prop}\)
\[
\text{DAB} : A \rightarrow B \rightarrow A \land B,
\]
\[
\text{fstAB} : A \land B \rightarrow A,
\]
and
\[
\text{sndAB} : A \land B \rightarrow B.
\]
Furthermore, it is easy to see that if \(M : A\) and \(N : B\), then
\[
\text{fstAB(DABMN)} =* M
\]
and
\[
\text{sndAB(DABMN)} =* N.
\]

Definition 21. The disjunction proposition operator and its associated injection and case operators are defined as follows:

(a) \(\forall E \equiv \lambda u : \text{Prop}. \lambda v : \text{Prop}. (\forall w : \text{Prop}. ((u \rightarrow w) \rightarrow ((v \rightarrow w) \rightarrow w))\);
(b) \(\text{inl} \equiv \lambda u : \text{Prop}. \lambda v : \text{Prop}. \lambda x : u. \lambda w : \text{Prop}. \lambda f : u \rightarrow w. \lambda g : v \rightarrow w. fx\);
(c) \(\text{inr} \equiv \lambda u : \text{Prop}. \lambda v : \text{Prop}. \lambda y : v. \lambda w : \text{Prop}. \lambda f : u \rightarrow w. \lambda g : v \rightarrow w. gy\); and
(d) \(\text{case} \equiv \lambda u : \text{Prop}. \lambda v : \text{Prop}. \lambda z : Vuw. \lambda w : \text{Prop}. \lambda f : u \rightarrow w. \lambda g : v \rightarrow w.zwfg\).

We use \(A \lor B\) as an abbreviation for \(\forall AB\).

It is easy to show that if \(A : \text{Prop}\) and \(B : \text{Prop}\), then
\[
\text{inlAB} : A \rightarrow A \lor B,
\]
\[
\text{inrAB} : B \rightarrow A \lor B,
\]
and
\[
\text{caseAB} : A \lor B \rightarrow (\forall w : \text{Prop}. ((A \rightarrow w) \rightarrow ((B \rightarrow w) \rightarrow w))\).
\]
Furthermore, it is easy to show that if \(C : \text{Prop}\), \(M : A\), \(N : B\), \(F : A \rightarrow C\), and \(G : B \rightarrow C\), then
\[
\text{caseAB(inlABM)}CFG =* FM
\]
and
\[
\text{caseAB(inrABN)}CFG =* GN.
\]
Definition 22. \( \text{void} \equiv \bot \equiv (\forall x : \text{Prop})x. \)

We shall use \( \bot \) when we are thinking of the proposition and \( \text{void} \) when we are thinking of the type (which is supposed to be empty).

Definition 23. The existential quantifier proposition operator and its associated pairing and projection functions are defined as follows:

(a) \( \Sigma \equiv \lambda u : \text{Prop}. \lambda v : u \to \text{Prop}. (\forall w : \text{Prop})(\forall x : u)(vx \to w) \to w; \)
(b) \( D' \equiv \lambda u : \text{Prop}. \lambda v : u \to \text{Prop}. \lambda x : u. \lambda y : vx. \lambda w : \text{Prop}. \lambda z : (\forall x : u)(vx \to w) \to zxy; \) and
(c) \( \text{proj} \equiv \lambda u : \text{Prop}. \lambda v : u \to \text{Prop}. \lambda w : \text{Prop}. \lambda z : (\forall x : u)(vx \to w). \lambda y : (\forall x : u)\text{vx} \cdot \text{ywz}. \)

We use \( (\exists x : A)B \) as an abbreviation for \( \Sigma A(\lambda x : A . B) \).

It is not hard to show that if \( A : \text{Prop} \) and \( B : A \to \text{Prop} \), then

\( (\exists x : A)B : \text{Prop}, \)

\( D'AB : (\forall u : A)(Bu \supset (\exists x : A)(Bx)), \)

and

\( \text{proj}AB : (\forall x : A)((\forall u : A)(\forall v : Bu) \supset (\exists w : A)(Bw) \supset x). \)

Furthermore, if in addition \( C : \text{Prop}, M : A, N : BM, \) and

\( Z : (\forall u : A)(Bu \to C), \)

then

\( \text{proj}ABCZ(D'ABMN) =_{\ast} ZMN. \)

Note that \( D' \) differs from \( D \) only in the types postulated for some of the bound variables. But this difference is enough to make it impossible to define a right projection for \( D' \) that is correctly typed: on this point, see [4]. Of course, a modified version of \( \text{fst} \) works as a left projection function for \( D' \):

\( \text{fst}' \equiv \lambda u : \text{Prop}. \lambda v : u \to \text{Prop}. \lambda x : \Sigma uv.xu(\lambda y : u. \lambda z : v. y). \)

We can also define equality over any type:

Definition 24. The equality proposition

\( M =_A N, \)

where \( A \) is assigned type \( \text{Prop} \), is defined to be

\( QAMN, \)
where

\[ Q \equiv \lambda u : \text{Prop}. \lambda x : u. \lambda y : u. (\forall z : u \rightarrow \text{Prop}) (zx \supset zy). \]

It is not hard to show that

\[ A : \text{Prop}, X : A \vdash \lambda z : A \rightarrow \text{Prop}. \lambda u : zX. u : X =_A X \]

and


This gives us the reflexive law of the equality proposition and the substitution property; these two properties are well known to imply all the usual properties of equality.

It is not hard to see from this that we have all the usual properties of constructive predicate logic with equality.

We can also interpret classical logic. One interpretation (see [9, Section 3.31, where this is done for propositional logic) is based on the following easily proved facts about intuitionistic logic:

\[ \vdash \neg \neg A \supset \neg A, \]

\[ \neg \neg A \supset A, \neg \neg B \supset B \vdash \neg \neg (A \land B) \supset (A \land B), \]

and

\[ \neg \neg A(x) \supset A(x) \vdash \neg \neg (\forall x)A(x) \supset (\forall x)A(x). \]

Results corresponding to these can easily be proved in the theory of constructions. This means that for formulas \( A \) which are classical, that is for which \( \vdash \neg \neg A \supset A \), the logic is classical. Furthermore, all negative formulas are classical and both \( \land \) and \( \lor \) preserve classical formulas. For other classical connectives and the existential quantifier, we can use their familiar classical properties to define them:

\[ A \supset_c B \equiv \neg (A \land \neg B), \]

\[ A \lor_c B \equiv \neg (\neg A \land \neg B), \]

and

\[ (\exists_c x : A)B \equiv \neg (\forall x : A) \cdot B. \]

Since these are all negative formulas, they are all classical.

It is not hard to prove that if \( A \) is classical (in a well-formed environment \( \Gamma \)), then there is a term \( M \) all of whose free variables are assigned types in \( \Gamma \) such that

\[ \Gamma \vdash M : \neg A \lor_c A. \]

If this method of representing classical logic is used in any "applied" theory, then it is necessary to be certain that

\[ \neg \neg E \supset E \]
is provable for each formula $E$ corresponding to an atomic formula in ordinary first-order logic. To assure this, it may well be necessary to take these formulas as new assumptions.

A second method of interpreting classical logic is as follows: define

$$\text{Bool} \equiv (\forall u : \text{Prop})(u \rightarrow u \rightarrow u),$$

$$T \equiv \lambda u : \text{Prop}. \lambda x : u \cdot \lambda y : u . x,$$

and

$$F \equiv \lambda u : \text{Prop}. \lambda x : u \cdot \lambda y : u . y.$$

Here, $\text{Bool}$ represents the boolean type familiar from the usual programming languages, and $T$ and $F$ for the familiar truth values. The familiar if ... then ... else operator is defined as follows:

$$\text{Cond} \equiv \lambda u : \text{Prop}. \lambda v : \text{Bool}. \lambda x : u \cdot \lambda y : u . v \cdot u x y.$$

It is easy to prove that $T : \text{Bool}$ and $F : \text{Bool}$ and, if $A$ is any type in $\text{Prop}$ and $M : A$ and $N : A$, then

$$\text{Cond} A T M N =* M$$

and

$$\text{Cond} A F M N =* N.$$

The propositional connectives familiar to most programmers can now be defined:

$$\neg_k \equiv \lambda x : \text{Bool}. \text{Cond} \text{Bool} x F T,$$

$$\wedge_k \equiv \lambda x : \text{Bool}. \neg_k x \text{Bool} F,$$

and

$$\vee_k \equiv \lambda x : \text{Bool}. x \text{Bool} T.$$

It is then easy to prove the following:

$$\neg_k T =* F \quad \neg_k F =* T$$

$$\wedge_k T T =* T \quad \wedge_k T F =* F$$

$$\wedge_k F T =* F \quad \wedge_k F F =* F$$

$$\vee_k T T =* T \quad \vee_k T F =* T$$

$$\vee_k F T =* T \quad \vee_k F F =* F$$
We can then get implication as usual by defining
\[
\supseteq_k \equiv \lambda x : \text{Bool}. \lambda y : \text{Bool}. \neg_k (x \land_k \neg_k y),
\]
and its usual truth table properties will follow.

In this formulation of classical logic, a proof of a proposition \( A \) is not a term with that proposition as its type, but rather a term with the type \( A =_{\text{Bool}} T \). Thus, unlike the first interpretation of constructive logic, this interpretation is based on a different set of terms to represent the propositions. In fact, it is based on the idea, originally due to Frege, that there are only two propositions, \( T \) and \( F \).

Extending this second interpretation to quantifier logic is a bit complicated. The obvious way to proceed is to assume that we have a propositional function \( A \) over some domain \( D \), which is a type. In this case, this means that \( A : D \rightarrow \text{Bool} \). We would want \( (\forall x : D)(Ax) \) to be \( T \) if and only if \( AM \) is \( T \) for every \( M : D \) and to be \( F \) otherwise; but this specification assumes classical logic, whereas the type \( (\forall x : D)(Ax =_{\text{Bool}} T) \) is treated constructively by TOC0, and in general there is no term with the type
\[
(\forall x : D)(Ax =_{\text{Bool}} T) \lor (\exists x : D)(Ax =_{\text{Bool}} F).
\]

One possible solution is to use the first interpretation of classical logic, and replace \( \exists \) by \( \exists_k \). But this will only work if \( D \) is a type for which there is a term of type
\[
(\forall x : D)(\negAx =_{\text{Bool}} T) \supset Ax =_{\text{Bool}} T).
\]

A third possible method of interpreting classical logic is to add a new axiom by assigning to an atomic constant of the type \( (\forall u : \text{Prop})(\neg u \lor u) \) or the type
\[
(\forall u : \text{Prop})(\neg u \supset u).
\]
We will have more to say about this in Section 9.

7. Adding logical and mathematical assumptions

As we have seen, when logic is represented in the theory of constructions, the formulas are all represented by types in \( \text{Prop} \); the terms in these types will represent proofs. One consequence of this is that assuming a new axiom \( A \) will mean taking a new atomic constant \( c \) and adding \( c : A \) as a new assumption to the environment.

Now the way we have proved the strong normalization theorem in Section 3 guarantees that such constants can be added without interfering with the proof of the theorem provided that these new constants do not occur at the heads of new redexes. But this
is just the way new axioms are added. Thus, adding new axioms does not have any effect on the strong normalization theorem.

But adding new axioms may well affect the consistency of the system. Suppose, for example, we assume \( c : \bot \). This amounts to assuming as an axiom \( \bot \), i.e., to assuming the inconsistency of the system. Another such set of assumptions is the following:

\[
c_1 : \text{Prop}, c_2 : c_1, c_3 : \neg c_1.
\]

The strong normalization theorem does, however, imply the consistency of the empty environment, and thus of the system TOC0 itself.

**Theorem 18.** There is no term \( M \) such that

\[ \vdash M : \bot. \]

**Proof.** Let \( \mathcal{D} \) be a normalized deduction with no undischarged assumptions of

\[ M : \bot. \]

Then for any new variable \( x \), the following is a valid deduction:

\[
\begin{array}{c}
\mathcal{D} \\
M : \bot \\
x : \text{Prop} \\
\hline
Mx : x.
\end{array}
\]

This deduction can be normalized without changing the conclusion or any undischarged assumptions. Since the type of the conclusion of this normalized deduction has complexity 0 and since the deduction is normal, there can be no inference in the main branch of the deduction by \((\forall \text{e})\). This means that the formula at the top of the main branch is an undischarged assumption, contradicting the assumption about \( \mathcal{D} \). \( \square \)

Note that this proves the consistency of the higher-order constructive logic of the previous section.

If this system is to be used as the basis of a proof development system that is to be of practical use in situations in which we need confidence in the validity of the proofs obtained, we will need to make assumptions and, at the same time, we will need confidence that the assumptions do not allow us to deduce that there is a term in \( \bot \). Assumptions satisfying this condition are important enough that they deserve a definition.

**Definition 25.** A valid set \( \Gamma \) of assumptions is said to be consistent if there is no term \( M \) such that there is a proof in TOC0 of

\[ \Gamma \vdash M : \bot. \]

Then another way to state Theorem 18 is that the empty set of assumptions is consistent.
Any application of a theorem-prover built on TOC will involve assumptions in order to represent notions from logic, mathematics, or computer science. In order to have sufficient confidence in the soundness of the proofs constructed with its aid, it is necessary to have some confidence that the sets of assumptions involved are consistent in this sense.

The proof of Theorem 18 shows that proving that there is no term \( M \) such that \( \Gamma \vdash M : \bot \) is equivalent to proving that there is no term \( N \) such that \( \Gamma, x : \text{Prop} \vdash N : x. \)

As an example of a deduction which can lead to this kind of conclusion, suppose that \( x : \text{Prop}, A : \text{Prop}, N : A, \) and \( M : (\forall z : A \rightarrow \text{Prop})(zN) \). Then we have

\[
\frac{x : \text{Prop} \quad A : \text{Prop} \quad (\forall Pi - v)}{M : (\forall z : A \rightarrow \text{Prop})(zN)}
\]

This is an example of the kind of deduction that must be prevented, and to prevent it we need to exclude such assumptions as

\( M : (\forall z : A \rightarrow \text{Prop})(zN) \).

Now if there is a deduction of \( \Gamma, x : \text{Prop} \vdash N : x \), then there is a normalized deduction of it, the left branch consists of major premises for \((\forall e)\) and \((\text{Eq''})\), and the assumption at the top of the branch is not discharged. The following definition gives a set of conditions on assumptions which make it impossible for any one of them to occur at the top of the left branch of such a normal deduction.

To state the definition, recall that each type converts to the form

\[
(\forall y_1 : B_1)(\forall y_2 : B_2)\ldots(\forall y_m : B_m)S,
\]

where \( S \) is a simple type. Let us call \( n \) the index of the type and \( S \) its tail. Then in the above example that we need to exclude, the variable at the head of \( zN \) (the tail of the type of \( M \)) has a type whose tail is \( \text{Prop} \). This seems to be the key property of the assumptions that need to be excluded.

**Definition 26.** Let \( \Gamma \) be a well-formed environment of the form

\[
x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n.
\]

For each \( i \), let \( A_i \) convert to

\[
(\forall y_{i1} : B_{i1})(\forall y_{i2} : B_{i2})\ldots(\forall y_{im_i} : B_{im_i})S_i,
\]

where \( S_i \) is the tail of \( A_i \). Then \( \Gamma \) is strongly consistent if for each \( A_i \) for which \( m_i > 0 \) and \( S_i \) converts to the form \( z_iM_{i1}M_{i2}\ldots M_{il} \), if \( z_i \) is a variable (and hence one of the \( x_j \) or \( y_{jk} \)), then the tail of its type does not convert to \( \text{Prop} \).

It follows immediately from the definition that any strongly consistent environment is consistent.
Note how weak this definition is. No type in a strongly consistent environment can have any of the following forms: \( A \land B, A \lor B, (\exists x : A)B, \bot, \neg A, \) or \( M =_A N. \)

Although a negative formula cannot occur in a strongly consistent environment, there are well-formed environments involving negative formulas which can be proved consistent:

**Definition 27.** Let \( \Gamma_0 \) be a strongly consistent environment. Let \( \Gamma_1 \) consist of assumptions of the form \( u : \neg B, \) where, under the assumptions of \( \Gamma_0, B \) is a small simple type (i.e., a simple type that is in \( \text{Prop} \) under the right assumptions) but \( B \) does not convert to the type of an assumption in \( \Gamma_0. \) Let \( \Gamma_2 \) consist of assumptions of the form \( v : \neg \neg B, \) where, under the assumptions of \( \Gamma_0, B \) is a small simple type and where \( \neg B \) does not convert to the type of an assumption in \( \Gamma_1 \) (but \( B \) may convert to the type of an assumption in \( \Gamma_0 \)). If \( \Gamma = \Gamma_0, \Gamma_1, \Gamma_2, \) then \( \Gamma \) is said to be strongly negation consistent.

**Theorem 19.** A strongly negation consistent environment is consistent.

**Remark.** Clearly, if \( B \) converts to \( C, \) then \( u : \neg B, v : C \vdash uv : \bot. \) What this theorem says is that if \( B \) is a small simple type, this is essentially the only way to get a contradiction.

**Proof of Theorem 19.** Suppose \( \Gamma \) is strongly negation consistent and suppose that for some term \( M \)

\[ \Gamma \vdash M : \bot. \]

then for a variable \( w \) which is not free in \( \Gamma, \) we have for some term \( M' \)

\[ \Gamma, w : \text{Prop} \vdash M' : w. \]

Normalize this deduction and let the result be \( \mathcal{D}. \) Suppose that there is no proper subdeduction of \( \mathcal{D} \) which proves either

\[ \Gamma' \vdash M'' : \bot \quad \text{or} \quad \Gamma', w : \text{Prop} \vdash M''' : w \]

for any other strongly negation consistent \( \Gamma''; \) otherwise we can begin with this proper subdeduction. (Here proper subdeduction means that there is more difference than the one inference necessary to go back and forth between a conclusion whose type is \( \bot \) and one whose type is \( w. \) ) Now the last inference in \( \mathcal{D} \) which differs from (Eq'') cannot be (\( \forall x \)) ; thus it must be (\( \forall e \)). It follows that the left branch of \( \mathcal{D} \) consists of inferences by (\( \forall e \)) and (Eq''), and hence the top of the left branch is not discharged. This assumption at the top of the left branch must be in \( \Gamma_1 \) or \( \Gamma_2. \)
Case 1: It is in $\Gamma_1$. Then it is $u : \neg B$ for a small simple type $B$ not convertible to a type in $\Gamma_0$ or $\Gamma_2$, and $\mathcal{D}$ is

$$\begin{align*}
&\frac{w : \text{Prop}, u : \neg B}{\mathcal{D}_1} \\
u : \neg B &\quad M'' : B \\
u M'' : \bot \quad (\forall e) \\
\quad \mathcal{D}_2 \\
M' : w.
\end{align*}$$

Now clearly no assumption of $\mathcal{D}_1$ is discharged in $\mathcal{D}_2$. Hence, since $B$ is a simple type, the top of the left branch of $\mathcal{D}_1$ must be in $\Gamma_1$ or $\Gamma_2$. Hence, $\mathcal{D}_1$ is

$$\begin{align*}
&\frac{w : \text{Prop}, u : \neg B, u' : \neg B'}{\mathcal{D}_3} \\
u' : \neg B' &\quad M''' : B' \\
u' M''' : \bot \quad (\forall e) \\
\quad \mathcal{D}_4 \\
M'' : B,
\end{align*}$$

where $B'$ is simple or the negation of a simple type. But then

$$\begin{align*}
&\frac{w : \text{Prop}, u : \neg B, u' : \neg B'}{\mathcal{D}_3} \\
u' : \neg B' &\quad M''' : B' \\
u' M''' : \bot \quad (\forall e) \\
\quad \mathcal{D}_4 \\
M'' : B,
\end{align*}$$

is a proper subdeduction of $\mathcal{D}$ contradicting the assumption about $\mathcal{D}$. Hence, the top of the left branch of $\mathcal{D}$ is not in $\Gamma_1$.

Case 2: It is in $\Gamma_2$. Then it is $u : \neg \neg B$, where $B$ is a small simple type, and $\mathcal{D}$ is

$$\begin{align*}
&\frac{w : \text{Prop}, u : \neg \neg B}{\mathcal{D}_1} \\
u : \neg \neg B &\quad u M'' : \neg B \\
u M'' : \bot \quad (\forall e) \\
\quad \mathcal{D}_2 \\
M' : w.
\end{align*}$$

The argument of Case 1 shows that the last inference in $\mathcal{D}_1$ which differs from $(\text{Eq}''')$ is not $(\forall e)$, so it must be by $(\forall \text{Pi})$, and $\mathcal{D}_1$ is
\[ \begin{align*}
\text{Prop, } u & : \neg B, [v : B] \\
\text{Prop, } v & : B \\
M'' & : \bot \\
\lambda v : B. M''' & : \neg B \\
\end{align*} \]

\[ \frac{M''' : \bot}{\lambda v : B. M''' : \neg B}, (\forall \Pi - 1) \]

where \( M'' \) converts to \( \lambda v : B. M''' \). But then \( D_3 \) is a proper subdeduction of \( D \) contradicting our assumption. Hence, the assumption at the top of the left branch of \( D \) is not in \( \Gamma \).

This shows that \( \Gamma \) is consistent. \( \Box \)

In the rest of the paper, we will look at some assumptions that cannot occur in strongly consistent or strongly negation consistent environments.

8. Negations of equations

There are environments involving negation other than strongly negation consistent environments that can be proved consistent. Thus, suppose \( \Gamma \) is

\[ x_1 : \neg A_1, x_2 : \neg A_2, \ldots, x_n : \neg A_n, \]

where \( \neg A \) is defined to be \( A \supset \bot \). It is possible to show that \( \Gamma \) is consistent by showing that there is no term \( M \) for which

\[ \Gamma \vdash M : A_i \]

for any \( i \). As an example, we can prove that negations of equations between terms with distinct normal forms are consistent if all the other assumptions are in a strongly consistent environment.

**Theorem 20.** Let \( \Gamma_1 \) be a set of assumptions in which each formula assigns to a (distinct) constant a type which converts to the form \( \neg P =_A Q \) for terms \( P \) and \( Q \) of type \( A \) with distinct normal forms. Suppose that \( \Gamma_2 \) is a strongly consistent environment. Suppose that there is a closed term \( R \) such that

\[ \Gamma_1, \Gamma_2 \vdash R : M =_B N. \]

Then

\[ M =_B N. \]

**Proof.** Let \( D \) be a deduction in normal form of

\[ \Gamma_1, \Gamma_2 \vdash R : M =_B N. \]

We proceed by induction on the structure of \( D \). Thus, we may suppose as part of the induction hypothesis that the theorem holds for any proper subdeduction of \( D \) whose
undischarged assumptions satisfy the hypotheses of the theorem. Suppose that the last
inference in \( \mathcal{D} \) (except for equality rules) is by \((\forall e)\). Because \( \mathcal{D} \) is normal, the only
inferences in the left branch of \( \mathcal{D} \) are \((\forall e)\) and \((\text{Eq}''')\). Consider the formula at the
top of the left branch of \( \mathcal{D} \). Because of the form of \( \mathcal{D} \) and of the rules of TOC0, this
formula is not a discharged assumption. It must thus have the form \( x : C \), where
\[
C = e (\forall y_1 : C_1)(\forall y_2 : C_2) \ldots (\forall y_k : C_k)(M =_B N).
\]
By Definition 26, a type of this form cannot occur in a strongly consistent environment;
hence it is in \( \Gamma_1 \), and \( C =_* \neg P =_A Q \). It follows that the deduction of the minor (right)
premise for the inference by \((\forall e)\) of which the formula in question is the major (left)
premise is a proper subdeduction of \( \mathcal{D} \) whose conclusion has the form \( S : P =_A Q \)
for a closed term \( S \) and terms \( P \) and \( Q \) with distinct normal forms, contradicting the
assumption that the theorem holds for any proper subdeduction of \( \mathcal{D} \). Hence, the last
non-equality inference in \( \mathcal{D} \) is not by \((\forall e)\).

Since
\[
M =_B N =_* (\forall z : B \rightarrow \text{Prop})(zM \supset zN),
\]
it follows that the last non-equality inference is by \((\forall Ti)\),
\[
R \equiv \lambda z : B \rightarrow \text{Prop}. P,
\]
and \( \mathcal{D} \) has the form (possibly modulo some manipulations involving rules \((\text{Eq}'P)\),
\((\text{Eq}'T)\), and \((\text{Eq}'')\); we will not bother to mention this fact again in what follows)
\[
1
\begin{array}{c}
[z : B \rightarrow \text{Prop}] \hline
\mathcal{D}_1(z) \\
P : zM \supset zN \\
\frac{B : \text{Prop} \quad \text{Prop} : \text{Type}}{B \rightarrow \text{Prop} : \text{Type}} \quad (\text{PTF} - v) \\
\frac{\lambda z : B \rightarrow \text{Prop}. P \quad (\forall z : B \rightarrow \text{Prop})(zM \supset zN)}{(\forall Ti - 1)}
\end{array}
\]
where \( z \) is a variable which does not occur free in \( \Gamma_1, \Gamma_2, M, \) or \( N \). Since \( z : A \rightarrow \text{Prop} \)
cannot occur in a strongly consistent environment, an argument similar to the above
argument for \( \mathcal{D} \) shows that the last non-eq inference in \( \mathcal{D}_1(z) \) is not by \((\forall e)\). Hence,
the last non-eq inference in \( \mathcal{D}_1(z) \) is by rule \((\forall Pi)\), \( P =_* \lambda w : zM \cdot Q \), and \( \mathcal{D}_1(z) \) has the form
\[
1
\begin{array}{c}
[w : zM] \\
\mathcal{D}_2(w) \\
\frac{z : B \rightarrow \text{Prop}}{z : B \rightarrow \text{Prop} \quad M : B} \quad (\rightarrow e) \\
\frac{zM : \text{Prop} \quad M : B} {zM : \text{Prop} \quad M : B} \quad (\forall Pi - 2)
\end{array}
\]
where \( w \) is a variable distinct from \( z \) which does not occur free in \( \Gamma_1, \Gamma_2, M, \) or \( N \).
By an argument similar to that above, the last inference in \( \mathcal{D}_2(w) \) is not by rule \((\forall e)\).
Furthermore, any deduction of \( Q : zN \) must use the hypothesis \( w : zM \). Since \( \mathcal{D}_2(w) \)
is normal and \( zM \) and \( zN \) are simple types, it is not hard to see that the only rule that can occur in \( \varnothing_2(w) \) is (Eq'), from which it follows that \( Q \equiv w \) and, more important, \( M =_{\pi} N \). \( \square \)

**Corollary 20.1.** If \( \Gamma_1, \Gamma_2 \) is as in the theorem, then it is consistent.

In case \( \Gamma_1 \) is empty, we have the following additional consequence of the theorem.

**Corollary 20.2.** If \( \Gamma \) is a strongly consistent environment, and if there is a closed term \( R \) such that

\[ \Gamma \vdash R : M =_{B} N, \]

then

\[ M =_{\pi} N. \]

This theorem can be generalized somewhat. For example, if the types of the variables are suitably restricted to prevent substitution instances of \( P \) and \( Q \) which are convertible to each other, it is presumably possible to prove a version of the theorem for universally quantified inequalities or for implications whose consequents are inequalities.

9. Arithmetic

Arithmetic is interpreted as in the second-order polymorphic typed \( \lambda \)-calculus. When modified for TOC0, this is as follows:

**Definition 28.** (a) \( N \equiv (\forall A : \text{Prop})((A \rightarrow A) \rightarrow (A \rightarrow A)) \);

(b) \( 0 \equiv \lambda A : \text{Prop}. \lambda x : A \rightarrow A. \lambda y : A. y; \)

(c) \( \sigma \equiv \lambda u : N. \lambda A : \text{Prop}. \lambda x : A \rightarrow A. \lambda y : A. \lambda z : A. x(u A xy); \)

(d) \( \pi \equiv \lambda u : N. \text{snd} N N(u(N \times N) Q(D N N00)), \)

where \( Q \equiv \lambda v : N \times N. \) D N N(\( \sigma (\text{fstNNv})))(\text{fst} N Nv); \) and

(e) \( R \equiv \lambda A : \text{Prop}. \lambda x : A. \lambda y : N \rightarrow A \rightarrow A. \lambda z : N. z(N \rightarrow A)P(\lambda w : N. x)z, \)

where \( P \equiv \lambda v : N \rightarrow A. \lambda w : N. y(\pi w)(v(\pi w)). \)

The term \( n \), which represents the natural number \( n \), is defined to be

\[ \sigma(\sigma(\ldots(\sigma(0)\ldots))), \]

It is not hard to show that

\( 0 : N, \)

\( \sigma : N \rightarrow N, \)

\( \pi : N \rightarrow N, \)
and
\[ R : (\forall A : \text{Prop})(A \to (N \to A \to A) \to N \to A). \]

It is also easy to show that
\[ n =_{*} \lambda A : \text{Prop}. \lambda x : A \to A. \lambda y : A. \overbrace{x(x(\ldots(x y)\ldots))}^{n}, \]
\[ \pi 0 =_{*} 0, \]
\[ \pi (\sigma n) =_{*} n, \]
and also, for any type \( A : \text{Prop} \) and any terms \( M \) and \( N \) of types \( A \) and \( N \to A \to A \), respectively,
\[ \text{RAMN} 0 =_{*} M, \]
and
\[ \text{RAMN} (\sigma n) =_{*} N n (\text{RAMN} n). \]

It is also not hard to show that
\[ N : \text{Prop}. \]

We know that this definition works in the sense that we can define all primitive recursive functions and that the peano axioms hold. However, our knowledge of the peano axioms is entirely metatheoretic; we do not get the formulas representing these axioms as theorems of \( \text{TOC0} \). To get the peano axioms holding formally within \( \text{TOC0} \), we need to add some new axioms. The first two axioms we need are obvious:
\[ \text{Peano1} \equiv (\forall n : N)(\neg \sigma n =_{N} 0) \]
and
\[ \text{Peano2} \equiv (\forall m : N)(\forall n : N)(\sigma m =_{N} \sigma n \supset m =_{N} n). \]

We also need the induction axiom:
\[ \text{Peano} \equiv (\forall A : N \to \text{Prop})((\forall m : N)(Am \supset A(\sigma m)) \supset A 0 \supset (\forall n : N)(An)). \]

Since the defining equations for + and \( \times \) follow from the reduction properties of \( R \) and rule (Eq’’), it may appear that we have everything we need for arithmetic.

However, we are not finished. For although the only closed terms of type \( N \) are known to be natural numbers (except for \( \lambda A : \text{Prop}. \lambda x : A \to A. x \), a term \( \eta \)-convertible but not \( \beta \)-convertible to 1; however, this term is not really something other than a natural number), so that the axiom \( \text{Peano} \) does not really restrict the domain of objects in \( N \), we do need to be able to talk about objects in other types which are not natural numbers. We may even want to create a supertype of \( N \), and in such a
supertype, where we will have things which are not natural numbers, we will want to be able to assert that an object is not a natural number. To do this, we need to be able to say that something is a natural number. And so far, we have no way of doing this that is part of the logic; we have only

\[ M : N, \]

which is definitely not the same thing. Thus, we need a predicate of the logic, \( \mathcal{N} \), which says that something is a natural number. The definition we want is as follows:

\[
\mathcal{N} \equiv \lambda n : N. (\forall A : N \to \text{Prop})(\forall m : N)(Am \supset A(am)) \supset A_0 \supset An.
\]

It is easy to prove

\[ \vdash \mathcal{N} : N \to \text{Prop}, \]

\[ \vdash M : \mathcal{N} 0, \]

\[ \vdash N : (\forall n : N)(\mathcal{N} n \supset \mathcal{N}(\sigma n)), \]

for some closed terms \( M \) and \( N \).

Now that we have the definition of \( \mathcal{N} \), we no longer need the axiom \text{Peano}, for it is easy to prove (this is not mentioned in [20] or [21]) that there is a closed term \( M \) such that

\[ \vdash M : (\forall A : N \to \text{Prop})(\forall m : N)(Am \supset A(am)) \supset A_0 \supset (\forall n : N)(\mathcal{N} n \supset An)). \]

While this is not exactly \text{Peano}, it is close enough for practical purposes. What \text{Peano} actually does is to say that the induction principle holds formally for the type \( N \). We know metatheoretically that it holds for \( N \), but without the axiom \text{Peano}, we do not have the result as a formal theorem of TOCO. Since we do have that formal knowledge about \( \mathcal{N} \), it is difficult to imagine circumstances in which this formal knowledge about \( N \) would be necessary.

This leaves us with the axioms \text{Peano1} and \text{Peano2}. These two axioms appear to constitute a minor variation of the well-formed environment \( \Gamma \) of Theorem 20. This fact was used in [37, Theorem 5.3] to prove the consistency of \( c_1 : \text{Peano1}, c_2 : \text{Peano2} \). However, it has since turned out that \text{Peano2} can be replaced by a provable result.

**Lemma 12.** For some term \( M \),

\[ \vdash M : (\forall n : N)(\mathcal{N} n \to \pi(\sigma n) =_N n). \]

**Remark 16.** This is proved as in [39, Lemmas 4.1–4.2].

**Proof of Lemma 12.** A direct calculation gives that \( \pi(\sigma(\sigma n)) =_* \sigma(\pi(\sigma n)) \). Hence, there is a term \( M_1 \) such that

\[ n : N, x : \pi(\sigma n) =_N n \vdash M_1 : \pi(\sigma(\sigma n)) =_N \sigma n. \]
Hence, by \((\forall \Pi_i)\), there is a term \(M_2\) such that

\[ \vdash M_2 : (\forall n : \mathbb{N})(\pi(\sigma n) =_N n \rightarrow \pi(\sigma(\sigma n)) =_N \sigma n). \]

This is the induction step. The basis is easy, since \(\pi 0 =_N 0\). Then induction (which follows from the definition of \(N\)) gives us the lemma. \(\square\)

**Lemma 13.** For some term \(M\),

\[ \vdash M : (\forall n : \mathbb{N})(\forall n : \mathbb{N})(N^n \rightarrow N^m \rightarrow \sigma n =_N \sigma m \rightarrow n =_N m). \]

**Proof.** We can easily formalize in this logic the following argument, where \(n = m\) represents \(n =_N m\): \(\pi n = \pi m\), therefore \(\pi(\sigma n) = \pi(\sigma m)\), and so \(n = m\). \(\square\)

Thus, arithmetic can be interpreted with only one assumption: \(c : \text{Peano1}\). ([38, Footnote 2, p. 437] is in error; the proof given by Coquand there is for another system and is invalid in TOCO.) This is a minor variation on the environment \(\Gamma\) of Theorem 20:

**Theorem 21.** If \(\Gamma\) be a strongly consistent environment, then \(\Gamma, c : \text{Peano1}\) is consistent.

**Proof.** This will be proved by showing that it is impossible to have a deduction of

\[ (14) \quad \Gamma, c : \text{Peano1}, z : \text{Prop} \vdash M : z, \]

where \(z\) does not occur free in \(\Gamma\). Thus, suppose that \(\mathcal{D}\) is a normalized deduction of (14), and suppose without loss of generality that it is the shortest possible such deduction. Then the last rule (aside from (Eq'')) must be \((\forall e)\). Then the formula at the top of the left (main) branch is not cancelled, and since it cannot be in \(\Gamma\) (because \(\Gamma\) is strongly consistent) and cannot be \(z : \text{Prop}\), it must be \(c : \text{Peano1}\), and \(\mathcal{D}\) has the form

\[
\frac{z : \text{Prop}}{c : (\forall n : \mathbb{N})(\exists 0 =_N \sigma n)} \quad \mathcal{D}_1(z) \quad m : \mathbb{N} \quad (\forall e) \quad z : \text{Prop} \quad \mathcal{D}_2(z) \quad M_1 : 0 =_N \sigma m \quad (\forall e) \quad z : \text{Prop} \quad \mathcal{D}_{21}(z).
\]

Now consider \(\mathcal{D}_2(z)\). Without abbreviations it is

\[
\frac{z : \text{Prop}}{\mathcal{D}_2(z)} \quad M_1 : (\forall r : \mathbb{N} \rightarrow \text{Prop})(r 0 \supset r(\sigma m)).
\]
Case 1: The last rule of $D_2(z)$ is $(\forall e)$. Then the top of the left branch must again be $c : \text{Peano1}$, so $D_2(z)$ has the form

\[
\begin{array}{c}
z : \text{Prop} \\
c : (\forall n : N)(\neg 0 =_N an) \\
m' : N \\
\frac{cm' : \neg 0 =_N \sigma m'}{M_2 : 0 =_N \sigma m'} \quad (\forall e) \\
\end{array}
\]

But then

\[
\begin{array}{c}
z : \text{Prop} \\
c : (\forall n : N)(\neg 0 =_N an) \\
m' : N \\
\frac{cm' : \neg 0 =_N \sigma m'}{M_2 : 0 =_N \sigma m'} \quad (\forall e) \\
\end{array}
\]

is a deduction of (14) shorter than $D$, contrary to hypothesis. Hence, this case is impossible.

Case 2: The last rule of $D_2$ is $(\forall Ti)$. Then $D_2$ is

\[
\begin{array}{c}
z : \text{Prop, } [r : N \rightarrow \text{Prop}] \\
M_2 : r0 \supset r(\sigma m) \\
\frac{\lambda r : N \rightarrow \text{Prop}.M_2 : (\forall r : N \rightarrow \text{Prop})(r0 \supset r(\sigma m)).}{(\forall Ti - 1)} \\
\end{array}
\]

Now consider $D_2(z, r)$. By the argument of Case 1, the last rule (except, of course, for $(Eq'')$) is not $(\forall e)$. Hence, it is $(\forall Ti)$, and $D_2(z, r)$ is

\[
\begin{array}{c}
z : \text{Prop, } r : N \rightarrow \text{Prop}, [u : r0] \\
M_2 : r0 \supset r(\sigma m) \\
\frac{\lambda u : r0.M_3 : r0 \supset r(\sigma m),}{(\forall Pi - 2)} \\
\end{array}
\]

where $M_2 \equiv \lambda u : r0.M_3$.

Now consider $D_2(z, r, u)$. By the argument of Case 1, the last inference is not $(\forall e)$. It cannot be by $(\forall Ti)$ or $(\forall Pi)$. The only rule left is $(Eq''')$, but this is impossible since $r0 \neq r(\sigma m)$. Hence, there can be no deduction of (14). \qed

The theory of arithmetic we have just seen is an excellent prototype for inductively generated free algebras, which can all be defined by similar methods (cf. [3]). It is not
strictly necessary to have definitions for the types and constants involved: the above theory would work just as well if \( N, 0, \sigma, \) and \( R \) are new atomic constants (of course, the reduction rules for \( R \) have to be postulated in this case; we can have confidence that there is no problem with the strong normalization theorem if these new constants are assumed precisely because we can define all of them as closed terms from which the reduction rules for \( R \) can be deduced). If we do take them as atomic constants, then \texttt{Peano} can be interpreted as saying that type \( N \) is assigned only to terms in the set \( \mathcal{N} \), and so we are justified in concluding the consistency of the system with axiom \texttt{Peano} added.

As an example of an inductively generated free algebra, let us consider lists. To have lists of terms of type \( A \), we need a type \texttt{List} which, when applied to \( A \), forms the type \texttt{List} \( A \) of lists of objects of type \( A \). We also need the empty list, \texttt{nil} \( A \), and the function \texttt{cons} \( A \) of type \( A \rightarrow \texttt{List} \( A \rightarrow \texttt{List} \( A \) \) which puts an object of type \( A \) at the front of a list of objects of type \( A \) to produce a new list of objects of type \( A \). We will want to be able to define recursively functions on lists and objects of type \( A \). For example, the function \texttt{append} which concatenates two lists, is defined as follows, where \( L_1 \) and \( L_2 \) are lists of type \texttt{List} \( A \) and \( M : A :\):

\[
\texttt{append} \ A (\texttt{nil} \ A) L_2 \equiv L_2,
\]

\[
\texttt{append} \ A (\texttt{cons} A ML_1) L_2 \equiv \texttt{cons} A M (\texttt{append} A L_1 L_2).
\]

To take another example, the function \texttt{reverse} which reverses the order of a list is defined by

\[
\texttt{reverse} AL \equiv \texttt{flip} AL (\texttt{nil} \ A),
\]

where \texttt{flip} is defined by

\[
\texttt{flip} \ A (\texttt{nil} \ A) L_2 \equiv L_2,
\]

\[
\texttt{flip} \ A (\texttt{cons} A ML_1) L_2 = \texttt{flip} A L_1 (\texttt{cons} A M L_2),
\]

To make definitions like this, we need a term which plays with respect to lists the role that \( R \) plays with respect to \( N \).

It turns out to be possible to define \texttt{List}, \texttt{nil}, and \texttt{cons} so that these recursive definitions become possible:

\[
\texttt{List} \equiv \lambda A . \texttt{Prop} . (\forall u : \texttt{Prop} . ((A \rightarrow u \rightarrow u) \rightarrow u \rightarrow u)),
\]

\[
\texttt{nil} \equiv \lambda A : \texttt{Prop} . \lambda B : \texttt{Prop} . \lambda f : A \rightarrow B \rightarrow B . \lambda y : B . y,
\]

\[
\texttt{cons} \equiv \lambda A : \texttt{Prop} . \lambda x : A . \lambda l : \texttt{List} A . \lambda B : \texttt{Prop} . \lambda f : A \rightarrow B \rightarrow B . \lambda y : B . f x (l B f y).
\]

The intention is that if \( L =_{\star} (x_1, x_2, \ldots, x_n) \) is a list in \texttt{List} \( A \), \( f : A \rightarrow B \rightarrow B \), and \( y : B \), then

\[
L B f y \upharpoonright f x_1 (f x_2 (\ldots (f x_n y) \ldots)).
\]
To show that this definition works, note that if \( h : A \rightarrow B \rightarrow B \) and \( M : B \), and if \( g \) is defined by

\[
g \equiv \lambda l : \text{List}A. IBhM,
\]

then \( g \) has the properties

\[
g(\text{nil}A) \triangleright M,
\]

\[
g(\text{cons}Axl) \triangleright h(xgL),
\]

for all \( x : A \) and \( L : \text{List}A \). This function \( g \) allows us to define \text{append}, \text{reverse}, and such other list functions as \text{null} and \text{length}. It is also possible to define \text{car}, but since there may not be any terms with type \( A \) (for example, if \( A \) is \text{void}), it must be defined to have three arguments, so that

\[
\text{car}AM(\text{nil}A) \triangleright M,
\]

\[
\text{car}AM(\text{cons}Axl) \triangleright x.
\]

I conjecture that it is possible to define \text{cdr}, but I am not certain as of this writing.

Just as we defined \( N \) corresponding to \( \mathbb{N} \), so we can define \( L \) corresponding to \( \text{List} \). The definition is as follows:

\[
L \equiv \lambda A : \text{Prop}. \lambda x : \text{List}A. (\forall y : \text{List}A \rightarrow \text{Prop})

\[
(\forall u : A) (\forall l : \text{List}A) (yA \triangleright yA(\text{cons}Aul)) \triangleright yA(\text{nil}A) \triangleright yx).
\]

It is then easy to prove

\[
\vdash L : (\forall A : \text{Prop})(\text{List}A \rightarrow \text{Prop}),
\]

\[
\vdash M : (\forall A : \text{Prop})(L A(\text{nil}A)),
\]

\[
\vdash N : (\forall A : \text{Prop})(\forall u : A)(\forall l : \text{List}A)(L A l \triangleright L A(\text{cons}Aul)),
\]

and

\[
\vdash P : (\forall A : \text{Prop})(\forall B : \text{List}A \rightarrow \text{Prop})

\[
((\forall u : A) (\forall l : \text{List}A) (Bl \triangleright B(\text{cons}Aul)) \triangleright B(\text{nil}A) \triangleright (\forall l : \text{List}A)(L l \triangleright Bl)),
\]

for some closed terms \( M, N, \) and \( P \). This gives us the desired induction property on lists. All we still need are axioms corresponding to \text{Peano1} and \text{Peano2}:

\[
(\forall A : \text{Prop})(\forall x : A)(\forall l : \text{List}A)(\neg \text{cons}Axl =_{\text{List}A} \text{nil}A),
\]

\[
(\forall A : \text{Prop})(\forall x : A)(\forall y : A)(\forall l : \text{List}A)(\forall m : \text{List}A)

\[
(\text{cons}Axl =_{\text{List}A} \text{cons}Aym \triangleright x =_{A} y),
\]
and

\((\forall A : \text{Prop})(\forall x : A)(\forall y : A)(\forall l : \text{List}A)(\forall m : \text{List}A)
\)
\((\text{consAx}l =_{\text{List}A} \text{consAy}m \supset l =_{\text{List}A} m)\).

I have not checked the details for any of these, but it appears that the second can be proved by the use of \text{carAx}. The third may not be provable as is, but can probably be proved in a modified form in a manner similar to the proof of Lemma 12:

\((\forall A : \text{Prop})(\forall x : A)(\forall y : A)(\forall l : \text{List}A)(\forall m : \text{List}A)
\)
\((\mathcal{L}l \supset \mathcal{L}m \supset \text{consAx}l =_{\text{List}A} \text{consAy}m \supset l =_{\text{List}A} m)\).

It follows that the representation of lists probably requires only the first of these, and this is enough like the negation of an equation between terms with distinct normal forms that it may possibly be proved consistent by a proof similar to that of Lemma 13. These details, like the definition of \text{cdr}, are left for further work.


10. Classical logic

In his posting to the TYPES network [33], Pottinger shows that if excluded middle and definite descriptions are added to TOCO, then any two terms in a small type (i.e., a type in \text{Prop}) are equal (in the sense of Leibniz equality). This conclusion is called “proof degeneracy”. Because of the interpretation of arithmetic, there is a small type in which there are terms asserted to be unequal (0 and \(\sigma0\)), and hence proof degeneracy together with the assumption needed for arithmetic implies inconsistency.

Coquand [8] showed by a model theoretic proof that excluded middle in the calculus of constructions is consistent. In this section I prove the equivalent result that excluded middle without definite descriptions does not imply proof degeneracy. My method is proof theoretic, a variant of the \(\neg\neg\)-interpretation, and it provides an alternative proof of Coquand's consistency result.

Coquand [8] also proved that excluded middle and strong sums imply proof degeneracy. Garrel Pottinger has pointed out to me (private communication) that the strong sums, when interpreted under the formulas-as-types isomorphism, have the disjunction property. Since this property is known to be characteristic of constructive logic and incompatible with classical logic, this result of Coquand is really a confirmation of what we should expect of classical logic. The result of Pottinger [33], on the other hand, is unwelcome, since both excluded middle and definite descriptions are desirable in some circumstances. The result proved here shows that we are more likely to have to give up definite descriptions than excluded middle.

To get classical logic, I will use the assumption

\(\text{cl} : (\forall u : \text{Prop})(\neg
\neg u \supset u)\),
which implies classical logic but cannot occur in a strongly consistent environment.

To simplify the notation, let $\text{CL}$ be an abbreviation for

$$\forall u : \text{Prop} \, (\neg \neg u \supset u).$$

We want each occurrence of $cl : \text{CL}$ as an assumption to occur in a subdeduction of the form

$$\frac{\text{cl} : \text{CL}}{\text{cl} \text{A} : \neg \neg A \supset A} \quad (\forall e)$$

$$\frac{\text{cl} \text{AM} : A.}{\text{cl} \text{AM} : A.}$$

This is not a difficult restriction to satisfy, since we can replace

$$\frac{\text{cl} : \text{CL}}{\text{cl} \text{A} : \neg \neg A \supset A,}$$

where the conclusion is not a major premise for $(\forall e)$, by

$$\frac{\text{cl} : \text{CL}}{\text{cl} \text{A} : \neg \neg A \supset A,} \quad (\forall e)$$

where $D_2$ is

$$\frac{\text{A} : \text{Prop}}{\bot : \text{Prop}} \quad (\text{PPF - v})$$

$$\frac{\bot : \text{Prop}}{\neg \neg A : \text{Prop},}$$

and where $D_v$ is

$$\frac{\text{Prop} : \text{Type}}{\bot : \text{Prop};} \quad (\text{TPF - n})$$

also, if $\text{cl} : \text{CL}$ is not the major premise for an inference by $(\forall e)$, then we can replace it by

$$\frac{\text{cl} : \text{CL}}{\text{cl} \text{xy} : x} \quad (\forall e)$$

$$\frac{\text{cl} \text{xy} : x}{{\lambda} x : \text{Prop.} \, \text{cl} \text{xy} : x} \quad (\forall e)$$

$$\frac{\text{cl} \text{xy} : x}{{\lambda} x : \text{Prop.} \, \text{cl} \text{xy} : x} \quad (\forall e)$$

$$\frac{\text{cl} \text{xy} : x}{{\lambda} x : \text{Prop.} \, \text{cl} \text{xy} : x} \quad (\forall e)$$

$$\frac{\text{cl} \text{xy} : x}{{\lambda} x : \text{Prop.} \, \text{cl} \text{xy} : x} \quad (\forall e)$$
A deduction in which both of these replacements have been made systematically in all possible places will be called \textit{prepared}. Note that in preparing a deduction, we replace terms in which $\text{cl}$ occurs by terms to which they are $\eta$-convertible.

Now consider in a prepared deduction a subdeduction of the form

\[
\begin{array}{c}
\text{cl}: \text{CL} \quad A : \text{Prop} \\
\hline
\text{cl}A : \neg A \supset A \\
\hline
\text{cl}AM : A
\end{array}
\]

Since there is a subdeduction of $A : \text{Prop}$, $A$ is a type; hence, it is either simple or compound.

The strategy is to follow the idea of [35] for classical logic, and eliminate occurrences of subdeductions like those above in which $A$ is compound. Thus, assume that $A$ is $(\forall y : B)C$. We need a lemma.

\textbf{Lemma 14.} Let $\Gamma$ be a well-formed environment, and suppose that

\[\Gamma \vdash A : X,\]

where $A =* (\forall y : B)C$ and $X =* \kappa$. Then

\[\Gamma \vdash B : \kappa' \quad \text{and} \quad \Gamma, y : B \vdash C : \kappa.\]

\textbf{Proof.} A straightforward induction on the normalized deduction of $\Gamma \vdash A : X$. $\square$

Now by this lemma and $\mathcal{D}_1$ above, it follows that there are deductions

\[\begin{array}{c}
y : B \\
\mathcal{D}_3(y) \\
B : \kappa, \\
C : \text{Prop.}
\end{array}\]

Then we can transform

\[
\begin{array}{c}
\text{cl} : \text{CL} \quad A : \text{Prop} \\
\hline
\text{cl}A : \neg A \supset A \\
\hline
\text{cl}AM : A
\end{array}
\]

into the following:

\[ \vdash \lambda y : B.C : (\forall y : B)\text{Prop}, ~ M : \neg\neg A, \quad [y : B] \]

\[ \ldots \]

where \( \mathcal{D}_5 \) is

\[ \vdash \frac{\mathcal{D}_1, \quad \mathcal{D}_5, \quad \mathcal{D}_2, \quad \mathcal{D}_4}{\mathcal{D}_6(B, \lambda y : B.C,M,y)} \]

\[ \vdash \mathcal{F}(B, \lambda y : B.C,M,y) : \neg\neg C (\forall y) \]

\[ \vdash \frac{\mathcal{D}_3}{\lambda y : B. \mathcal{C}(B, \lambda x : B.C,M,N) : (\forall y : B)C, ~ B : \kappa (\forall \kappa - 1)} \]

\[ \mathcal{F}(u', v', u, v) \]

is

\[ \lambda w : \neg u'.v. u(\lambda y : (\forall x : u')(v'x). w(yv)), \]

and \( \mathcal{D}_6(u', v', u, v) \) is the obvious normalized deduction of

\[ u' : \kappa, v' : (\forall x : u')\text{Prop}, ~ u : \neg\neg(\forall x : u')(v'x), w : u' \vdash \mathcal{F}(u', v', u, v) : \neg\neg u'v. \]

In the special case in which \( A =_* \bot \), we have a special transformation: we replace

\[ \mathcal{D}_v \]

by

\[ \vdash \frac{\mathcal{D}_1, \quad \mathcal{D}_5, \quad \mathcal{D}_2, \quad \mathcal{D}_4}{\mathcal{D}_6(B, \lambda y : B.C,M,y)} \]

\[ \vdash \mathcal{F}(B, \lambda y : B.C,M,y) : \neg\neg C (\forall y) \]

\[ \vdash \frac{\mathcal{D}_3}{\lambda y : B. \mathcal{C}(B, \lambda x : B.C,M,N) : (\forall y : B)C, ~ B : \kappa (\forall \kappa - 1)} \]

If we repeatedly apply these transformations to a deduction, we will eventually reach a point at which in all occurrences of a part of a deduction of the form

\[ \mathcal{D}_1, \quad \mathcal{D}_5, \quad \mathcal{D}_2 \]

\[ \vdash \frac{\mathcal{D}_1, \quad \mathcal{D}_5, \quad \mathcal{D}_2}{\mathcal{D}_6(B, \lambda y : B.C,M,y)} \]

\[ \vdash \mathcal{F}(B, \lambda y : B.C,M,y) : \neg\neg C (\forall y) \]

\[ \vdash \frac{\mathcal{D}_3}{\lambda y : B. \mathcal{C}(B, \lambda x : B.C,M,N) : (\forall y : B)C, ~ B : \kappa (\forall \kappa - 1)} \]

\[ \lambda y : B. clC\mathcal{F}(B, \lambda x : B.C,M,N) \]

\[ A \text{ is a simple type. Of course, this will have replaced terms of the form } clAM \text{ for } \]

\[ \mathcal{D}_1, \quad \mathcal{D}_5, \quad \mathcal{D}_2 \]

\[ \vdash \frac{\mathcal{D}_1, \quad \mathcal{D}_5, \quad \mathcal{D}_2}{\mathcal{D}_6(B, \lambda y : B.C,M,y)} \]

\[ \vdash \mathcal{F}(B, \lambda y : B.C,M,y) : \neg\neg C (\forall y) \]

\[ \vdash \frac{\mathcal{D}_3}{\lambda y : B. \mathcal{C}(B, \lambda x : B.C,M,N) : (\forall y : B)C, ~ B : \kappa (\forall \kappa - 1)} \]

\[ \lambda y : B. clC\mathcal{F}(B, \lambda x : B.C,M,N) \]
and cl ⊢ M by M(\lambda u : \bot . u). If we repeat these replacements, we will eventually eliminate all occurrences of the assumption cl : CL as the major premise for an inference by (\forall e) in which the term of the minor premise is a compound type. We can go on to eliminate all occurrences of this assumption by changing some small simple types B to \neg \neg B; this will convert a deduction of \Gamma, cl : CL ⊢ M : A to a deduction of \Gamma' ⊢ M^* : A', where \Gamma' and A' are obtained from \Gamma and A by replacing some small simple types B by \neg \neg B and changing some of the terms. Note that all the terms so changed have occurrences of cl in them; it follows from the subject-construction property (see [18, Notes 14.18 and 15.12 and Remark 16.37] and [37, p. 301]; it says that a deduction follows the construction of the term) that if a term without an occurrence of cl occurs in a type in \Gamma or in A, then that term is unchanged, and so is any type to which it is proved to belong in the deduction. These terms occurring in the types of \Gamma' or A (whether changed or not) will be called type arguments.

Since it is trivial to prove in constructive logic that A ⊢ \neg \neg A, we can put all this in the form of the following theorem.

**Theorem 22.** If there is a deduction of \Gamma, cl : CL ⊢ M : A, and if \Gamma' and A' are obtained from \Gamma and A by (1) replacing every simple small type B by \neg \neg B provided that B occurs in \Gamma or A but does not occur inside an occurrence of the type \bot or in the type of a type argument in which cl does not occur, and (2) by changing type arguments in which cl does occur, then for some term M^* there is a deduction of \Gamma' ⊢ M^* : A'.

Now suppose that we have a deduction cl : CL ⊢ M : \bot (where \Gamma is empty). Then by the theorem, there is a deduction \mathcal{D} of \bot ⊢ M : \bot. Since there is no such deduction (by the normalization theorem), this gives us a proof of Coquand's consistency result.

**Corollary 22.1.** Classical logic is consistent in the calculus of constructions.

Arithmetic is also consistent with classical logic. To prove this, it is enough to prove that cl : CL, peano1 : (\forall n : N)(\neg \sigma n =_N 0) is consistent.

**Theorem 23.** Let \Gamma be a strongly consistent environment in which all simple types which have universal prefixes are large. Then

\[ \Gamma, cl : CL, peano1 : (\forall n : N)(\neg \sigma n =_N 0) \]

is consistent.

**Proof.** Suppose there is a term M such that

\[ \Gamma, cl : CL, c : (\forall n : N)(\neg \sigma n =_N 0) \vdash M : \text{void}. \]
Then by Theorem 22 there is a term $M'$ such that

$$ \Gamma', c : (\forall n : N) \neg (\forall z : N \rightarrow \text{Prop})(\neg \neg z(\sigma n) \rightarrow \neg \neg z0) \vdash M' : \text{void}. $$

(Recall that $P =_A Q$ converts to $(\forall z : A \rightarrow \text{Prop})(zP \rightarrow zQ)$.) It is not hard to see that $\Gamma'$ is strongly negation consistent. Now in a normal deduction of

$$ \Gamma', c : (\forall n : N) \neg (\forall z : N \rightarrow \text{Prop})(\neg \neg z(\sigma n) \rightarrow \neg \neg z0) \vdash M' : \text{void}, $$

the top of the left branch must be

$$ c : (\forall n : N) \neg (\forall z : N \rightarrow \text{Prop})(\neg \neg z(\sigma n) \rightarrow \neg \neg z0), $$

and the minor premise for an inference by $(Ve)$ is

$$ Q : (\forall z : N \rightarrow \text{Prop})(\neg \neg z(aU) \rightarrow \neg \neg z0), $$

where there is an assumption $U : N$. This is proved impossible by the following lemma. □

Lemma 15. If $\Gamma$ is strongly negation consistent, and if for some term $M$

$$ \Gamma, c : (\forall n : N) \neg (\forall z : N \rightarrow \text{Prop})(\neg \neg z(\sigma n) \rightarrow \neg \neg z0) \vdash M : (\forall z : A \rightarrow \text{Prop})(\neg \neg zR \rightarrow \neg \neg zS), $$

then $R =_* S$.

Proof. Assume that $D$ is a normalized deduction as in the lemma and that there is no proper subdeduction of something of this form whose undischarged assumptions are in some strongly negation consistent environment or are of the given form for $\text{peano1}$. If the last inference in $D$ (Except for $(Eq'')$) is $(Ve)$, then the top of the left branch must be

$$ c : (\forall n : N) \neg (\forall z : N \rightarrow \text{Prop})(\neg \neg z(\sigma n) \rightarrow \neg \neg z0), $$

and then the deduction ending in the minor premise violates the assumptions about $D$. In fact, this shows that

$$ c : (\forall n : N) \neg (\forall z : N \rightarrow \text{Prop})(\neg \neg z(\sigma n) \rightarrow \neg \neg z0) $$

cannot occur anywhere in $D$ at the top of the left branch, and, since $D$ is normalized, this implies that it is not used in the deduction. It follows (by repeating this argument about subdeductions ending in $(Ve)$) that we can decompose $D$ until we have a deduction of

$$ \Gamma, z : A \rightarrow \text{Prop}, u : \neg \neg zR, v : \neg \neg zS \vdash Q : \text{void}. $$

By Theorem 19, this is only possible if $R =_* S$. □
11. Sets and functions

We spoke in Section 9 of the predicate \( \mathcal{N} \) of natural numbers. But most mathematicians prefer to think of the set of natural numbers. This point of view is easily accommodated in the theory of constructions, since it is easy to think of a predicate as a set. This material is based on the work of Huet [20, Ch 12; 21].

Thus, suppose we have some type \( U : \text{Prop} \) or \( U : \text{Type} \). Then we may think of \( U \) as the current universe. Sets over \( U \) are defined to be predicates of type \( U \rightarrow \text{Prop} \). More formally, we may define

\[
\text{Set}_U \equiv U \rightarrow \text{Prop}.
\]

In terms of this definition, \( \mathcal{N} : \text{Set}_U \) and, if \( A : \text{Prop} \), \( \mathcal{L}A : \text{Set}_{\text{List}_A} \). If \( A : \text{Set}_U \), then we define \( x \in A \) to be \( \lambda x : U.E \). The set \( \{ x : U \mid E \} \) is defined to be \( \lambda x : U.E \). Inclusion of set \( A \) in set \( B \) can be defined by

\[
A \subseteq B \equiv (\forall x : U)(x \in A \supset x \in B)
\]

and the corresponding extensional equality by

\[
A =_{\text{ex}} B \equiv A \subseteq B \wedge B \subseteq A.
\]

From this definition, the axiom of extensionality follows immediately:

\[
(\forall A : \text{Set}_U)(\forall B : \text{Set}_U)((\forall x : U)(x \in A \leftrightarrow x \in B) \supset A =_{\text{ex}} B),
\]

where \( A \leftrightarrow B \) is \((A \supset B) \wedge (B \supset A)\). This is false if we replace \( =_{\text{ex}} \) by \( =_{\text{Set}_U} \), since the latter is intensional. Similarly, \( =_U \) is intensional on \( U \). Furthermore, \( =_U \) corresponds to the definition of equality in set theories, since

\[
(\forall A : \text{Set}_U)(x \in A \supset y \in A)
\]

becomes, when abbreviations are removed,

\[
(\forall A : U \rightarrow \text{Prop})(A \supset Ay) =_U x =_U y.
\]

Huet [20, Ch. 12; 21] uses this to define an intensional equality on \( U \).

Many of the usual sets and set operations can be easily defined. For example,

\[
\emptyset = \{ x : U \mid \bot \},
\]

so we have the axiom of the empty set,

\[
\{ x \} = \{ y : U \mid y =_U x \},
\]

\[
A \cap B = \{ x : U \mid x \in A \land x \in B \},
\]

\[
A \cup B = \{ x : U \mid x \in A \lor x \in B \},
\]

from which it follows that

\[
\{ x, y \} =_U \{ z : U \mid z =_U x \lor z =_U y \},
\]

\[
\text{predicate } \mathcal{N} \text{ of natural numbers}
\]

But most mathematicians prefer to think of the set of natural numbers. This point of view is easily accommodated in the theory of constructions, since it is easy to think of a predicate as a set. This material is based on the work of Huet [20, Ch 12; 21].

Thus, suppose we have some type \( U : \text{Prop} \) or \( U : \text{Type} \). Then we may think of \( U \) as the current universe. Sets over \( U \) are defined to be predicates of type \( U \rightarrow \text{Prop} \). More formally, we may define

\[
\text{Set}_U \equiv U \rightarrow \text{Prop}.
\]

In terms of this definition, \( \mathcal{N} : \text{Set}_U \) and, if \( A : \text{Prop} \), \( \mathcal{L}A : \text{Set}_{\text{List}_A} \). If \( A : \text{Set}_U \), then we define \( x \in A \) to be \( \lambda x : U.E \). The set \( \{ x : U \mid E \} \) is defined to be \( \lambda x : U.E \). Inclusion of set \( A \) in set \( B \) can be defined by

\[
A \subseteq B \equiv (\forall x : U)(x \in A \supset x \in B)
\]

and the corresponding extensional equality by

\[
A =_{\text{ex}} B \equiv A \subseteq B \wedge B \subseteq A.
\]

From this definition, the axiom of extensionality follows immediately:

\[
(\forall A : \text{Set}_U)(\forall B : \text{Set}_U)((\forall x : U)(x \in A \leftrightarrow x \in B) \supset A =_{\text{ex}} B),
\]

where \( A \leftrightarrow B \) is \((A \supset B) \wedge (B \supset A)\). This is false if we replace \( =_{\text{ex}} \) by \( =_{\text{Set}_U} \), since the latter is intensional. Similarly, \( =_U \) is intensional on \( U \). Furthermore, \( =_U \) corresponds to the definition of equality in set theories, since

\[
(\forall A : \text{Set}_U)(x \in A \supset y \in A)
\]

becomes, when abbreviations are removed,

\[
(\forall A : U \rightarrow \text{Prop})(A \supset Ay) =_U x =_U y.
\]

Huet [20, Ch. 12; 21] uses this to define an intensional equality on \( U \).

Many of the usual sets and set operations can be easily defined. For example,

\[
\emptyset = \{ x : U \mid \bot \},
\]

so we have the axiom of the empty set,

\[
\{ x \} = \{ y : U \mid y =_U x \},
\]

\[
A \cap B = \{ x : U \mid x \in A \land x \in B \},
\]

\[
A \cup B = \{ x : U \mid x \in A \lor x \in B \},
\]

from which it follows that

\[
\{ x, y \} =_U \{ z : U \mid z =_U x \lor z =_U y \},
\]
which gives us the axiom of pairing, and

\[ \sim A \equiv \{ x : U \mid \sim x \in A \} . \]

When no confusion results, we can leave out \( U \) and write \( \{ x \mid E \}, \text{Set}, \text{etc.} \).

It is important to remember the constructive nature of the logic. This means that the set operations given above are not exactly like those in ordinary mathematics. For example, we have

\[ A \subseteq \sim \sim A, \]

but not, in general, the converse.

One operation on sets that we do not have here is the power set operation. For the power set of \( A \), i.e. the set of all subsets of \( A \), is defined by

\[ \mathcal{P}A \equiv \lambda B : \text{Set}. B \subseteq A, \]

and the type of \( \mathcal{P}A \) is not \( \text{Set} \), which is \( U \to \text{Prop} \), but instead \( \text{Set} \to \text{Prop} \). Terms of type \( \text{Set} \to \text{Prop} \) will be called classes, and we will give the formal definition

\[ \text{Class}_U \equiv \text{Set}_U \to \text{Prop}. \]

Since \( U \) can be replaced by \( \text{Set}_U \), all set operations are also class operations. We can define other class operations, for example,

\[ \bigcap C \equiv \{ x \mid (\forall A : \text{Set})(CA \supset x \in A) \}, \]

and

\[ \bigcup C \equiv \{ x \mid (\exists A : \text{Set})(CA \wedge x \in A) \}. \]

We can also define the singleton in terms of classes:

\[ \{ x \} \equiv \bigcap (\lambda A : \text{Set}. x \in A). \]

With these definitions,

\[ \mathcal{N} : \text{Set}_{\mathcal{N}}. \]

We know metatheoretically that the closed terms which are elements of the set \( \mathcal{N} \) are exactly the closed terms of type \( \mathcal{N} \) (modulo \( \eta \)-conversion). Thus, the set \( \mathcal{N} \) represents the type \( \mathcal{N} \) in a special way. There is no known uniform method of defining sets to represent types for arbitrary types that does not require extra axioms. It is, of course, possible to add an axiom of the form \( \mathcal{A}M \) for each closed term \( M : A \), where \( A \) is a type and \( \mathcal{A} \) is the set intended to represent it, but many of these axioms are likely to upset the proof of strong normalization.

Most mathematicians think of functions as sets of ordered pairs, but this conception is not really appropriate here. For we already have functions built into the theory of constructions as primitive. A function is simply a term assigned to a type of the form
(\forall x : A)B. Functions can, of course, be elements of sets, especially if the sets correspond to types the way \( N \) corresponds to \( \mathbb{N} \). Since a set corresponding to a type \( A \) is a term of type \( A \rightarrow \text{Prop} \), a set of functions from type \( A \) to type \( B \) is a term of type \( (A \rightarrow B) \rightarrow \text{Prop} \). To say that a function \( f : U \rightarrow U \) is a function from set \( A \) to set \( B \), we use the type

\[(\forall x : U)(x \in A \supset f x \in B)\].

It follows that the set of functions from set \( A \) to set \( B \) is

\[\lambda f : U \rightarrow U. (\forall x : U)(x \in A \supset f x \in B)\].

If \( f : U \rightarrow U \), then for \( A : \text{Set} \) we can define

\[\text{Preserve}\, f A \equiv (\forall x : U)(x \in A \supset f x \in A)\].

In terms of this operator, the induction axiom \( \text{Peano} \) can be written as

\[\text{Peano} = \ast (\forall A : \mathbb{N} \rightarrow \text{Prop})((\text{Preserve} \, \sigma A) \supset 0 \in A \supset (\forall n : \mathbb{N})(n \in A))\],

and the definition of \( N \) as

\[N = \ast \lambda n : \mathbb{N}. (\forall A : \mathbb{N} \rightarrow \text{Prop})((\text{Preserve} \, \sigma A) \supset 0 \in A \supset n \in A)\].

This may help to show how to standardize the definition of inductively defined free algebras.

**Remark 17.** It may be interesting to see what axioms of set theory are true in this interpretation. Since the underlying logic is constructive, let us consider constructive set theory. In particular, consider \( \text{IZF} \) as given in Beeson [1, p. 164]. We do not, of course, have a predicate for set which, together with \( N \), satisfies Beeson's axiom A1: \( N(x) \lor S(x) \) or his A4 \((x \in y \supset S(y))\). But his A2 \((- (N(x) \land S(x)))) \) is true metatheoretically, since the types \( N \) and \( \text{Set} \) are always disjoint. Furthermore, the formulas corresponding to axioms A3, A5, and his entire group B are all provable, as we have seen in Section 9.

Let us therefore proceed to group C, the set theoretic axioms. We have already seen that axioms C1 (extensionality), C2 (empty set), and C3 (pairing) are provable, at least with the right equality for C1. For the rest of the list, we have the following:

Axiom C4. Infinity. This depends on \( U \). If \( U \) is \( \mathbb{N} \), then it can be easily proved since \( N \) is the required infinite set.

Axiom C5. Union. We cannot have a set of sets, but we can have a class of sets, and so we can prove the following variant of this axiom:

\[\forall A : \text{Class}(\exists B : \text{Set}) (\forall x : U)(x \in B \rightarrow (\exists A : \text{Set})(A \in A \land x \in A)).\]

The proof comes by letting \( B \) be

\[\lambda x : U. (\exists A : \text{Set})(A \in A \land x \in A)\].
Axiom C6. Separation. We can prove

\((\forall \phi : U \rightarrow \text{Prop})(\forall A : \text{Set})(\exists B : \text{Set})(\forall x : U)(x \in B \leftrightarrow x \in A \land \phi x)\)

by taking \(B \equiv \lambda x : U . x \in A \land \phi x\).

Axiom C7. Power set. We can prove

\((\forall x : \text{Set})(\exists u : \text{Class})(\forall z : \text{Set})(z \in u \leftrightarrow z \subseteq x)\).

This is like our version of the axiom of union above in referring to classes. But unlike
the axiom of union, it seems to lead us out of sets. For this reason, I think we must
regard this axiom as failing here.

Axiom C8. \(\in\)-induction. To state this axiom, we need a predicate \(A\) which applies
to both sets and their elements. For this \(A\), we would need both \(A : U \rightarrow \text{Prop}\) and
\(A : \text{Set} \rightarrow \text{Prop}\), and this is impossible. Thus, we cannot even state this axiom here.

Axiom C9. Collection. We can prove a form of this axiom,

\((\forall \phi : U \rightarrow U \rightarrow \text{Prop})(\forall x : \text{Set})((\forall y : U)(y \in x \supset (\exists z : U)(\phi y z)))
\supset (\exists u : \text{Set})(\forall y : U)(y \in x \supset (\exists z : U)(z \in u \land \phi y z))\),

by taking \(u\) to be

\(\lambda z : U . (\exists y : U)(y \in x \land \phi y z)\).

Note. The idea for checking the axioms of set theory in this way is due to Edward
Belaga. In the summer of 1991, after reading his [2] in which it is suggested that the
power set axiom only holds for countable sets, it occurred to me that because of the
problems in interpreting the power set axiom in the calculus of constructions it might
be worth trying to represent this alternative set theory. I did some preliminary work on
this in [40], but this work is still much too preliminary to publish, and unfortunately
I have not had time to return to this topic since then.

This much set theory is sufficient for most practical mathematical purposes, but from
the point of view of a set theorist it is incomplete. Its major weakness is that if \(A\) is a
set, \(\mathcal{P}A\) is not a set but a class; in the standard set theories it is also a set. To make this
a set, we would need to have \(\text{Set}\) include not only the terms in \(U \rightarrow \text{Prop}\) but also
in \((U \rightarrow \text{Prop}) \rightarrow \text{Prop}, ((U \rightarrow \text{Prop}) \rightarrow \text{Prop}) \rightarrow \text{Prop},\) etc. This can be represented
in the theory of constructions as follows (this is not done in [20] or [21]): first define

\[
\text{Set}_1 \equiv U \rightarrow \text{Prop},
\]

\[
\text{Set}_{n+1} \equiv \text{Set}_n \rightarrow \text{Prop}.
\]

Then we want to introduce a new type \(\text{Set}\) which will be assigned to terms in any of
the types \(\text{Set}_n\). This requires that each type \(\text{Set}_n\) be a subtype of \(\text{Set}\).

There is a general method of making type \(A\) a subtype of type \(B\): it is to take as an assumption

\(\lambda x : A . x : A \rightarrow B\).
From this assumption and $M : A$, we get $(\lambda x : A . x)M : B$, and clearly $(\lambda x : A . x)M$ represents the same object as $M$, in fact, it reduces to $M$. Assumptions of this form have not been considered so far in the theory of constructions, and cannot occur in well-formed environments. However, they have been considered in connection with ordinary type assignment; see [13, pp. 304 and 453], where they are called proper inclusions. Furthermore, conditions under which these assumptions are compatible with the normal form theorem are given in [36, Remark 2, p. 23]. It is possible to extend condition (i) of that Remark to TOC0.

**Theorem 24.** Let $\Gamma$ be a well-formed environment, and let $\Gamma'$ be a sequence of assumptions each of which has the form

$$\lambda x : A . x : A \rightarrow B,$$

where $B$ is an atomic constant, the assumption $B : \kappa$ occurs in $\Gamma$, and $B \rightarrow C$ is not a type in $\Gamma'$ for any type $C$. Then any deduction of

$$\Gamma, \Gamma' \vdash M : A$$

is strongly normalizable and both $M$ and $A$ have normal forms.

**Proof.** We begin by proving that the required deductions are SN. Begin by replacing in each assumption in $\Gamma'$ the term $\lambda x : A . x$ by a variable which does not occur free in either $\Gamma$ or $\Gamma'$, using a distinct variable for each such assumption. The resulting deductions are all SN by Theorem 11. Hence, the deductions in which we are interested, which are all obtained by substituting terms for variables, are also all SN.

Now let us consider the terms in these deductions. These terms may contain redexes of the form

$$(\lambda x : A . x)M.$$ 

A contraction will replace this redex by $M$. What we need to know is that this will not produce a new redex. This could only happen if the original redex occurred in a subterm of the form

$$(\lambda x : A . x)MN_1N_2\ldots N_n,$$

and since the type of

$$(\lambda x : A . x)M$$

is $B$, which is by hypothesis a new constant and hence not convertible to the form $$(\forall y : C)D$$, this is impossible. $\square$

Now, in order to interpret a set theory in which the power set of a set is a set, we need only define $\text{Set}_n$ as indicated above for each $n \geq 1$, define $\text{Set}$ to be a new atomic constant, assume $\text{Set} : \text{Prop}$ or $\text{Set} : \text{Type}$, and then assume

$$\lambda x : \text{Set}_n . x : \text{Set}_n \rightarrow \text{Set}$$
for each $n \geq 1$. (This involves an infinite number of assumptions, but they can all be described in a finite manner, and so it is not unreasonable to suppose that this can be implemented.) It follows from what we have just proved that this is consistent; for $\text{Set}$ is essentially the union of all the $\text{Set}_n$, and in deductions, it will be possible to replace $\text{Set}$ by the union of a finite number of the $\text{Set}_n$ and thus avoid using any new assumptions.

**Remark 18.** Note that with this new definition of $\text{Set}$, certain axioms can no longer be proved in the form stated in Remark 17. This is because we cannot infer from an assumption of the form $x : \text{Set}$ that there is an $n$ with $x : \text{Set}_n$, and it follows that we cannot prove $M \in x : \text{Prop}$ even when we have $M : U$. This affects, in particular, the axioms of union, separation, power set, and collection. However, we can carry out the constructions involved in these axioms metatheoretically: thus, if we have $M : \text{Set}$ for a term $M$, and this comes from $M : \text{Set}_n$, then the term representing the union of $M$ has the right type to be a set. A similar result will hold for separation, power set (and the power set of any set in some $\text{Set}_n$ is a set), and collection. The axiom of $\epsilon$-induction can now be stated, but it will not be possible to prove it.

Note that we are doing all of this set theory without any assumptions. This interpretation is thus a consistency proof of a part of set theory.

**References**