A short and flexible proof of strong normalization for the calculus of constructions

Citation for published version (APA):
Geuvers, J. H. (1994). A short and flexible proof of strong normalization for the calculus of constructions. (Computing science reports; Vol. 9450). Technische Universiteit Eindhoven.

Document status and date:
Published: 01/01/1994

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 31. Aug. 2023
A short and flexible proof of Strong Normalization for the Calculus of Constructions

by

Herman Geuvers

94/50

ISSN 0926-4515

All rights reserved
editors: prof.dr. J.C.M. Baeten
prof.dr. M. Rem

Computing Science Report 94/50
Eindhoven, December 1994
1. Introduction

In the literature there are several different proofs of Strong Normalization (SN) for the Calculus of Constructions (CC). Some of them are of purely syntactical nature (like the ones in [Coquand 1985], [Geuvers and Nederhof 1991] and in [Coquand and Gallier 1990]), while others give a proof of normalization by describing an appropriate semantics [like [Ong and Ritter 1994] and [Altenkirch 1993], who describe an denotational semantics, but also [Goguen 1994], who describes a typed operational semantics). Apart from these, proofs of SN for CC can be found in [Berardi 1988], [Luo 1990] (containing a proof of SN for the 'Extended' Calculus of Constructions), [Terlouw 1993] and [Geuvers 1993] (containing a proof of SN for CC with \( \beta \) and \( \eta \) reduction). Each of these proofs exploits the idea of interpreting types as specific sets of strongly normalizing \( \lambda \)-terms. Then the terms are interpreted in such a way that, (1) if \( t \) is of type \( \sigma \), then the interpretation of \( t \) is in the set associated with \( \sigma \), and (2) for any term \( t \), if its interpretation is SN, then \( t \) itself is SN.

For systems without type dependency (like the polymorphic \( \lambda \) calculus), it is rather well-known by now how to give a proof of SN using so called 'saturated sets' as interpretations for the types. These saturated sets are sets of untyped \( \lambda \) terms that satisfy some specific closure conditions and that are rather easy to work with. A possible drawback of this approach is that the interpretation of the typed term \( t \) should be an untyped term, and hence the interpretation will remove all type information from the term \( t \) (and hence it may remove some redexes). For the polymorphic \( \lambda \) calculus, this is not a real problem, because the reduction that comes from type-abstractions and type-applications can not be the source of an infinite reduction. In a system with type dependency, the situation is rather more complicated, because types can contain terms as subexpressions. (So, if one removes all types, then one also removes some terms.) In the Calculus of Constructions the situation is furthermore complicated by the fact that the system is higher order, which means that there are reductions in type-constructors.

One possible approach to cope with type dependency is to look at sets of typed terms instead of untyped terms. This is done, for example, in [Berardi 1988] and [Coquand and Gallier 1990]. Another possibility is to reduce the question of SN for a system with type dependency to SN for a system without type dependency. This is done in [Geuvers and Nederhof 1991].
approaches lead to rather involved proofs that consist of putting several steps together. Furthermore, these proofs do not easily scale up to larger systems.

The approach that we use here is based on saturated sets. It yields a (relatively short) direct proof of SN for CC using two different interpretations, \([-\!-\!-\!-\!-\!-\!-\!-\!\xi\] and \([-\!-\!-\!-\!-\!-\!-\!-\!\rho\]. The first gives a set or a set-theoretic function for every type, constructor, kind or universe of CC. This is done modulo a valuation function \(\xi\), which assigns a set or set-theoretic function to the constructor variables. (For those not familiar with CC, this terminology is explained below.) The second gives an untyped term for every object, type, constructor or kind of CC. This is done modulo a valuation function \(\rho\), which assigns an untyped term to the constructor variables and the object variables. SN for CC then follows from the fact that

1. if \(\xi\) and \(\rho\) are valuations that 'agree with' the context \(\Gamma\) and \(\Gamma \vdash M : T\), then \([M]_{\rho} \in [T]_{\xi}\)
2. one can choose these valuations \(\xi\) and \(\rho\) in such a way that \([M]_{\rho}\) is SN if and only if \(M\) is SN.

In 3.1 we give some more technical intuition for the proof.

One nice aspect of this approach is that the proof of SN for CC is carried out in exactly the same structure as where the proof of SN for F\(\omega\) is usually done. This again emphasises that the proof of SN for CC is of the same proof-theoretic complexity as the proof of SN for F\(\omega\). (This has already been shown in [Berardi 1988] and [Geuvers and Nederhof 1991].) Furthermore, the proof uses only a minimal part of the meta-theory of CC. This makes it possible to extend the proof of SN for CC to larger systems (with more type operators). This will be shown in Section 4, where we prove SN for CC with \(\Sigma\)-types, \(W\)-types, sum-types and recursive types. For each of these extensions, the proof of SN is a natural generalization of the proof of SN for CC.

2. The Calculus of Constructions

We now give a precise definition of the Calculus of Constructions and at the same time we fix some terminology. In CC there are two specific constants, \(\ast\) and \(\Box\). The first represents the universe of types (so we shall say that \(\sigma\) is a type if \(\sigma : \ast\)) and the second represents the universe of kinds (so we shall say that \(A\) is a kind if \(A : \Box\)). The universe \(\ast\) is a specific example of a kind, so it will be the case that \(\ast : \Box\). To present the derivation rules for CC we first fix the set of pseudoterms from which the derivation rules select the (typable) terms.

2.1. Definition. The set of pseudoterms, \(T\), is defined by

\[ T ::= \ast | \Box | \text{Var} | (\Pi \text{Var}:T.T) | (\lambda \text{Var}:T.T) | TT, \]

where \(\text{Var}\) is a countable set of expressions, called variables. Both \(\Pi\) and \(\lambda\) bind variables and we have the usual notions of free variable and bound variable. The substitution of \(N\) for \(v\) in \(M\) is denoted by \(M[N/v]\). On \(T\) we have the usual notion of \(\beta\)-reduction, denoted by \(\rightarrow_{\beta}\). We also adopt from the untyped \(\lambda\) calculus the conventions of denoting the transitive reflexive closure of \(\rightarrow_{\beta}\) by \(\leftrightarrow_{\beta}\) and the transitive symmetric closure of \(\rightarrow_{\beta}\) by \(\Rightarrow_{\beta}\).

The typing of terms is done under the assumption of specific types for the free variables that occur in the term. These are listed in a context, which is a sequence of declarations \(v_1:T_1, \ldots, v_n:T_n\), where the \(v_i\) are distinct variables and the \(T_i\) are pseudoterms. Contexts are
denoted by the symbol $\Gamma$. For $\Gamma$ a context and $v$ a variable, $v$ is said to be $\Gamma$-$\text{fresh}$ if it is not among the variables that are declared in $\Gamma$.

2.2. DEFINITION. The Calculus of Constructions (CC) is the typed $\lambda$ calculus with the following deduction rules.

\[(\text{ax})\quad \Gamma \vdash \star : \square\]

\[(\text{var})\quad \frac{\Gamma \vdash T : \star / \square \quad \Gamma, v : T \vdash v : T}{\Gamma, v : T \vdash \Gamma, v : T \vdash : T} \quad \text{if } v \text{ is } \Gamma\text{-fresh}\]

\[(\text{weak})\quad \frac{\Gamma \vdash T : \star / \square \quad \Gamma, v : T \vdash M : U}{\Gamma, v : T \vdash \Gamma, v : T \vdash M : U} \quad \text{if } v \text{ is } \Gamma\text{-fresh}\]

\[(\Pi)\quad \frac{\Gamma \vdash T : \star / \square \quad \Gamma, v : T \vdash U : s}{\Gamma \vdash \Pi v : T. U : s} \quad \text{if } s \in \{\star, \square\}\]

\[(\lambda)\quad \frac{\Gamma, v : T \vdash M : U \quad \Gamma \vdash \Pi x : T. U : \star / \square}{\Gamma \vdash \lambda v : T. M : \Pi v : T. U}\]

\[(\text{app})\quad \frac{\Gamma \vdash M : \Pi v : T. U \quad \Gamma \vdash N : T}{\Gamma \vdash MN : U[T/N]}\]

\[(\text{conv})\quad \frac{\Gamma \vdash M : T \quad \Gamma \vdash U : \star / \square}{\Gamma \vdash M : U} \quad T = U\]

The equality in the side condition to the conversion rule (conv) is the $\beta$-equality on the set of pseudoterminals $T$.

The set of terms of CC is defined by $\text{Term} = \{A | \exists \Gamma, B[\Gamma \vdash A : B \lor \Gamma \vdash B : A]\}$.

The set of terms of CC is divided into layers, because, if $M \in \text{Term}$, then one of the following six situations occurs:

1. $M \equiv \square$
2. $M \equiv \star$
3. $\Gamma \vdash M : \square$
4. $\Gamma \vdash M : \star$
5. $\Gamma \vdash M : T$ with $\Gamma \vdash T : \square$
6. $\Gamma \vdash M : T$ with $\Gamma \vdash T : \star$ (for some $\Gamma$ and $T$).

Of course, (2) is a special case of (3) and (4) is a special case of (5), so we can restrict to four cases. It is well-known that these cases are disjoint if we are slightly more careful with the presentation of the syntax. Hence the following definition is useful.
2.3. DEFINITION. 1. The set of *kinds* is defined by $\text{Kind} := \{ A \mid \exists \Gamma[\Gamma \vdash A : \square] \}$.

2. The set of *types* is defined by $\text{Type} := \{ A \mid \exists \Gamma[\Gamma \vdash A : \star] \}$.

3. The set of *constructors* is defined by $\text{Constr} := \{ P \mid \exists A, \Gamma[\Gamma \vdash P : A : \square] \}$.

4. The set of *objects* is defined by $\text{Obj} := \{ P \mid \exists A, \Gamma[\Gamma \vdash P : A : \star] \}$.

Here $\Gamma \vdash P : A : \star$ denotes the fact that $\Gamma \vdash P : A$ and $\Gamma \vdash A : \star$.

For convenience we devide the set of variables $\text{Var}$ in two disjoint sets $\text{Var}^*$ and $\text{Var}^\gamma$. We use $x, y$ and $z$ to denote variables of the first type, also called *object variables*, and we use $\alpha, \beta$ and $\gamma$ to denote variables of the second type, also called *constructor variables*. In the (var) and (weak) rules we now make the restriction that, if $\Gamma \vdash T : \star$, then the new variable has to be taken from the set $\text{Var}^*$ and if $\Gamma \vdash T : \square$, then the new variable has to be taken from the set $\text{Var}^\gamma$.

The usefulness of this definition is due to the following lemma. (For a detailed proof see [Geuvers 1993].)

2.4. LEMMA (Classification). In CC we have

\[
\text{Kind} \cap \text{Type} = \emptyset,
\]

\[
\text{Constr} \cap \text{Obj} = \emptyset.
\]

The Lemma implies that when we define a property for terms of CC by induction on the structure we can distinguish cases according to whether a specific subterm is a kind or type, respectively a constructor or object.

3. Strong Normalization for the Calculus of Constructions

3.1. Intuition for the proof

Before giving the technical details we want to give some (technical) intuition for the proof. In order to do that we first look at the situation for $F\omega$. In that case one defines mappings $\mathcal{V} : \text{Kind} \rightarrow \text{Set}$, $[-]_\xi : \text{Constr} \rightarrow \text{Set}$, and $([-])_\rho : \text{Obj} \rightarrow \Lambda$. Here, $\xi$ is a valuation of constructor-variables and $\rho$ is a valuation of object-variables. These mappings are such that, if $\xi, \rho$ form a *valuation of* $\Gamma$ (this notion will be defined in detail later), then

\[
\Gamma \vdash P : A ; \Box \quad \Rightarrow \quad [P]_\xi \in \mathcal{V}(A),
\]

\[
\Gamma \vdash t : \sigma ; \star \quad \Rightarrow \quad ([t])_\rho \in [\sigma]_\xi.
\]

Furthermore, $\rho$ can always be chosen in such a way that

\[
[M]_\rho \text{ is SN iff } M \text{ is SN}.
\]

The situation is represented in the first picture on the next page. Here $\text{SAT}$ denotes the set of saturated sets and $(\text{SAT})^* \text{ denotes the union of the function spaces built from } \text{SAT}$, so $(\text{SAT})^* := \bigcup \{ \text{SAT}, \text{SAT} \rightarrow \text{SAT}, (\text{SAT} \rightarrow \text{SAT}) \rightarrow \text{SAT}, \text{SAT} \rightarrow \text{SAT} \rightarrow \text{SAT}, \ldots \}$, where the arrow denotes set-theoretic function space.

This construction will only prove SN for the objects of $F\omega$, and it requires some further tricks to show that this implies SN for all terms of $F\omega$. For CC the situation is more complicated,
because constructors and kinds can also contain objects as subterms. So, even if one would have constructed mappings $V$, $\llbracket \_ \rrbracket_\xi$ and $\llbracket \_ \rrbracket_\rho$ as above, it is not so easy to see how SN for the objects of CC implies SN for the full CC.

The solution that we propose here is to define the mapping $\llbracket \_ \rrbracket_\rho$ for all terms of CC. To show that the image of $\llbracket \_ \rrbracket_\rho$ is a strongly normalizing term, we also have to extend the mapping $\llbracket \_ \rrbracket_\xi$ to kinds. So, the kinds of CC will have two interpretations: first as sets under $V$ ($\ast$ is interpreted as SAT and the other kinds are interpreted by appropriate elements of $V := \{ \text{SAT}, \text{SAT} \to \text{SAT}, (\text{SAT} \to \text{SAT}) \to \text{SAT}, \text{SAT} \to \text{SAT} \to \text{SAT}, \ldots \}$, where the arrow denotes set-theoretic function space $(\text{SAT})^*$), second as saturated sets (elements of SAT). This is done to allow an interpretation of constructors as untyped $\lambda$ terms under $\llbracket \_ \rrbracket_\rho$, making sure that
the constructors are strongly normalizing as well. The new situation is visualized in the second picture.

3.2. The proof

Let in the following \( SN \subseteq \Lambda \) be the set of untyped lambda terms that is Strongly Normalizing under \( \beta \)-reduction. For reasons of presentation we extend the untyped \( \lambda \) with one specific constant \( d \), for which there are no special reduction rules. The well-known notion of 'saturated set of \( \lambda \)-terms' is defined in a slightly more general way than is necessary. This is done to make it easier to extend the proof of SN later.

3.1. Definition. The set of base terms \( B \) is defined by

1. \( \text{Var} \subseteq B \) and \( d \in B \),
2. If \( M \in B \) and \( N \in SN \), then \( MN \in B \).

Notice that all base terms are SN.

3.2. Definition. The key redex of an untyped lambda term is defined by

1. If \( M \) is a redex, then \( M \) is its own key redex,
2. If \( M \) has key redex \( N \), then \( MP \) has key redex \( N \).

The term that is obtained from \( M \) by contracting its key redex is denoted by \( \text{red}_k(M) \).

Notice that the key redex of \( M \) is unique, if it exists. Furthermore, every key redex is a head redex (but not the other way around).

3.3. Definition. A set of untyped lambda terms \( X \) is saturated if

1. \( X \subseteq SN \),
2. \( B \subseteq X \),
3. If \( \text{red}_k(M) \in X \) and \( M \in SN \), then \( M \in X \).

The collection of saturated sets is denoted by \( \text{SAT} \).

This definition of saturated set is equivalent to saying that \( X \) is saturated if

1. \( X \subseteq SN \),
2. \( \forall Q \in SN \forall x \in \text{Var} \{xQ \in X\} \),
3. \( \forall Q \in SN [dQ \in X] \),
4. \( \forall Q, M, P \in SN [M[P/x]Q \in X \Rightarrow (\lambda x.M)PQ \in X] \).
By definition, \( SN \) is itself saturated and all saturated sets are nonempty. Furthermore, if \( N \in SN \) and \( M \in X \in SAT \), then \( KMN \in X \), where \( K \) is the well-known \( K \)-combinator \( \lambda xy.x \).

As we already pointed out, the types of \( CC \) will be interpreted as saturated sets. This requires some closure properties for the set of saturated sets which will be proved in Lemma 3.5. The set-interpretation of the kinds of \( CC \) (by the map \( V \)) can be seen as first taking the underlying \( Fw \)-kind (which is a kind that consists of just the symbols \( \to \) and \( * \)), and then taking the set-interpretation of kinds of \( Fw \). Here we define the set-interpretation of \( CC \)-kinds immediately.

3.4. DEFINITION. For \( A \in \text{Kind}(CC) \), the set-interpretation of \( A \), \( V(A) \), is defined inductively as follows.

\[
V(*) = SAT (= \{ X \mid X \subset A \text{ is saturated} \})
\]

\[
V(\Pi \alpha : B.C) = \{ f \mid f : V(B) \to V(C) \}, \text{ if } B : \Box,
\]

\[
V(\Pi x : u.C) = V(C) \text{ if } u : *.
\]

The collection of all set-interpretations is denoted by \( (SAT)^* \), so \( (SAT)^* := \cup \{ V(A) \mid A \in \text{Kind}(CC) \} \).

The types are interpreted as saturated sets and the kinds also have a second interpretation as saturated sets. We need the following (well-known) closure properties on \( SAT \).

3.5. LEMMA. The set of saturated sets \( (SAT) \) is closed under arbitrary intersections and function spaces. That is,

1. for \( I \) a set and \( X_i \) saturated for all \( i \in I \),

\[
\cap_{i \in I} X_i \text{ is saturated}
\]

2. for \( X \) and \( Y \) saturated,

\[
X \to Y := \{ M \in A \mid \forall N \in X \exists MN \in Y \} \text{ is saturated}
\]

3.6. DEFINITION. For \( \Gamma \) a context of \( CC \), a constructor valuation of \( \Gamma \) is a map \( \xi : \text{Var}^\Box \to (SAT)^* \) (notation \( \xi =^\Box \Gamma \)) such that

\[
\alpha : A \in \Gamma \implies \xi(\alpha) \in V(A).
\]

3.7. DEFINITION. For \( \Gamma \) a context of \( CC \) and \( \xi \) a constructor valuation of \( \Gamma \), the interpretation function

\[
[-]_\xi : \Gamma \text{-Term}(CC) \setminus \Gamma \text{-Obj}(CC) \to (SAT)^*
\]

is defined inductively as follows.

\[
[\Box]_\xi = \[\square]_\xi = SN,
\]

\[
[\alpha]_\xi = \xi(\alpha),
\]

\[
[PQ]_\xi = [P]_\xi ([Q]_\xi), \text{ if } Q \text{ is a constructor},
\]

\[
[Pt]_\xi = [P]_\xi, \text{ if } t \text{ is an object},
\]

\[
[\lambda \alpha : A.Q]_\xi = \lambda a \in V(A).Q_{[\xi(\alpha := a)]}, \text{ if } A \text{ is a kind},
\]

\[
[\lambda x : \sigma.Q]_\xi = [Q]_\xi, \text{ if } \sigma \text{ is a type},
\]

\[
[\Pi x : \sigma.T]_\xi = [\sigma]_\xi \to [T]_\xi, \text{ if } \sigma \text{ is a type},
\]

\[
[\Pi \alpha : A.T]_\xi = [A]_\xi \cap_{a \in V(A)} [T]_{[\xi(\alpha := a)]}, \text{ if } A \text{ is a kind}.
\]
It is easy to verify the substitution property for $[-]_\xi$. From it one concludes that $[-]_\xi$ preserves equality:

3.8. FACT. Let $\xi \models \bigwedge \Gamma$ and let $P$ be a constructor, $t$ an object, and $Q$ a constructor or a kind in $\Gamma$. Then

1. $[Q_P]_{\xi} = [Q]_{\xi(\alpha = [P]_\xi)}$,

2. $[Q_t]_{\xi} = [Q]_{\xi}$,

3. $Q =_\beta \bar{P} \iff [Q]_{\xi} = [P]_{\xi}$.

The following Lemma states that the interpretations of the constructors under $[-]_\xi$ are elements of the right set.

3.9. LEMMA (Soundness for $[-]_\xi$). For $\Gamma$ a context of CC, $Q, A \in \text{Term}(\text{CC})$ and $\xi \models \bigwedge \Gamma$,

$\Gamma \vdash Q : A(\Box) \implies [Q]_{\xi} \in \nu(A)$,

$\Gamma \vdash A : \Box \implies [A]_{\xi} \in \text{SAT}$.

PROOF. By simultaneous induction on the structure of $Q$, respectively $A$. □

3.10. DEFINITION. For $\Gamma$ a context of CC and $\xi \models \bigwedge \Gamma$, an object valuation of $\Gamma$ with respect to $\xi$ is a map $\rho : \text{Var} \rightarrow \Lambda$ (notation $\rho, \xi \models \Gamma$) such that

$\alpha : A \in \Gamma \implies \rho(\alpha) \in [A]_{\xi}$,

$x : \sigma \in \Gamma \implies \rho(x) \in [\sigma]_{\xi}$.

3.11. DEFINITION. For $\Gamma$ a context of CC with $\rho, \xi \models \Gamma$, the interpretation function

$(-)_\rho : \Gamma \cdot \text{Term}(\text{CC}) \setminus \{\Box\} \rightarrow \Lambda$

is defined inductively as follows.

$[\text{var}]]_\rho = d$, 

$[v]]_\rho = \rho(v)$,

$[MN]]_\rho = ([M]]_\rho([N]]_\rho$,

$[\lambda v : T.M]]_\rho = K(v,(M)]_{\rho(v := v)}([T]]_\rho$, 

$[\Pi v : T.U]]_\rho = d([T]]_\rho(\lambda v,(U)]_{\rho(v := v)}).

In this definition, $v$ is either an object variable or a constructor variable, $T$ and $U$ are either types or kinds and $M$ and $N$ are either objects or constructors. The term $K$ is the combinator $\lambda xy.x$ and $d$ is the extra constant that has been added to $\Lambda$.

The interpretation of terms (by $[-]_\rho$) does not depend on the interpretation of the constructors and kinds (by $[-]_\xi$). This is also expressed by the following fact.

3.12. FACT. For $M$ a term, $\rho$ a valuation as in the definition and $\vec{x}$ the vector of free variables in $M$, the interpretation $[M]_\rho$ can equivalently be defined by taking

$[M]_\rho := \Gamma : \neg [\rho(\vec{x})/\vec{x}]$. 

8
where \( \rho(\overline{x}) \) is the vector obtained by consecutively applying \( \rho \) to \( \overline{x} \) and \( T \) is inductively defined by

\[
\begin{align*}
\tau^v & := v, \text{ for } v \text{ a variable,} \\
\tau^{MN} & := \tau^M \tau^N, \\
\tau^{\lambda v:T.M} & := \text{К}(\lambda v.\tau^M) T, \\
\tau^{\Pi v:T.U} & := d^T(\lambda v.\tau^U).
\end{align*}
\]

3.13. DEFINITION. For \( \Gamma \) a context and \( M \) and \( T \) terms of \( \text{CC} \), we say that \( \Gamma \) satisfies that \( M \) is of type \( T \), notation \( \Gamma \models M : T \) if

\[ \forall \rho, \xi[\rho, \xi \models \Gamma \implies ([M]_\rho) \in [T]_\xi]. \]

3.14. THEOREM (Soundness Theorem). For \( \Gamma \) a context and \( M \) and \( T \) terms of \( \text{CC} \),

\[ \Gamma \vdash M : T \implies \Gamma \models M : T. \]

PROOF. By induction on the structure of \( M \) we prove that if \( \rho, \xi \models \Gamma \), then \( QMD p \in [T]_\xi \).

- \( M \equiv v \in \text{Var} \). Then \( v:U \in \Gamma \) with \( U = \beta T \). So \( ([M]_\rho) = \rho(v) \in [U]_\xi \) and \( [U]_\xi = [T]_\xi \) and we are done.

- \( M \equiv \lambda x:\tau.Q \) with \( \tau \) a type and \( Q \) a constructor. Then \( \Gamma, x:\tau \vdash Q : B \) for some \( B \) with \( T = \beta \Pi x:\tau.B \). By IH \( ([Q]_{p(x:=p)}) \in [B]_\xi \) for all \( p \in [\tau]_\xi \), so \( \lambda x.([Q]_{p(x:=x)}) \in [\tau]_\xi \rightarrow [B]_\xi \). By IH we also find that \( ([\tau]_\rho) \in [\tau]_\xi \) (and hence \( [\tau]_\rho \in \text{SN} \)). But then \( ([\lambda x:\tau.Q]_\rho = \text{К}(\lambda x.([Q]_{p(x:=x)}))([\tau]_\rho) \in [\tau]_\xi ; [B]_\xi = [T]_\xi \).

- \( M \equiv \lambda \alpha:B.t \), with \( B \) a kind and \( t \) an object. Then \( \Gamma, \alpha:B \vdash t : \tau \) for some \( \tau \) with \( T = \beta \Pi \alpha:B.\tau \). By IH we find that \( ([t]_{p(\alpha:=p)}) \in [\tau]_{\xi(\alpha:=\xi)} \) for all \( f \in \text{V}(B) \) and all \( p \in [B]_\xi \), so \( ([t]_{p(\alpha:=x)}) \in \cap f \in \text{V}(B)[\tau]_{\xi(\alpha:=f)} \) for all \( p \in [B]_\xi \). By IH we also find that \( ([B]_\rho) \in [\square]_\xi \) (and hence \( [B]_\rho \in \text{SN} \)). But then \( ([\lambda \alpha:B.t]_\rho = \text{К}(\lambda \alpha.([t]_{p(\alpha:=x)}))([B]_\rho) \in [B]_\xi ; [\square]_\xi \cap f \in \text{V}(B)[\tau]_{\xi(\alpha:=f)} = [T]_\xi \).

- \( M \equiv t_q \), with \( t \) and \( q \) objects. Then \( \Gamma \vdash t : \Pi x:\sigma.Q \) and \( \Gamma \vdash q : \tau \) for some \( \tau \) and \( \sigma \) with \( \sigma[q/x] = \beta T \). By IH \( ([t]_\rho) \in [\Pi]_\xi \cap [\sigma]_\xi \) and \( ([q]_\rho) \in [\tau]_\xi \), so \( ([tq]_\rho) = ([t]_\rho([q]_\rho) \in [\sigma]_\xi = [T]_\xi \). (Note that \( [\sigma]_\xi = [\sigma[q/x]]_\xi \), due to Fact 3.8.)

- \( M \equiv PQ \), with \( P \) and \( Q \) constructors. Then \( \Gamma \vdash P : \Pi \alpha:A.B \) and \( \Gamma \vdash Q : A \) for some \( B \) with \( B[Q/\alpha] = \beta T \). By IH \( ([P]_\rho) \in [A]_\xi \cap \cap f \in \text{V}(A)[B]_{\xi(\alpha:=f)} \) and \( ([Q]_\rho) \in [A]_\xi \), so \( ([P]_{p(\alpha:=x)})([Q]_\rho) \in \cap f \in \text{V}(A)[B]_{\xi(\alpha:=f)} \). Furthermore, \( ([Q]_\xi) \in \text{V}(A) \), so \( \cap f \in \text{V}(A)[B]_{\xi(\alpha:=f)} \subset [B]_{\xi(\alpha:=f)} \). We conclude that \( ([PQ]_\rho) \in [B]_{\xi(\alpha:=f)} = [T]_\xi \).

- \( M \equiv \Pi x:\alpha.B \), with \( \sigma \) a type and \( B \) a kind. Then \( \Gamma \vdash \sigma : \ast, \Gamma, x:\alpha \vdash B : \square \) and \( T = \square \). By IH \( ([\sigma]_\rho) \in [\square]_\xi \) and \( ([B]_{p(\alpha:=p)}) \in [\square]_\xi \) for all \( p \in [\sigma]_\xi \). Hence \( ([\sigma]_\rho) \in \text{SN} \) and \( \lambda x.([B]_{p(\alpha:=x)}) \in \text{SN} \), so \( ([\Pi x:\alpha.B]_\rho = d([\sigma]_\rho)(\lambda x.([B]_{p(\alpha:=x)})) \in [\square]_\xi \). □

3.15. THEOREM.

\[ \forall M \in \text{Term}(\text{CC})[\text{SN}(M)]. \]
PROOF. Let $M$ be a term of CC. Then either $M \equiv \Box$ or $\Gamma \vdash M : T$ for some $\Gamma$ and $T$. In the first case, $M$ is of course SN. In the second case, $\Gamma \vdash M : T$ by the previous theorem. Now we define canonical elements $c^A$ in the sets $\mathcal{V}(A)$ (for $A \in \text{Kind}(CC)$) as follows.

$$
c^* := \text{SN},
$$

$$
c^{\Pi_a:A.B} := \lambda f \in \mathcal{V}(A).c^B \text{, if } A: \Box
$$

$$
c^{\Sigma.a.B} := c^B \text{, if } a: \star.
$$

For the constructor valuation for $\Gamma$ we take $\xi$ with $\xi(\alpha) = c^A$ if $\alpha : A \in \Gamma$ (and $\xi(\alpha)$ arbitrary otherwise), and for the object valuation for $\Gamma$ with respect to this $\xi$ we take $\rho$ with $\rho(v) = v$.

Now $\rho, \xi \models \Gamma$ and so $[M]_\rho \in [T]_\xi$, where $[M]_\rho$ is just $\Gamma M^\rho$ (due to Fact 3.12). This means that $\Gamma M^\rho$ is SN, because $[T]_\xi \subseteq \text{SN}$. It is easy to see that $M$ is SN if and only if $\Gamma M^\rho$ is SN. Hence $M$ is SN. □

4. Beyond CC

The above proof of SN for CC is very flexible and can be extended to many other cases. The main cause for this flexibility is that the proof does not rely on too much (difficult) meta theory of CC. For one thing, we don’t require the set of typable terms to be closed under reduction (the so called Subject Reduction property). The only property that is seriously used (although we did not mention it explicitly) is the fact that if two CC-terms $M$ and $N$ (of the same type in the same context) are convertible, then they are convertible via a path through the well-typed terms. In [Geuvers and Werner 1994] this property is called the ‘soundness’ of the system, because it implies the equivalence of the presentation of CC with a typed conversion rule with the presentation in Definition 2.2, in which the conversion is untyped. Soundness is a very desirable property because it conforms with our intuition about typed $\lambda$-calculi that, if two types are convertible as pseudoterms (and hence the sets of terms of these types are the same), then there should be a reduction-expansion path from one type to the other via the well-typed terms. In the proof of SN for CC the soundness has been used in the proof of Fact 3.8 (iii). For our presentation of CC above, the soundness is an immediate consequence of the Church-Rosser property (for $\beta$) and Subject Reduction (for $\beta$), so we have no problem. However, a problem may arise if we want to add new type-constructors with new reduction rules for which the Church-Rosser property is unknown. A simple general solution to avoid all difficulties is to replace the conversion rule by a more fine-grained one.

4.1. DEFINITION. In the following, the conversion rule (conv) will not be the one in Definition 2.2, but the following.

$$
\frac{\Gamma \vdash M : T \quad \Gamma \vdash U : \star/\Box}{\Gamma \vdash M : U} \quad \text{if } U \longrightarrow T \text{ or } T \longrightarrow U
$$

Here $\longrightarrow$ is a one-step-reduction. (In Section 3 this would be $\longrightarrow^{\beta}$.)

The advantage of this slightly different conversion rule is that, in order to show the soundness of the (conv) rule in the proof of Theorem 3.14, one only needs that $Q \longrightarrow P \Rightarrow [Q]_\xi = [P]_\xi$, for $Q$ and $P$ typable.

We treat four examples of extensions of CC and show that they are SN by adapting the proof of Section 3. The extensions that we treat are the ones with $W$-types (for representing types
of well-founded trees), sum-types recursive types and Σ-types. Before studying these examples we list some general properties about saturated sets that will be used. These properties are proved for the saturated set notion as it has been given in the previous paragraph. For each extension of CC that is treated hereafter, the notion of saturated set is slightly adapted, but the proofs of these properties will still go through.

4.1. Saturated sets

Saturated sets are sets of untyped λ terms that contain all so-called 'base terms' and are closed under expanding a key redex. We define the notion of key reduction separately.

4.2. DEFINITION. For M and N λ terms, we say that M key-reduces to N, notation $M \rightarrow^k N$, if N is obtained from M by contracting the key redex in M. The transitive reflexive closure of $\rightarrow^k$ is denoted by $\rightarrow^*$. An easy fact about key reduction is the following.

4.3. FACT. If $X$ is a saturated set and $N \in X$ with $M \rightarrow^k N$ and $M \in SN$, then $M \in X$.

We have already seen two constructions that can be performed on saturated sets, namely the function space construction and the intersection. There are many more of those, some of which will be defined and used later. An important trivial fact about SAT is the following.

4.4. FACT. SAT is a complete lattice. The ordering is the inclusion and suprema and infima are given by union and intersection, respectively.

4.5. REMARK. The function space construction on SAT:

$$X \rightarrow Y := \{ M \mid \forall N \in X[MN \in Y]\},$$

does NOT satisfy

$$X \cap Y \subseteq Z \Rightarrow X \subseteq Y \rightarrow Z.$$

So, $(SAT, \cup, \cap, \rightarrow)$ is not a Heyting algebra and SAT (with $\cap$ and $\rightarrow$ representing universal quantification and implication, respectively) is not an algebraic model of second or higher order propositional logic. (See, for example, [Geuvers 1994].) Of course, SAT can be made into a Heyting algebra by just taking for the arrow between $X$ and $Y$ the set $\cap\{ Z \mid X \subseteq Y \rightarrow Z\}$, but for the strong normalization proof this choice is not useful.

In spite of the previous Remark, the function space construction on SAT enjoys many nice properties.

4.6. DEFINITION. A morphism from SAT to SAT is an expression $\Phi(X)$ built up from variables ranging over SAT (among which $X$ is one), arrows and intersections.

A morphism $\Phi(X)$ is positive if $X$ occurs only to the left of an even number of arrows. A morphism $\Phi(X)$ is negative if $X$ occurs only to the left of an odd number of arrows.

In Definition 4.6 we allow arbitrary intersections, so if $\Phi_i(X)$ is a morphism for every $i \in I$, then $\Phi(X) = \cap_{i \in I} \Phi_i(X)$ is also a morphism. This morphism is positive (resp. negative) if $\Phi_i(X)$ is positive (resp. negative) for every $i \in I$. 

11
4.7. **Lemma.** If $\Phi(X)$ is a positive morphism, then $\lambda X.\Phi(X)$ is monotone increasing (if $Y \subseteq Z \implies \Phi(Y) \subseteq \Phi(Z)$) and if $\Phi(X)$ is a negative morphism, then $\lambda X.\Phi(X)$ is monotone decreasing ($Y \subseteq Z \implies \Phi(Z) \subseteq \Phi(Y)$).

**Proof.** By induction on the structure of $\Phi(X)$. □

The following is an immediate consequence of the fact that a positive morphism is a monotone increasing function on the complete lattice of saturated sets.

4.8. **Corollary.** If $\Phi(X)$ is a positive morphism on SAT, then there is a smallest saturated set $\text{lfp}(\Phi)$ for which $\Phi(\text{lfp}(\Phi)) = \text{lfp}(\Phi)$.

4.2. **CC with W-types**

We now look at the extension of CC with Martin-Löf's W-types, a type constructor for representing types of well-founded trees. (See [Martin-Löf 1984] or [Nordström et al. 1990] for an extensive treatment of W-types and examples.) We just give the rules for W-types and the proof that the addition of these rules to CC preserves the SN property.

4.9. **Definition.** The *Calculus of Constructions with W-types*, CC$^W$, has the following additional rules.

\begin{align*}
\text{(W)} & \quad \Gamma \vdash \sigma : \star \quad \Gamma, x : \sigma \vdash \tau : \star \\
& \quad \Gamma \vdash Wx : \sigma.\tau : \star \\
\text{(sup)} & \quad \Gamma \vdash p : \sigma \quad \Gamma \vdash q : \tau[p/x] \rightarrow Wx : \sigma.\tau \\
& \quad \Gamma \vdash \text{sup}(p, q) : Wx : \sigma.\tau \\
\text{(wrec)} & \quad \Gamma \vdash Q : (Wx : \sigma.\tau) \rightarrow \star \\
& \quad \Gamma \vdash t : \Pi x : \sigma.\Pi z : \tau \rightarrow Wx : \sigma.\tau.\Pi y : \tau.\Pi Q(zy) \rightarrow Q(\text{sup}(x, z)) \\
& \quad \Gamma \vdash \text{wrec } t : \Pi w : (Wx : \sigma.\tau).Qw
\end{align*}

The reduction rule associated with wrec and sup(−, −) is

\[\text{wrec } t(\text{sup}(p, q)) \rightarrow_w t p q(\lambda y : \tau[p/x].\text{wrec } t(q y)).\]

The conversion rule is adapted to this new reduction.

Now, we extend the untyped $\lambda$ calculus with wrec and sup(−, −) operators that have the reduction behaviour

\[\text{wrec } P(\text{sup}(N, Q)) \rightarrow_w P N Q(\lambda y . \text{wrec } P(Q y)).\]

The definition of the set of base terms $B$ is adapted by adding to Definition 3.1 the clause

3. If $M \in B$ and $P \in \text{SN}$, then $\text{wrec } P M \in B$.

The notion of key redex is extended by adding to Definition 3.2 the clause

3. If $M$ has key redex $N$, then $\text{wrec } P M$ has key redex $N$ (for any $P$).

The definition of saturated set is the same as in Definition 3.3, with the notions of 'base term' and 'key redex' replaced by the above ones. This new collection of saturated sets is ambiguously denoted by SAT (but there will be no confusion).
4.10. DEFINITION. For $X, Y \in \text{SAT}$, the saturated set $W(X, Y)$ is defined by

$$W(X, Y) := \text{ifp}(\lambda W.\{ M | \forall Z \forall P \in X \rightarrow (Y \rightarrow W) \rightarrow (Y \rightarrow Z) \rightarrow Z[\text{wrec } PM \in Z]\}).$$

(Ifp denotes the least fixed point.)

That this least fixed point exists is due to the fact that

$$\lambda W.\{ M | \forall Z \forall P \in X \rightarrow (Y \rightarrow W) \rightarrow (Y \rightarrow Z) \rightarrow Z[\text{wrec } PM \in Z]\}$$

is a monotone function on SAT. This can be seen as follows.

Write $\Phi(W)$ for $\{ M | \forall Z \forall P \in X \rightarrow (Y \rightarrow W) \rightarrow (Y \rightarrow Z) \rightarrow Z[\text{wrec } PM \in Z]\}$ and let $W, W' \in \text{SAT}$, with $W \subseteq W'$. Let $M \in \Phi(W)$ Then, for all $Z$ and for all $P \in X \rightarrow (Y \rightarrow W) \rightarrow (Y \rightarrow Z) \rightarrow Z$, we have $\text{wrec } PM \in Z$. Now, $W$ is negative in $X \rightarrow (Y \rightarrow W) \rightarrow (Y \rightarrow Z) \rightarrow Z$, so $\forall Z \forall P \in X \rightarrow (Y \rightarrow W') \rightarrow (Y \rightarrow Z) \rightarrow Z[\text{wrec } PM \in Z]$ and so $M \in \Phi(W')$.

The set $W(X, Y)$ can equivalently be defined as

$$\cap\{W \mid \text{wrec } \in \text{SAT}(X \rightarrow (Y \rightarrow W) \rightarrow (Y \rightarrow Z) \rightarrow Z) \rightarrow W \rightarrow Z\}.\]$$

The essential closure properties for the $W$-constructor on SAT are the following.

4.11. LEMMA. Let $X$ and $Y$ be saturated sets and write $W$ for $W(X, Y)$.

1. If $M \in X$ and $N \in Y \rightarrow W$, then $\sup(M, N) \in W$.
2. If $P \in X \rightarrow (Y \rightarrow W) \rightarrow (Y \rightarrow Z) \rightarrow Z$, then $\text{wrec } P \in W \rightarrow Z$.

PROOF. We use the fact that

$$W = \{ M | \forall Z \forall P \in X \rightarrow (Y \rightarrow W) \rightarrow (Y \rightarrow Z) \rightarrow Z[\text{wrec } PM \in Z]\}.$$

For the first, let $Z \in \text{SAT}$ and $P \in X \rightarrow (Y \rightarrow W) \rightarrow (Y \rightarrow Z) \rightarrow Z$. Then $\text{wrec } P(\sup(M, N)) \supseteq \text{PMN}(\lambda y.\text{wrec } P(Ny) \in Z$ and $\text{wrec } P(\sup(M, N))$ is SN, so $\text{wrec } P(\sup(M, N)) \in Z$ and hence $\sup(M, N) \in W$. For the second, let $M \in W$. Then $\text{wrec } PM \in Z$ by definition, so $\text{wrec } P \in W \rightarrow Z$.

The definition of set-interpretation of 3.4 does not have to be extended, because there are no kinds of the form $W X: 17. T$. The notion of $\sim FD r'$ is defined analoguously to Definition 3.6.

4.12. DEFINITION. The function $[-]_\xi$ is defined by extending Definition 3.7 with the following clause.

$$[W x: \sigma. \tau]_\xi = W([\sigma]_\xi, [\tau]_\xi).$$

We have the following property for the extended $[-]_\xi$.

4.13. FACT. Let $Q$ and $P$ be constructors or kinds with $\Gamma \vdash Q, P : T$ and $\xi$ a valuation with $\xi \models \Gamma$, then

$$Q \rightarrow_\beta \gamma P \Rightarrow [Q]_\xi = [P]_\xi.$$

The Soundness Lemma 3.9 is also easily verified:

4.14. LEMMA (Soundness for $[-]_\xi$). For $\Gamma$ a context of $\text{CC}^W, Q, A \in \text{Term}(\text{CC}^W)$ and $\xi \models \Gamma$,

\[
\begin{align*}
\Gamma \vdash Q : A(: \Box) & \Rightarrow [Q]_\xi \in V(A), \\
\Gamma \vdash Q : \Box & \Rightarrow [Q]_\xi \in \text{SAT}.
\end{align*}
\]
Let \( \rho \) be a valuation that assigns terms to the free variables, as in Definition 3.10.

4.15. **DEFINITION.** For \( \Gamma \) a context of \( \text{CC}^W \) with \( \rho, \xi \models \Gamma \), the interpretation function \( \llbracket - \rrbracket_\rho : \Gamma \text{-Term}(\text{CC}^W) \rightarrow \Lambda \) is defined by adding to Definition 3.11 the following clauses.

\[
\begin{align*}
\llbracket \text{wrec } t \rrbracket_\rho &= \text{wrec } \llbracket t \rrbracket_\rho, \\
\llbracket \text{sup}(t, q) \rrbracket_\rho &= \text{sup}(\llbracket t \rrbracket_\rho, \llbracket q \rrbracket_\rho), \\
\llbracket Wx: \sigma . t \rrbracket_\rho &= \Delta(\llbracket \sigma \rrbracket_\rho(\lambda x. (\llbracket t \rrbracket_\rho(x := x)))).
\end{align*}
\]

Again we have that

\[
\llbracket M \rrbracket_\rho := \Gamma \upharpoonright \rho(\mathcal{F} / \mathcal{F}),
\]

where \( \mathcal{F} \) are the free variables in \( M \), \( \rho(\mathcal{F}) \) is obtained by consecutively applying \( \rho \) to \( \mathcal{F} \) and \( \Gamma M \upharpoonright \) is defined in Fact 3.12. (The extension of \( \Gamma \upharpoonright \) to include cases for \( \text{wrec}, \text{sup}(\_ ,\_ ) \) and \( W \) is straightforward.)

The Strong Normalization follows immediately from the Soundness Theorem for \( \llbracket - \rrbracket_\rho \). To prove the soundness we only have verify the extra cases that arise from the additional derivation rules.

4.16. **THEOREM** (Soundness Theorem). For \( \Gamma \) a context and \( M \) and \( T \) terms of \( \text{CC}^W \),

\[
\Gamma \vdash M : T \implies \Gamma \models M : T.
\]

**PROOF.** By induction on the derivation; we verify the two relevant cases, using Lemma 4.11. Let \( \rho \) and \( \xi \) be valuations such that \( \rho, \xi \models \Gamma \).

- \( M \equiv \text{wrec } t \) with \( \Gamma \vdash t : \Pi x: \sigma . \Pi y: \tau . (\Pi y: \tau . Q(zy)) \rightarrow Q(\text{sup}(x, z)) \), \( \Gamma \vdash Q : (Wx: \sigma . t ) \rightarrow \ast \) and \( T \equiv \Pi w: Wx: \sigma . t . Qw. \) By IH

\[
\llbracket t \rrbracket_\rho \in \llbracket \sigma \rrbracket_\xi \rightarrow \llbracket \tau \rrbracket_\xi \rightarrow W(\llbracket \sigma \rrbracket_\xi, \llbracket \tau \rrbracket_\xi) \rightarrow \llbracket Q \rrbracket_\xi \rightarrow \llbracket Q \rrbracket_\xi,
\]

so \( \text{wrec } t \in W(\llbracket \sigma \rrbracket_\xi, \llbracket \tau \rrbracket_\xi) \rightarrow \llbracket Q \rrbracket_\xi \). (IH)

- \( M \equiv \text{sup}(p, q) \) with \( \Gamma \vdash p : \sigma \), \( \Gamma \vdash q : q : p / x \rightarrow Wx: \sigma . t \) and \( T \equiv Wx: \sigma . t. \) By IH \( \llbracket p \rrbracket_\rho \in \llbracket \sigma \rrbracket_\xi \) and \( \llbracket q \rrbracket_\rho \in \llbracket \tau \rrbracket_\xi \rightarrow W(\llbracket \sigma \rrbracket_\xi, \llbracket \tau \rrbracket_\xi) \). Hence, \( \text{sup}(\llbracket p \rrbracket_\rho, \llbracket q \rrbracket_\rho) \in W(\llbracket \sigma \rrbracket_\xi, \llbracket \tau \rrbracket_\xi) \). (IH)

4.17. **COROLLARY.**

\[\forall M \in \text{Term}(\text{CC}^W)[\text{SN}(M)].\]

**PROOF.** Totally similar to the proof of Theorem 3.15. \( \square \)

4.3. **CC with Sum types and recursive types**

Strong Normalization for the extension of CC with product types and sum types can be proved in the same style as we have done before for the extension with \( \Sigma \)-types and \( W \)-types.

Moreover, if products and sums are added only on the type level, the proof of SN follows immediately from the fact that these can already be defined inside the system: \( \sigma \times \tau := \Pi o: \ast . (\sigma \rightarrow \tau \rightarrow \alpha ) \rightarrow \alpha \) and \( \sigma + \tau := \Pi o: \ast . (\sigma \rightarrow \alpha ) \rightarrow (\tau \rightarrow \alpha ) \rightarrow \alpha \). It is not difficult to define closed terms \( (\_ ,\_ ) : \sigma \rightarrow \tau \rightarrow (\sigma \times \tau ) \), \( p_1 : (\sigma \times \tau ) \rightarrow \sigma \) and \( p_1 : (\sigma \times \tau ) \rightarrow \tau \) such that \( p_1(\llbracket t \rrbracket_\rho, \llbracket q \rrbracket_\rho) \) \( \rightarrow_\beta g \) \( t \) and \( p_2(\llbracket t \rrbracket_\rho, \llbracket q \rrbracket_\rho) \) \( \rightarrow_\beta q \). Similarly one can define closed terms inl : \( \sigma \rightarrow (\sigma + \tau ) \), \( \text{inr } : \tau \rightarrow (\sigma + \tau ) \) and (for every \( \rho : \ast \)) case : \( (\sigma \rightarrow \rho ) \rightarrow (\tau \rightarrow \rho ) \rightarrow (\sigma + \tau ) \rightarrow \rho \) such that \( \text{case } t q \text{inl } p \) \( \rightarrow_\beta \text{tp} \) and \( \text{case } t q \text{inr } p \) \( \rightarrow_\beta \text{qp} \).
Other product types will be treated in the next Section, when we look at the Σ-type constructor.

For sum types, we can also allow \( A + \sigma : \square, \sigma + A : \square \) and \( A + B : \square \), for \( A, B : \square \) and \( \sigma : \star \). The proof of SN uses techniques that were introduced in for CCW. To be a bit more specific, we give the rules for a rather general form of sum types and sum kinds and show how they are interpreted as sets and as saturated sets.

\[
\begin{align*}
(Sum^*) \quad & \frac{\Gamma \vdash \sigma : \star, \Gamma \vdash \tau : \star}{\Gamma \vdash \sigma + \tau : \star} & (Sum^{\square}) \quad & \frac{\Gamma \vdash T : s_1, \Gamma \vdash U : s_2}{\Gamma \vdash T + U : \square} \\
(inl) \quad & \frac{\Gamma \vdash M : T, \Gamma \vdash U : s}{\Gamma \vdash \text{inl}_TUM : T + U} & (inr) \quad & \frac{\Gamma \vdash T : s, \Gamma \vdash M : U}{\Gamma \vdash \text{inr}_TUM : T + U} \\
(case) \quad & \frac{\Gamma \vdash M : \Pi v:T_1.U[\text{inl} v], \Gamma \vdash N : \Pi v:T_2.U[\text{inr} v]}{\Gamma, v:T_1 + T_2 \vdash U[v] : s} & & \frac{\Gamma \vdash \text{case}_{T_1,T_2}MN : \Pi v:T_1 + T_2.U[v]}{}
\end{align*}
\]

In the above rules, when we write \( U[v] \), we mean that the expression \( U \) may contain the variable \( v \) and \([-]\) marks all free occurrences of \( v \) in \( U \). As a matter of fact, if \( s \) would be \( \star \) we could just have written \( U : T_1 + T_2 \rightarrow * \) and use the notation \( Uv \) in stead of \( U[v] \). (In the (case) rule, we have omitted some subscripts in the premises for reasons of readability.)

The reduction rules are

\[
\begin{align*}
\text{case} \ M_1M_2(\text{inl}_TUMN) & \rightarrow_c M_1N, \\
\text{case} \ M_1M_2(\text{inr}_TUMN) & \rightarrow_c M_2N.
\end{align*}
\]

The set of untyped lambda terms \( \Lambda \) is now extended with case, inl and inr for which we have the reduction rules

\[
\begin{align*}
\text{case} \ MN(\text{inl} P) & \rightarrow_c MP, \\
\text{case} \ MN(\text{inr} P) & \rightarrow_c NP.
\end{align*}
\]

The notions of base term, key redex and of saturated set are extended with case for case and inl and inr. To be precise, if \( P \in B \) and \( M, N \in SN \), then we let \( \text{case} \ MNP \in B \); if \( P \) has key redex \( Q \), then \( \text{case} \ MNP \) has key redex \( Q \) and for \( X \) and \( Y \) saturated sets, \( X + Y \) is defined as

\[
\{ P \mid \forall Z \in \text{SAT}\forall M \in X \rightarrow Z \forall N \in Y \rightarrow Z \forall [\text{case} \ MNP \in Z] \}.
\]

Equivalently, one can define \( X + Y \) as \( \cap \{ V \mid \text{case} \in \cap \exists \text{ESAT}(X \rightarrow Z) \rightarrow \neg Z \rightarrow V \rightarrow Z \} \). The closure properties for \( X + Y \) are then easily verified.

The interpretation of \( T + U \) as a set (if \( T + U \) is a kind) is now as follows.

\[
\begin{align*}
\forall (A + B) & := \forall (A) \cup \forall (B), \\
\forall (A + \sigma) & := \forall (A) \cup \{1\}, \\
\forall (\sigma + A) & := 1 \cup \forall (A),
\end{align*}
\]

where \( A \) and \( B \) stand for kinds and \( \sigma \) stands for a type. Here, \( \uplus \) denotes the set-theoretic disjoint union, say \( \forall (A) \cup \forall (B) := \{(0,a) \mid a \in \forall (A)\} \cup \{(1,b) \mid b \in \forall (B)\} \), and 1 denotes the singleton set with element 1.
The interpretation of types and kinds under $[-]_\xi$, where $\xi$ is a valuation that satisfies the context $\Gamma$ is as follows.

$$[T + U]_\xi := [T]_\xi \uplus [U]_\xi.$$ 

Here, $\uplus$ stands for the disjoint union of saturated sets, as defined in 5.3. Furthermore,

$$[	ext{inl}_{A,B} P]_\xi := (0, [P]_\xi),$$
$$[	ext{inr}_{A,B} P]_\xi := (1, [P]_\xi),$$
$$[	ext{inl}_{A,\tau} P]_\xi := (0, [P]_\xi),$$
$$[	ext{inr}_{A,\tau} t]_\xi := (1, 1),$$

where $A$ and $B$ stand for kinds and $\tau$ stands for a type. (The other cases for $\text{inl}$ and $\text{inr}$ are similar.) Finally,

$$[\text{case}_{A,B} PQ]_\xi := \lambda f \in \mathcal{V}(A + B). \begin{cases} P & \text{if } f = (0, a) \text{ then } [P]_\xi a \text{ else } \\ Q & \text{if } f = (1, b) \text{ then } [Q]_\xi b, \end{cases}$$

$$[\text{case}_{A,\tau} PQ]_\xi := \lambda f \in \mathcal{V}(A + \tau). \begin{cases} P & \text{if } f = (0, a) \text{ then } [P]_\xi a \text{ else } \\ Q & \text{if } f = (1, 1) \text{ then } [Q]_\xi, \end{cases}$$

$$[\text{case}_{\sigma,\tau} PQ]_\xi := \lambda f \in \mathcal{V}(\sigma + \tau). \begin{cases} P & \text{if } f = (0, a) \text{ then } [P]_\xi a \text{ else } \\ Q & \text{if } f = (1, 1) \text{ then } [Q]_\xi, \end{cases}$$

where $A$ and $B$ stand for kinds and $\sigma$ and $\tau$ stand for a types. Note that a case-expression only gets an interpretation under $[-]_\xi$ if its range is a kind. So, for example in the second case we have $P : \Pi x: A.U[\text{inl}_{A,\tau} x]$ and $Q : \Pi x: \tau.U[\text{inl}_{A,\tau} x]$ with $U[-] : \Box$.

The interpretation of terms under $[-]_\rho$ is what one expects:

$$([U + T])_\rho := d([U])_\rho ([T])_\rho,$$
$$([\text{inl}_{T,U} M])_\rho := \text{inl} ([M])_\rho,$$
$$([\text{inr}_{T,U} M])_\rho := \text{inr} ([M])_\rho,$$
$$([\text{case}_{T,U} M N])_\rho := \text{case} ([M])_\rho ([N])_\rho.$$

With these definitions it is not difficult to verify the Soundness Theorem (3.14) for $\text{ee}$ with sum-types. The Strong Normalization follows easily from it.

It is also possible to add recursive types to $\text{ee}$ and we can extend our proof method to show that this extension is SN. Of course we can not allow solutions for all type equations, but only for equations of the form

$$\alpha = \sigma(\alpha),$$

where the type variable $\alpha$ occurs only positive in the type expression $\sigma(\alpha)$. The solution to such a type equation is (as usual) denoted by $\mu \alpha.\sigma(\alpha)$ and the interpretation in terms of saturated sets is defined by

$$[\mu \alpha.\sigma(\alpha)]_\xi := \text{fix}(\lambda X. [\sigma(\alpha)]_{\xi(\alpha = X)}).$$

We shall be a bit more specific about the rules for recursive types.

4.18. DEFINITION. Let $\alpha$ be a variable and $\alpha : A$ with $A : \Box$ in $\text{ee}$. We define when $\alpha$ is positive, respectively negative in $Q$, for $Q : B(\Box)$ in $\text{CC}$.

1. If $\alpha \notin \text{FV}(Q)$, then $\alpha$ is positive and negative in $Q$. 

16
2. If \( \alpha \notin \text{FV}(\bar{M}) \), then \( \alpha \) is positive in \( \alpha \bar{M} \), with \( \bar{M} \) possibly empty,

3. If \( \alpha \) is positive in \( \sigma \) and negative in \( \tau \), then \( \alpha \) is positive in \( \Pi_x:\tau.\sigma \) and \( \alpha \) is negative in \( \Pi_x:\sigma.\tau \),

4. If \( \alpha \) is positive in \( \sigma \), negative in \( \tau \) and \( C:\Box \) with \( \alpha \notin \text{FV}(C) \), then \( \alpha \) is positive in \( \Pi\beta:C.\sigma \) and \( \alpha \) is negative in \( \Pi\beta:C.\tau \).

4.19. DEFINITION. The extension of \( \text{CC} \) with recursive types, \( \text{CC}^R \), has the following extra rules.

\[
\begin{align*}
\text{mu: } & \frac{\Gamma, \alpha:A \vdash Q: A \quad \Gamma \vdash A: \Box \quad \alpha \text{ positive in } Q}{\Gamma \vdash \mu \alpha:A.Q : A} \\
\text{in: } & \frac{\Gamma \vdash P : Q[\mu \alpha:A.Q/\alpha]}{\Gamma \vdash P : \mu \alpha:A.Q} \\
\text{out: } & \frac{\Gamma \vdash P : Q[\mu \alpha:A.Q/\alpha]}{\Gamma \vdash P : Q[\mu \alpha:A.Q/\alpha]}
\end{align*}
\]

In \( \text{CC}^R \) it is possible to define, for example, the type of natural numbers as \( N := \mu \alpha:* \cdot \Pi\beta:* . \beta → (\beta → \alpha → \beta) → \beta \). With this type it is straightforward to define functions by primitive recursion: if \( t: \sigma \) and \( f: \sigma → N → \sigma \), then \( \text{Rec}\ t f \) is defined by

\[
\text{Rec}\ t f := \lambda x:N.\text{Rec}\ f t x : N → \sigma
\]

and we have \( \text{Rec}\ t f 0 = t \) and \( \text{Rec}\ t f(Sx) = f(\text{Rec}\ t f x)x \), where

\[
0 := \lambda \beta:* . \lambda y: \beta. \lambda g: \beta → N → \beta, y \quad \text{and} \quad S := \lambda n:N. \lambda a: N. \lambda \beta:* . \lambda y: \beta. \lambda g: \beta → N → \beta, g(n, \beta, g)y)n.
\]

The Strong Normalization for \( \text{CC}^R \) follows from the fact that if \( \alpha(:A) \) is positive in \( Q(:A) \), then \( \lambda F \in \mathcal{V}(A).[Q]_{\xi(\alpha:=F)} \) is a monotone function from \( \mathcal{V}(A) \) to itself. Hence it has a fixed point. So, we extend the mapping \( [-]_\xi \) by putting

\[
[\mu \alpha:A.Q]_{\xi} := \text{lfp}(\lambda F \in \mathcal{V}(A).[Q]_{\xi(\alpha:=F)}).
\]

The mapping \( [-]_\rho \), has to be extended to include a case for the \( \mu \)-constructor as well. We put

\[
[\mu \alpha:A.Q]_{\rho} := \text{d}(\text{f}(\alpha(:A).[Q]_{\rho(\alpha:=\alpha)})).
\]

5. CC with \( \Sigma \)-types, extending the method to inductive kinds

It is well-known that one can not extend \( \text{CC} \) with arbitrary \( \Sigma \)-types: \( \Sigma \alpha:A.\sigma :* \) is not allowed if \( A : \Box \). (If one allows this, it is possible to type non-normalizing terms.) In the proof of SN for \( \text{CC} \) with 'safe' \( \Sigma \)-types that we give here, it can be seen why the proof-construction does not extend to the 'unsafe' \( \Sigma \)-types.

It is possible to proof SN for \( \text{CC}^R \) by a direct extension of the proof of SN for \( \text{CC} \) (as it was given in Section 3). This approach was taken in a previous version of this paper, which was spread at the BRA-Workshop itself. A drawback of this approach is that the untyped \( \lambda \)-calculus has to be extended with \textit{infinitary} \( \lambda \)-terms of the form \textit{case} \{\textit{Mi} | i \in I\}, where \( I \) is an infinite index set and \( M_i \) is a \( \lambda \)-term for every \( i \in I \). These 'infinite terms' are needed to interpret terms of type \( \Sigma \alpha:T.U \), for the case where \( T \) is a kind. It turns out that, if one modifies the proof of Section 3 a little bit, then these infinite terms can be avoided. This modification turns out to be of more general importance, since it also allows the interpretation of inductive kinds (like a kind of natural numbers that allows the same flexibility as the inductive type of natural numbers in Coq). This modification will be discussed later.

We now first give the rules for \( \Sigma \)-types.
5.1. **DEFINITION.** The *Calculus of Constructions with $\Sigma$-types, $\text{CC}^\Sigma$, has the following additional rules. (In these rules $s$, $s_1$ and $s_2$ stand for $\star$ or $\square$.)

\[
\frac{\Gamma \vdash \sigma : \star}{\Gamma \vdash \Sigma x: \sigma \tau : \star} \quad \frac{\Gamma \vdash T : s_1 \quad \Gamma, v : T \vdash U : s_2}{\Gamma \vdash \Sigma v : T.U : \square} \quad \text{if } s_1 \equiv \square \text{ or } s_2 \equiv \square,
\]

\[
\frac{\Gamma \vdash M : \Sigma v : T.U}{\Gamma \vdash \pi_1 M : T} \quad \frac{\Gamma \vdash M : \Sigma v : T.U}{\Gamma \vdash \pi_2 M : U[\pi_1(M)/v]} \quad \frac{\Gamma \vdash T : \square \quad \Gamma \vdash N : U[M/v]}{\Gamma \vdash \langle M, N \rangle : \Sigma v : T.U}
\]

The reduction rules associated with pairing and projection are

$$\pi_1(M, N) \rightarrow_{\pi} M, \quad \pi_2(M, N) \rightarrow_{\pi} N.$$  

The conversion rule is adapted to this new reduction, that is, the side condition $T \rightarrow U$ now stands for $\rightarrow_{\beta \pi}$, the equivalence relation generated from $\beta$- and $\pi$-reduction. For convenience we shall speak of $\text{CC}^\Sigma$ in case we want to restrict to $\Sigma$-types of the first sort, so $\Sigma v : T.U$, where $T$ and $U$ are types. ($T : \star$ and $U : \star$)

The proof of SN for $\text{CC}^\Sigma$ is a direct extension of the proof of SN for CC. We first extend the untyped $\lambda$ calculus with pairing and projection operators $(-,-), \pi_1$ and $\pi_2$ that have the required reduction behaviour

$$\pi_i([M_1, M_2]) \rightarrow_{\pi} M_i \quad (i \in \{1, 2\}).$$

5.2. **DEFINITION.** For $\text{CC}^\Sigma$, the set of base terms $B$ is defined by adding to Definition 3.1 the clause

3. If $M \in B$, then $\pi_1 M \in B$ and $\pi_2 M \in B$.

The notion of key redex is extended by adding to Definition 3.2 the clause

3. If $M$ has key redex $N$, then $\pi_i M$ has key redex $N$ (for $i \in \{1, 2\}$).

The definition of saturated set is the same as in Definition 3.3, with the notions of 'base term' and 'key redex' replaced by the above ones. We ambiguously denote this new collection of saturated sets again by SAT (but there will be no confusion).

5.3. **DEFINITION.** For $X, Y \in \text{SAT}$, the *product of $X$ and $Y$, $X \times Y$ is defined by

$$X \times Y := \{M | \pi_1 M \in X \& \pi_2 M \in Y\}.$$  

That SAT is closed under products and that elements of product sets behave correctly is stated in the following two lemmas. (The first is immediate.)

5.4. **LEMMA.** If $X, Y \in \text{SAT}$ then $X \times Y \in \text{SAT}$.

5.5. **LEMMA.** Let $X$, $Y$ and $X_i$ ($\forall i \in I$) be saturated sets.

1. If $M \in X$ and $N \in Y$, then $\langle M, N \rangle \in X \times Y$.

2. If $M \in X \times Y$, then $\pi_1 M \in X$ and $\pi_2 M \in Y$.
PROOF. The second follows immediately from the definition of product. For the first, note that \( \pi_1((M, N)) \xrightarrow{k} M \in X \) and \( \pi_1((M, N)) \) is SN, hence \( \pi_1((M, N)) \in X \). Similarly, \( \pi_2((M, N)) \xrightarrow{k} N \in Y \), so \( \pi_2((M, N)) \in Y \). \( \square \)

The notion of \( \xi \models^{D} \Gamma \) is defined analoguously to Definition 3.6.

5.6. DEFINITION. The function \([\_]_{\xi}^{E} \) is defined for \( CC^{E} \) by extending Definition 3.7 with the clause

\[ [\Sigma x:\sigma.\tau]_{\xi} = [\sigma]_{\xi} \times [\tau]_{\xi}. \]

We have the following property. (Compare with Fact 3.8.)

5.7. FACT. Let \( Q \) and \( P \) be constructors or kinds with \( \Gamma \vdash Q, P : T \) and \( \xi \) a valuation with \( \xi \models \Gamma \), then

\[ Q \xrightarrow{\beta_{\tau}} P \Rightarrow [Q]_{\xi} = [P]_{\xi}. \]

The Soundness Lemma 3.9 is also easily verified:

5.8. LEMMA (Soundness for \([\_]_{\xi}^{E} \)). For \( \Gamma \) a context of \( CC^{E} \), \( Q, A \in \text{Term}(CC^{E}) \) and \( \xi \models^{D} \Gamma \),

\[ \Gamma \vdash Q : A ; \square \Rightarrow [Q]_{\xi} \in \text{TV}(A), \]

\[ \Gamma \vdash Q : \square \Rightarrow [Q]_{\xi} \in \text{SAT}. \]

The interpretation of typable terms as (strongly normalizing) untyped \( \lambda \) terms is again done modulo a valuation \( \rho \) that assigns terms to the free variables. So, let \( \rho \) be as in Definition 3.10.

5.9. DEFINITION. For \( \Gamma \) a context of \( CC^{E} \) with \( \rho, \xi \models \Gamma \), the interpretation function \([\_]_{\rho}^{E} : \Gamma ; \text{Term}(CC^{E}) \rightarrow A \) is defined by adding to Definition 3.11 the following clauses.

\[ ([t, q])_{\rho} = ([t])_{\rho} \times ([q])_{\rho}, \]

\[ ([\pi_1 t])_{\rho} = \pi_1([t])_{\rho}, \]

\[ ([\pi_2 t])_{\rho} = \pi_2([t])_{\rho}, \]

\[ ([\Sigma x:\sigma.\tau])_{\rho} = d([\sigma])_{\rho}(\lambda x.([\tau])_{\rho(x:=x)}). \]

Again we have that

\[ [M]_{\rho} : = \Gamma M^\gamma[\rho(\overline{v})/\overline{v}], \]

where \( \overline{v} \) are the free variables in \( M \), \( \rho(\overline{v}) \) is obtained by consecutively applying \( \rho \) to \( \overline{v} \) and \( \Gamma M^\gamma \) is defined in Fact 3.12. (The extension of \( \Gamma - \gamma \) to include cases for \( \langle -, - \rangle, \pi_i \) and \( \Sigma \) is straightforward.)

The Strong Normalization follows immediately from the Soundness Theorem for \([\_]_{\rho}^{E} \). To prove the soundness we only have to verify the extra cases that arise from the additional derivation rules. This is straightforward.

5.10. THEOREM (Soundness Theorem). For \( \Gamma \) a context and \( M \) and \( T \) terms of \( CC^{E} \),

\[ \Gamma \vdash M : T \Rightarrow \Gamma \models M : T. \]
5.11. Corollary.

\[ \forall M \in \text{Term}(\text{CC}^\Sigma)[SN(M)]. \]

Proof. Totally similar to the proof of Theorem 3.15. \(\square\)

We now come to the interpretation of so-called 'large' \(\Sigma\)-types (i.e., where the \(\Sigma\)-type is actually a kind) as saturated sets. It turns out that if \(\sigma\) is a type, then \([\Sigma \sigma: A.B]_\xi\) can be defined as \([\sigma]_\xi \times [B]_\xi\). (Note that \([B]_\xi\) does not depend on objects; \(\xi\) does not give a value to object variables.) If \(A\) is a kind, then one can not define \([\Sigma \sigma: A.T]_\xi := [A]_\xi \times [T]_\xi\), because now \([T]_\xi\) depends on the value that \(\xi\) takes for \(\sigma\). One would like to define a 'dependent product of saturated sets' and interpret \(\Sigma \sigma: A.T\) as such a dependent product. This turns out to be very complicated and we therefore take a different approach.

Instead of interpreting kinds as saturated sets under \([-\]_\xi\), we shall interpret kinds as saturated sets parametrized over their set-interpretation. So, if \(A\) is a kind, we define \([A]_\xi\) as a function from \(\forall a \in A\) to SAT. For the interpretation of types we take (as before) saturated sets. Then the statement of Soundness of the interpretation will have the following form.

\[
\Gamma \vdash t : \sigma \quad \Rightarrow \quad \forall \rho, \xi \models \Gamma[[\sigma]]_\rho \in [\sigma]_\xi,
\]

\[
\Gamma \vdash P : A \quad \Rightarrow \quad \forall \rho, \xi \models \Gamma[[P]]_\rho \in [A]_\xi([[P]_\xi]),
\]

where \(\sigma\) stands for a type and \(A\) for a kind.

We now make precise how the definitions of \(\forall\), \([-\]_\xi\) and \([-\]_\rho\) have to be adapted to achieve the above.

5.12. Definition. The extension of the set-interpretation \(\forall\) to the kinds of \(\text{CC}^\Sigma\) is done by adding the following clauses to Definition 3.4.

\[
\forall(\Sigma \sigma: A.B) := \forall(A) \times \forall(B), \text{ if } A, B : \Box,
\]

\[
\forall(\Sigma \sigma : A.\top) := \forall(A), \text{ if } A : \Box \text{ and } \sigma : \top,
\]

\[
\forall(\Sigma \sigma : A.B) := \forall(B), \text{ if } B : \Box \text{ and } \sigma : \top.
\]

The notion of \(\xi \models \Box \Gamma\) (the constructor valuation \(\xi\) satisfies \(\Gamma\)) is as before in Definition 3.6.

5.13. Definition. The extension of \([-\]_\xi\) (definition 3.7) to \(\text{CC}^\Sigma\) is done by changing the clauses for \(\top\) and \(\Pi\)-kinds and by adding clauses for \(\Sigma\)-types and its constructors as follows.

\[
[-\]_\xi = \lambda X \in \text{SAT.SN},
\]

\[
[\Pi x : \sigma.B]_\xi = \lambda b \in \forall(B).[[\sigma]_\xi \rightarrow [B]_\xi(b)),
\]

\[
[\Pi \alpha : A.B]_\xi = \lambda f \in \forall(A) \rightarrow \forall(B). \bigcap_{a \in \forall(A)} [A]_\xi(a) \rightarrow [B]_\xi(a = \alpha)(fa),
\]

\[
[\Sigma x : \sigma.B]_\xi = \lambda b \in \forall(B). \bigcap_{X \text{ ranging over SAT}} (([[\sigma]_\xi \rightarrow [B]_\xi(b) \rightarrow X) \rightarrow X),
\]

\[
[\Sigma \sigma : A.B]_\xi = \lambda f \in \forall(A) \times \forall(B). \bigcap_{a \in \forall(A)} \bigcap_{b \in \forall(B)} [B]_\xi(a = \alpha)(b) \rightarrow X((a, b)))) \rightarrow X(p),
\]

\[
X \text{ ranging over } \forall(A) \times \forall(B) \rightarrow \text{SAT},
\]

20
\[
[\Sigma \alpha : A. \tau]_\xi = \lambda a \in V(A) \bigcap \bigcap_{b \in V(A)} [A]_\xi(b) \rightarrow [\tau]_\xi(a : = b) \rightarrow X(b) \rightarrow X(a),
\]

\(X\) ranging over \(V(A) \rightarrow \text{SAT}\).

\[
[(P, Q)]_\xi = ([P]_\xi, [Q]_\xi),
\]

\[
[(p, q)]_\xi = [P]_\xi,
\]

\[
[\pi_1 Q]_\xi = \text{fst} ([Q]_\xi),
\]

\[
[\pi_2 Q]_\xi = \text{snd} ([Q]_\xi).
\]

Here, \(\rightarrow\) denotes set-theoretic function space construction if it is in the subscript of a \(\cap\); otherwise it denotes the function space on saturated sets. Furthermore, \((- , - )\) denotes pairing and \(\text{fst}\) and \(\text{snd}\) denote projections in set-theory. Remember that \(\sigma\) and \(\tau\) stand for types, \(A\) and \(B\) stand for kinds, \(p\) and \(q\) stand for objects and \(P\) and \(Q\) stand for constructors.

To understand why the \(\Sigma\)-type has this interpretation, one can take a look at the interpretation of the weak existential quantifier \(\exists \alpha : A. \tau\) in CC. Following Definition 3.7, we have

\[
[\exists \alpha : A. \tau]_\xi = \text{SN}\rightarrow \bigcap_{X \in \text{SAT}}\bigcap_{a \in V(A)} [A]_\xi(a) \rightarrow [\tau]_\xi(\alpha : = a) \rightarrow X \rightarrow X.
\]

Now, for this interpretation we do not have a second projection. (It would have to be \(\lambda z. \tau(\lambda y. y)\), but this term is not in the right saturated set, because one cannot take \([\tau]_\xi(\alpha : = a)\) for \(X\) in the intersection.) Therefore we have to adapt the interpretation to get a real strong \(\Sigma\)-type (with projections).

It is now easy to verify the substitution property for \([-]_\xi\) and to show that \([-]_\xi\) preserves reduction (compare with Fact 3.8): \([M[Q/a]]_\xi = [M]_\xi(\alpha : = [Q]_\xi)\), \([M[q/x]]_\xi = [M]_\xi\) and if \(M \rightarrow_{\beta} N\), then \([M]_\xi = [N]_\xi\), provided that \(M\) is a kind or a constructor.

Hence we can prove the Soundness Lemma (compare with Lemma 3.9 and Lemma 5.8).

5.14. LEMMA (Soundness Lemma). For \(\Gamma\) a context of \(\text{CC}^\Sigma\), \(Q, A \in \text{Term}(\text{CC}^\Sigma)\) and \(\xi \vdash \Gamma\),

\[\Gamma \vdash Q : A : \Box \Rightarrow [Q]_\xi \in V(A),\]

\[\Gamma \vdash A : \Box \Rightarrow [A]_\xi \in V(A) \rightarrow \text{SAT}.\]

PROOF. By simultaneous induction on the derivation. \(\Box\)

To define the interpretation \([-]_\rho\), we have to say when a valuation \(\rho\) satisfies \(\Gamma\) with respect to \(\xi\) (notation \(\rho, \xi \models \Gamma\); see also Definition 3.10). This is the case when

\[
\begin{align*}
x : \sigma \in \Gamma & \Rightarrow \rho(x) \in [\sigma]_\xi, \\
\alpha : A \in \Gamma & \Rightarrow \rho(\alpha) \in [A]_\xi(\xi(\alpha)).
\end{align*}
\]

5.15. DEFINITION. The interpretation of objects, constructors and kinds of \(\text{CC}^\Sigma\) under \([-]_\rho\) is done by adding to Definition 3.11 the following clauses.

\[
\begin{align*}
\llbracket (M, N) \rrbracket_\rho &= \lambda z. \tau(\lambda y. [N]_\rho)_{[M]_\rho}, \\
\llbracket \pi_1 M \rrbracket_\rho &= [M]_\rho(\lambda xy. x), \\
\llbracket \pi_2 M \rrbracket_\rho &= [M]_\rho(\lambda xy. y), \\
\llbracket \Sigma v : T . U \rrbracket_\rho &= d([T]_\rho(\lambda v. [U]_\rho)_{[v = 0]}).
\end{align*}
\]
Here, \( v \) is either an object variable or a constructor variable, \( T \) and \( U \) are either types or kinds and \( M \) and \( N \) are either objects or constructors.

An important thing to notice is that, if \( M \rightarrow_{\gamma} N \), then \( \{M\}_\rho \rightarrow_{\beta} \{N\}_\rho \) in at least one step. This can be seen by observing that \( \{\pi_1(M_1, M_2)\}_\rho \equiv (\lambda y. y(M_1\rho)(M_2\rho))(\lambda x_1 x_2 x_4) \rightarrow_{\beta} \ldots \rightarrow_{\beta} \{M_i\}_\rho \). Hence it is easy to verify that, if one defines \( \rho_0 \) by taking \( \rho_0(v) = v \) for all variables \( v \), then
\[
\{M\}_\rho \text{ is SN } \implies M \text{ is SN}.
\]

Due to the definition of saturated sets, the valuation \( \rho_0 \) will always satisfy \( \Gamma \) with respect to \( \xi \).

The notion of \( \Gamma \models M : T \) (\( \Gamma \) satisfies that \( M \) is of type \( T \)) now takes the following form.

(Compare with Definition 3.13.)

5.16. DEFINITION. For \( \Gamma \) a context and \( t \) an object, \( \sigma \) a type, \( P \) a constructor and \( A \) a kind of \( CC^E \), we define
\[
\Gamma \models t : \sigma \quad \text{iff} \quad \forall \xi, \rho, \xi \models \Gamma \implies \{t\}_\rho \in \llbracket \sigma \rrbracket_\xi.
\]
\[
\Gamma \models P : A \quad \text{iff} \quad \forall \xi, \rho, \xi \models \Gamma \implies \{P\}_\rho \in \llbracket A \rrbracket_\xi(\llbracket P \rrbracket_\xi).
\]

5.17. THEOREM (Soundness Theorem). For \( \Gamma \) a context and \( M \) and \( T \) terms of \( CC^E \),
\[
\Gamma \models M : T \implies \Gamma \models M : T.
\]

PROOF. The proof is by induction on the derivation. We treat a few cases.

- \( M \equiv \langle P, t \rangle \) with \( P : A \) and \( t : \tau(P/\alpha) \). Then by IH, \( \{P\}_\rho \in \llbracket A \rrbracket_\xi(\llbracket P \rrbracket_\xi) \) and \( \{t\}_\rho \in \llbracket \tau(P/\alpha) \rrbracket_\xi(\llbracket P \rrbracket_\xi) \). Now, let \( X \in \forall(A) \rightarrow \text{SAT} \) and 
\[
y \in \bigcap_{b \in \forall(A)} \llbracket A \rrbracket_\xi(b) \rightarrow \llbracket T \rrbracket_\xi(b) \rightarrow X(b).
\]
Then, taking \( \{P\}_\rho \) for \( b \), we find that \( y(\{P\}_\rho(\{t\}_\rho) \in X(\llbracket P \rrbracket_\xi) \).

So, \( \lambda y. y(\{P\}_\rho(\{t\}_\rho) \in \Sigma_\alpha : A. \tau(\llbracket P \rrbracket_\xi), \) that is, \( \llbracket \langle P, t \rangle \rrbracket_\rho \in \Sigma_\alpha : A. \tau(\llbracket P \rrbracket_\xi) \).

- \( M \equiv \langle P, Q \rangle \) with \( P : A \) and \( Q : B[P/\alpha] \). Then by IH, \( \{P\}_\rho \in \llbracket A \rrbracket_\xi(\llbracket P \rrbracket_\xi) \) and \( \{Q\}_\rho \in \llbracket B[P/\alpha] \rrbracket_\xi(\llbracket Q \rrbracket_\xi) \). Now, let \( X \in \forall(A) \times \forall(B) \rightarrow \text{SAT} \) and 
\[
y \in \bigcap_{a \in \forall(A)} \llbracket A \rrbracket_\xi(a) \rightarrow \bigcap_{b \in \forall(B)} \llbracket B \rrbracket_\xi(b) \rightarrow X((a, b)).
\]
Then, taking \( \{P\}_\xi \) for \( a \) and \( \{Q\}_\xi \) for \( b \), we find that \( y(\{P\}_\rho(\{Q\}_\rho) \in X(\llbracket P \rrbracket_\xi, \llbracket Q \rrbracket_\xi) \).

So, \( \lambda y. y(\{P\}_\rho(\{Q\}_\rho) \in \Sigma_\alpha : A. B(\llbracket P \rrbracket_\xi, \llbracket Q \rrbracket_\xi), \) that is, \( \llbracket \langle P, Q \rangle \rrbracket_\rho \in \Sigma_\alpha : A. B(\llbracket P \rrbracket_\xi) \).

- \( M \equiv \pi_1 P \) with \( P : \Sigma_\alpha : A. \tau \). Then by IH, \( \{P\}_\rho \in \Sigma_\alpha : A. \tau(\llbracket P \rrbracket_\xi), \) that is, \( \llbracket P \rrbracket_\rho \in \bigcap_{x \in \forall(A)} \forall(x) \rightarrow \text{SAT} \cap \bigcap_{y \in \forall(V)} \forall(y) [A]_{\xi}(b) \rightarrow [\tau]_{\xi(\pi_1 \alpha = p)} \rightarrow X(\{P\}_\xi) \). Now, taking \( \{A\}_\xi \) for \( X \) we find that \( \lambda x. y.x \in \bigcap_{x \in \forall(A)} [A]_{\xi}(b) \rightarrow [\tau]_{\xi(\pi_1 \alpha = p)} \rightarrow [A]_{\xi}(b) \). So, \( \{P\}_\rho(\lambda x. y.x) \in [A]_{\xi}(\llbracket P \rrbracket_\xi) \), that is, \( \{\pi_1 \}P \rho \in \llbracket A \rrbracket_\xi(\llbracket \pi_1 \rrbracket_\xi). \)

- \( M \equiv \pi_2 P \) with \( P : \Sigma_\alpha : A. B \). Then by IH, \( \{P\}_\rho \in \Sigma_\alpha : A. B(\llbracket P \rrbracket_\xi), \) that is, \( \{P\}_\rho \in \bigcap_{y \in \forall(V)} [A]_{\xi}(a) \rightarrow \bigcap_{x \in \forall(V)} [B]_{\xi(\alpha = x)}(b) \rightarrow X((a, b)) \). Now, taking \( \lambda x \in \forall(A) \times \forall(B) \) for any \( X \) in \( \forall(A) \times \forall(B) \rightarrow \text{SAT} \), we find that \( \lambda x. y.y \in \bigcap_{y \in \forall(V)} [A]_{\xi}(a) \rightarrow \bigcap_{x \in \forall(V)} [B]_{\xi(\alpha = x)}(b) \rightarrow [B]_{\xi(\alpha = x)}(b) \). So, \( \{P\}_\rho(\lambda x. y.y) \in [B]_{\xi(\alpha = \text{fst}(\{P\}_\xi))} \text{snd}(\{P\}_\xi), \) that is, \( \{\pi_2 \}P \rho \in \llbracket B \pi_1 \alpha / \alpha \} \llbracket P \rrbracket_\xi(\llbracket \pi_2 \rrbracket_\xi). \)
The following is now an immediate consequence of the fact that we have for every context \( r \) a constructor valuation \( \xi \) such that \( \xi \models^{\text{constr}} \Gamma \) and furthermore, that for the identity valuation \( \rho_0 \), we have \( \rho_0, \xi \models \Gamma \). (See the proof of 3.15 for details.)

5.18. **COROLLARY** (Strong Normalization for \( \text{CC}^D \)).

\[ \forall M \in \text{Term}(\text{CC}^D)[\text{SN}(M)]. \]

The version of \( \Sigma \)-types that makes \( \text{CC} \) inconsistent is the one that lets \( \Sigma A : A. \tau : \star \) if \( A : \Box \) and \( \tau : \star \). It is instructive to see why this version of \( \Sigma \)-types does not fit into the proof of SN above. Suppose we let \( \Sigma A : A. \tau : \star \), with \( A : \Box \) and \( \tau : \star \). Then we do not define \( V(\Sigma A : A. \tau) \), because this is not a kind. Furthermore, we can define \( [\Sigma A : A. \tau]_\xi \) as before. The problem arises when we try to define \( [\pi_1 t]_\xi \) for \( t : \Sigma A : A. \tau : \star \), because \( [\pi_1 t]_\xi \) can not be defined in terms of \( [t]_\xi \), for the simple reason that \( [t]_\xi \) does not exist. (Note that \( t \) is an object and for objects \( [-]_\xi \) is not defined.)

The approach to proving strong normalization can be generalised to inductive kinds. We treat the example for natural numbers. In the following, note that our 'inductive types' are *kinds*, whereas in a system like Coq, they are *types*. Having the natural numbers on the kind-level conforms better with a more traditional view on logical systems, where the level of 'domains' and the level of 'formulas' are separated. We now give the syntactic rules for the kind \( \text{Nat} \).

\[
\begin{align*}
\text{(Nat)} & \quad \vdash \text{Nat} : \Box, & \text{(Zero)} & \quad \vdash Z : \text{Nat}, & \text{(Succ)} & \quad \vdash S : \text{Nat} \rightarrow \text{Nat}, \\
\text{(Elim)} & \quad \Gamma, \alpha : \text{Nat} \vdash T : \star / \Box & \Gamma \vdash M_1 : T[\alpha Z/a] & \Gamma \vdash M_2 : \Pi \alpha : \text{Nat}. T \rightarrow T[Z \sigma/a] & \Gamma, \alpha : \text{Nat} \vdash \text{RecM}_1 M_2 \alpha : T
\end{align*}
\]

with the reduction rules

\[
\begin{align*}
\text{RecM}_1 M_2 Z & \rightarrow_r M_1, & \text{RecM}_1 M_2 (S \alpha) & \rightarrow_r M_2 \alpha (\text{RecM}_1 M_2 \alpha)
\end{align*}
\]

The system \( \text{CC} \) extended with this scheme for natural numbers will be denoted by \( \text{CC}^N \).

5.19. **DEFINITION.** Adapt the mappings \( V, [-]_\xi \) and \( [-]_\rho \) by adding the following clauses.

\[
\begin{align*}
V(\text{Nat}) := & \quad \text{N}, \\
[Z]_\xi := & \quad 0, \\
[S]_\xi := & \quad \lambda n \in \text{N}. n + 1, \\
[\text{RecP}_1 P_2]_\xi := & \quad \text{the function defined by primitive recursion from } ([P_1]_\rho \text{ and } ([P_2]_\rho), \\
[\text{Nat}]_\xi := & \quad \text{ifp}(\Phi), \text{ where for } N \in \text{N} \rightarrow \text{SAT}, \\
\Phi(N) := & \quad \lambda n \in \text{N}. \bigcap_{X \in \text{N} \rightarrow \text{SAT}} X(0) \rightarrow \bigcap_{m \in \text{N}} (N(m) \rightarrow X(m) \rightarrow X(m + 1)) \rightarrow X(n), \\
([Z]_\rho := & \quad \lambda xy. x, \\
([S]_\rho := & \quad \lambda zxy. (\lambda uv. xxy)z), \\
([\text{RecM}_1 M_2]_\rho := & \quad \lambda z. [M_1]_\rho ([M_2]_\rho).
\end{align*}
\]

We shall ambiguously denote \( [\text{Nat}]_\xi \) by \( \text{Nat} \).
The function $\lambda n : \mathbb{N} \to \text{SAT}. \lambda n \in \mathbb{N} \cap \bigcap_{X \in \mathbb{N} \to \text{SAT}} X(0) \to \bigcap_{m \in \mathbb{N}} N(m) \to X(m) \to X(m+1) \to X(n)$ is a positive morphism from $\mathbb{N} \to \text{SAT}$ to $\mathbb{N} \to \text{SAT}$ and hence it has a least fixed point (lfp). (Compare with Definition 4.6 and Corollary 4.8.) A term $\text{Rec}_1 \text{Rec}_2$ can be a constructor (if $T$ in the scheme is a kind) or an object (if $T$ in the scheme is a type). In the second case it only has an interpretation under $\langle \cdot \rangle_\rho$, in the first case it has two interpretations. If $\text{Rec}_1 \text{Rec}_2 : \text{I} : \text{Nat} \to T$, with $T$ a kind, then $\langle \text{Rec}_1 \text{Rec}_2 \rangle_\xi$ is the function $F : \mathbb{N} \to \mathcal{V}(T)$, defined by $F(0) = \langle M_1 \rangle_\xi$ and $F(n+1) = \langle M_2 \rangle_\xi(n)(F(n))$.

5.20. LEMMA (Soundness Lemma). For $\Gamma$ a context of $\text{CC}^N$, $P, A \in \text{Term}(\text{CC}^N)$ and $\xi \models \Box \Gamma$, $\Gamma \vdash P : A(\Box) \Rightarrow \langle P \rangle_\xi \in \mathcal{V}(A)$,

\begin{align*}
\Gamma \vdash A : \Box & \Rightarrow \langle A \rangle_\xi \in \mathcal{V}(A) \to \text{SAT}.
\end{align*}

PROOF. By induction on the derivation. The only interesting case is when the last rule was (Elim) and $P \equiv \text{Rec}_1 \text{Rec}_2$. Then, by IH, $\langle P_1 \rangle_\xi \in \mathcal{V}(T)$ and $\langle P_2 \rangle_\xi \in \mathbb{N} \to \mathcal{V}(T) \to \mathcal{V}(T)$. So, indeed $\langle \text{Rec}_1 \text{Rec}_2 \rangle_\xi \in \mathbb{N} \to \mathcal{V}(T)$ and we are done. □

The notion of the valuation $\rho$ satisfies $\Gamma$ with respect to $\xi$, $\rho, \xi \models \Gamma$, is defined as before:

\begin{align*}
x : \alpha \in \Gamma & \Rightarrow \rho(x) \in \sigma \langle \xi \rangle,
\alpha : A \in \Gamma & \Rightarrow \rho(\alpha) \in \langle A \rangle_\xi(\xi(\alpha)).
\end{align*}

So is the notion of $\Gamma$ satisfies $M : T (\Gamma \models M : T)$, which is the same as in Definition 5.16.

5.21. THEOREM (Soundness Theorem). For $\Gamma$ a context and $M$ and $T$ terms of $\text{CC}^N$, $\Gamma \vdash M : T \Rightarrow \Gamma \models M : T$.

PROOF. By induction on the derivation.

- $M \equiv Z$. Let $X \in \mathbb{N} \to \text{SAT}$. If $x \in X(0)$ and $y \in \bigcap_{m \in \mathbb{N}} \text{Nat}(m) \to X(m) \to X(m+1)$, then $x \in X(0)$, so $\lambda xy. x \in \text{Nat}(0)$.

- $M \equiv S$. We have to prove that $\langle S \rangle_\rho \in \bigcap_{p \in \mathbb{N}} \text{Nat}(p) \to \text{Nat}(p+1)$. Let $p \in \mathbb{N}$ and $x \in \text{Nat}(p)$. Let also $X \in \mathbb{N} \to \text{SAT}$, $x \in X(0)$ and $y \in \bigcap_{m \in \mathbb{N}} \text{Nat}(m) \to X(m) \to X(m+1)$. Then $yz \in X(p) \to X(p+1)$ and $\lambda vxz.y \in X(p)$, so $yz(\lambda vxz.y) \in X(p+1)$. Hence, $\lambda x y z(yz(\lambda vxz.y)) \in \bigcap_{p \in \mathbb{N}} \text{Nat}(p) \to \text{Nat}(p+1)$.

- $M \equiv \text{Rec}_1 \text{Rec}_2$. We have to prove that $\langle \text{Rec}_1 \text{Rec}_2 \rangle_\rho \in \bigcap_{n \in \mathbb{N}} \text{Nat}(n) \to \bigcap_{\alpha \in \mathbb{N}} \text{Nat}(\alpha) \to \text{Nat}(n) \to \text{Nat}(n+1)$. By IH, $\langle M_1 \rangle_\rho \in \bigcap_{n \in \mathbb{N}} \text{Nat}(n) \to \bigcap_{\alpha \in \mathbb{N}} \text{Nat}(\alpha) \to \bigcap_{n \in \mathbb{N}} \text{Nat}(n) \to \text{Nat}(n+1)$.

Let $n \in \mathbb{N}$ and $x \in \text{Nat}(n)$. Take for $X$ the map $\lambda m \in \mathbb{N}. \langle T \rangle_{\xi(\alpha = n)}(\langle \text{Rec}_1 \text{Rec}_2 \rangle_\xi(m))$, then $\langle M_1 \rangle_\rho \in X(0)$ and $\langle M_2 \rangle_\rho \in \bigcap_{n \in \mathbb{N}} \text{Nat}(n) \to X(n) \to X(n+1)$, by taking $t$ to be $\langle \text{Rec}_1 \text{Rec}_2 \rangle_\xi(n)$. Hence we find that $\langle \text{Rec}_1 \text{Rec}_2 \rangle_\rho = \lambda z. \langle M_1 \rangle_\rho(\langle M_2 \rangle_\rho(\langle X \rangle_\rho(\langle T \rangle_{\xi(\alpha = n)}(\langle \text{Rec}_1 \text{Rec}_2 \rangle_\xi(n))))$. □

24
5.22. COROLLARY (Strong Normalization for $\text{CC}^N$).

$$\forall M \in \text{Term}(\text{CC}^N)[SN(M)].$$

The Corollary follows in a standard way from the Theorem (see the proof of Theorem 3.15) by taking for $\rho$ the identity valuation $\rho_0$ and by observing that, if $M \rightarrow^* N$, then $[M]_{\rho_0}$ reduces to $[N]_{\rho_0}$ in at least one step. For the latter:

$$[\text{Rec}_M^1 M_2(Sx)]_\rho = (\lambda z.(M_1)_\rho(M_2)_\rho)((\lambda z p q. q z ((\lambda v. v p q) z)) \rho(x)),$$

which reduces to

$$[M_2]_{\rho}(x)((\lambda v.(M_1)_\rho(M_2)_\rho)\rho(x)) \equiv [M_2(\text{Rec}_M^1 M_2 x)]_\rho.$$

Concluding Remarks

We have given a short and flexible proof of Strong Normalization for the Calculus of Constructions. The flexibility lies in the fact that the framework of saturated sets allows many basic constructions like function types, product types, sum types and positive recursive types. A question that has not been addressed here is whether this construction can be extended to higher universes (adding a sort $\Box_1$ with $\Box : \Box_1$, etcetera). We did look into the extension with inductive types: the example of the natural numbers strongly suggests a general procedure for other inductive types by (roughly) interpreting an inductive type $T$ as the parametrized saturated that corresponds with the elimination scheme of $T$. Note however, that, different from a system like Coq, the inductive types are in fact kinds here (also called ‘large types’). This fits rather naturally with the approach that we have chosen, where the interpretation of a type does not depend on the interpretation of an object. It is not clear to us whether this approach yields some principle restriction to the extendibility of our proof. Furthermore, it is also not clear to us whether the fact that we have inductive kinds puts a limitation on the expresibility of the system (compared with inductive types).

References

[McAllester et al. 1994] D. McAllester, J. Kučan and D.F. Otth, A proof of Strong Normalization for $F_2$, $F_\omega$ and beyond MIT Laboratory for Computer Science, Manuscript, 16 pp.

[Altenkirch 1993a] Th. Altenkirch, Yet another Strong Normalization proof for the Calculus of Constructions, Laboratory for Foundations of Computer Science, Manuscript, 11 pp.

[Altenkirch 1993] Th. Altenkirch, Constructions, Inductive types and Strong Normalization proof, Ph. D. Thesis, University of Edinburgh, UK.

[Barendregt 1984] H.P. Barendregt, The lambda calculus: its syntax and semantics, revised edition. Studies in Logic and the Foundations of Mathematics, North Holland.

[Barendregt 1992] H.P. Barendregt, Typed lambda calculi. In Abramski et al. (eds.), Handbook of Logic in Computer Science, Oxford Univ. Press.

[Berardi 1988] S. Berardi, Towards a mathematical analysis of the Coquand-Huet calculus of constructions and the other systems in Barendregt's cube. Dept. Computer Science, Carnegie-Mellon University and Dipartimento Matematica, Universita di Torino, Italy.
[Berardi 1989] S. Berardi, Talk given at the 'Jumelage meeting on typed lambda calculus', Edinburgh, September 1989.

[Berardi 199+] S. Berardi, Encoding of data types in Pure Construction Calculus: a semantic justification. To appear in the Proceedings of the second BRA meeting on Logical Frameworks, Edinburgh, May 1991.

[Coquand 1985] Th. Coquand, Une théorie des constructions, Thèse de troisième cycle, Université Paris VII, France.

[Coquand 1986] Th. Coquand, An analysis of Girard's paradox, Proceedings of the first symposium on Logic in Computer Science, Cambridge Mass., IEEE, pp 227-236.

[Coquand 1990] Th. Coquand, Metamathematical investigations of a calculus of constructions. In Logic and Computer Science, ed. P.G. Odifreddi, APIC series, vol. 31, Academic Press, pp 91-122.

[Coquand and Gallier 1990] Th. Coquand and J. Gallier, A proof of Strong Normalization for the Theory of Constructions using a Kripke-like interpretation, In the Informal Proceedings of the Workshop on Logical Frameworks, Antibes, May 1990.

[Coquand and Huet 1988] Th. Coquand and G. Huet, The calculus of constructions, Information and Computation, 76, pp 95-120.

[Coquand and Mohring 1990] Th. Coquand and Ch. Paulin-Mohring Inductively defined types, In P. Martin-Löf and G. Mints editors. COLOG-88 : International conference on computer logic, LNCS 417.

[Dowek e.a. 1991] G. Dowek, A. Felty, H. Herbelin, G. Huet, Ch. Paulin-Mohring, B. Werner, The Coq proof assistant version 5.6, user's guide. INRIA Rocquencourt - CNRS ENS Lyon.

[Geuvers and Nederhof 1991] J.H. Geuvers and M.J. Nederhof, A modular proof of strong normalisation for the calculus of constructions. Journal of Functional Programming, vol 1 (2), pp 155-189.

[Geuvers 1993] J.H. Geuvers, Logics and Type Systems, Ph. D. thesis, Universiteit Nijmegen, the Netherlands.

[Geuvers 1994] J.H. Geuvers, Conservativity between Logics and typed λ calculi, to appear in Proceedings of the 1993 BRA Workshop on Types for Proofs and Programs, LNCS.

[Geuvers and Werner 1994] H. Geuvers and B. Werner, On the Church-Rosser property for Expressive Type Systems and its Consequences for their Metatheoretic Study, in Proceedings of the Ninth Annual Symposium on Logic in Computer Science, Paris, France, IEEE Computer Society, pp 320-329.

[Gallier 1990] On Girard's "Candidats de Reductibilité". In Logic and Computer Science, ed. P.G. Odifreddi, APIC series, vol. 31, Academic Press, pp 123-204.

[Girard 1972] J.-Y. Girard, Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur. Ph.D. thesis, Université Paris VII, France.
[Girard 1986] J.-Y. Girard, The system F of variable types, fifteen years later. TCS 45, pp 159-192.

[Girard et al. 1989] J.-Y. Girard, Y. Lafont and P. Taylor, Proofs and types, Camb. Tracts in Theoretical Computer Science 7, Cambridge University Press.

[Goguen 1994] H. Goguen, A Typed Operational Semantics for Type Theory, submitted for PhD. examination University of Edinburgh, UK, 1994.

[Luo 1990] Z. Luo, An Extended Calculus of Constructions, Ph. D. Thesis, University of Edinburgh, UK.

[Luo 1989] Z. Luo, ECC: An extended Calculus of Constructions. Proc. of the fourth ann. symp. on Logic in Comp. Science, Asilomar, Cal. IEEE, pp 386-395.

[Martin-Löf 1971] P. Martin-Löf, A theory of types, manuscript, October 1971.

[Martin-Löf 1975] P. Martin-Löf, An intuitionistic theory of types: predicative part. Logic Colloquium '73, North-Holland 1975, pp 73-118.

[Martin-Löf 1982] P. Martin-Löf, Constructive mathematics and computer programming. Sixth International Congress for Logic, Methodology, and Philosophy of Science VI, 1979, North-Holland 1982, pp 153-175.

[Martin-Löf 1984] P. Martin-Löf, Intuitionistic Type Theory, Studies in Proof theory, Bibliopolis, Napoli.

[Nordström et al. 1990] B. Nordström, K. Petersson, J.M. Smith, Programming in Martin-Löf's Type Theory. Oxford University Press.

[Ong and Ritter 1994] L. Ong and E. Ritter, A generic Strong Normalization argument: application to the Calculus of Constructions, University of Cambridge Computer Laboratory, Manuscript, 19 pp.

[Scedrov 1990] A guide to polymorphic types. In Logic and Computer Science, ed. P.G. Odifreddi, APIC series, vol. 31, Academic Press, pp 387-420.

[Tait 1965] W.W. Tait, Infinitely long terms of transfinite type. In Formal Systems and Recursive Functions, eds. J.N. Crossley and M.A.E. Dummett, North-Holland.

[Tait 1975] W.W. Tait, A realizability interpretation of the theory of species. In Proceedings of Logic Colloquium, ed. R. Parikh, LNM 453, pp 240-251.

[Terlouw 1993] J. Terlouw, Strong Normalization in type systems: a model theoretic approach, In the Dirk van Dalen Festschrift, Eds. H. Barendregt, M. Bezem and J.W. Klop, Department of Philosophy, Utrecht University, the Netherlands, pp 161-190.
In this series appeared:

91/01 D. Alstein  
Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.

91/02 R.P. Nederpelt  
H.C.M. de Swart  
Implication. A survey of the different logical analyses "if...,then...", p. 26.

91/03 J.P. Katoen  
L.A.M. Schoenmakers  
Parallel Programs for the Recognition of \( P \)-invariant Segments, p. 16.

91/04 E. v.d. Sluis  
A.F. v.d. Stappen  
Performance Analysis of VLSI Programs, p. 31.

91/05 D. de Reus  
An Implementation Model for GOOD, p. 18.

91/06 K.M. van Hee  
SPECIFICATIEMETHODEN, een overzicht, p. 20.

91/07 E.Poll  
CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.

91/08 H. Schepers  
Terminology and Paradigms for Fault Tolerance, p. 25.

91/09 W.M.P.v.d. Aalst  
Interval Timed Petri Nets and their analysis, p. 53.

91/10 R.C. Backhouse  
P.J. de Bruin  
P. Hoogendijk  
G. Malcolm  
E. Voermans  
J. v.d. Woude  
POLYNOMIAL RELATORS, p. 52.

91/11 R.C. Backhouse  
P.J. de Bruin  
G. Malcolm  
E. Voermans  
J. van der Woude  
Relational Catamorphism, p. 31.

91/12 E. van der Sluis  
A parallel local search algorithm for the travelling salesman problem, p. 12.

91/13 F. Rietman  
A note on Extensionality, p. 21.

91/14 P. Lemmens  
The PDB Hypermedia Package. Why and how it was built, p. 63.

91/15 A.T.M. Aerts  
K.M. van Hee  
Eldorado: Architecture of a Functional Database Management System, p. 19.

91/16 A.J.J.M. Marcelis  
An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.
| No. | Authors | Title |
|-----|---------|-------|
| 91/17 | A.T.M. Aerts, P.M.E. de Bra, K.M. van Hee | Transforming Functional Database Schemes to Relational Representations, p. 21. |
| 91/18 | Rik van Geldrop | Transformational Query Solving, p. 35. |
| 91/19 | Erik Poll | Some categorical properties for a model for second order lambda calculus with subtyping, p. 21. |
| 91/20 | A.E. Eiben, R.V. Schuwer | Knowledge Base Systems, a Formal Model, p. 21. |
| 91/21 | J. Coenen, W.-P. de Roever, J. Zwiers | Assertional Data Reification Proofs: Survey and Perspective, p. 18. |
| 91/22 | G. Wolf | Schedule Management: an Object Oriented Approach, p. 26. |
| 91/23 | K.M. van Hee, L.J. Somers, M. Voorhoeve | Z and high level Petri nets, p. 16. |
| 91/24 | A.T.M. Aerts, D. de Reus | Formal semantics for BRM with examples, p. 25. |
| 91/25 | P. Zhou, J. Hooiman, R. Kuiper | A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and completeness, p. 52. |
| 91/26 | P. de Bra, G.J. Houben, J. Paredaens | The GOOD based hypertext reference model, p. 12. |
| 91/27 | F. de Boer, C. Palamidessi | Embedding as a tool for language comparison: On the CSP hierarchy, p. 17. |
| 91/28 | F. de Boer | A compositional proof system for dynamic process creation, p. 24. |
| 91/29 | H. Ten Eikelder, R. van Geldrop | Correctness of Acceptor Schemes for Regular Languages, p. 31. |
| 91/30 | J.C.M. Baeten, F.W. Vaandrager | An Algebra for Process Creation, p. 29. |
| 91/31 | H. ten Eikelder | Some algorithms to decide the equivalence of recursive types, p. 26. |
| 91/32 | P. Struik | Techniques for designing efficient parallel programs, p. 14. |
| 91/33 | W. v.d. Aalst | The modelling and analysis of queueing systems with QNM-ExSpect, p. 23. |
| 91/34 | J. Coenen | Specifying fault tolerant programs in deontic logic, p. 15. |
| 91/35 | F.S. de Boer  
J.W. Klop  
C. Palamidessi | Asynchronous communication in process algebra, p. 20. |
| 92/01 | J. Coenen  
J. Zwiers  
W.-P. de Roever | A note on compositional refinement, p. 27. |
| 92/02 | J. Coenen  
J. Hooman | A compositional semantics for fault tolerant real-time systems, p. 18. |
| 92/03 | J.C.M. Baeten  
J.A. Bergstra | Real space process algebra, p. 42. |
| 92/04 | J.P.H.W.v.d.Eijnde | Program derivation in acyclic graphs and related problems, p. 90. |
| 92/05 | J.P.H.W.v.d.Eijnde | Conservative fixpoint functions on a graph, p. 25. |
| 92/06 | J.C.M. Baeten  
J.A. Bergstra | Discrete time process algebra, p.45. |
| 92/07 | R.P. Nederpelt | The fine-structure of lambda calculus, p. 110. |
| 92/08 | R.P. Nederpelt  
F. Kamareddine | On stepwise explicit substitution, p. 30. |
| 92/09 | R.C. Backhouse | Calculating the Warshall/Floyd path algorithm, p. 14. |
| 92/10 | P.M.P. Rambags | Composition and decomposition in a CPN model, p. 55. |
| 92/11 | R.C. Backhouse  
J.S.C.P.v.d.Woude | Demonic operators and monotype factors, p. 29. |
| 92/12 | F. Kamareddine | Set theory and nominalisation, Part I, p.26. |
| 92/13 | F. Kamareddine | Set theory and nominalisation, Part II, p.22. |
| 92/14 | J.C.M. Baeten | The total order assumption, p. 10. |
| 92/15 | F. Kamareddine | A system at the cross-roads of functional and logic programming, p.36. |
| 92/16 | R.R. Seljée | Integrity checking in deductive databases; an exposition, p.32. |
| 92/17 | W.M.P. van der Aalst | Interval timed coloured Petri nets and their analysis, p. 20. |
| 92/18 | R.Nederpelt  
F. Kamareddine | A unified approach to Type Theory through a refined lambda-calculus, p. 30. |
| 92/19 | J.C.M.Baeten  
J.A.Bergstra  
S.A.Smolka | Axiomatizing Probabilistic Processes:  
ACP with Generative Probabilities, p. 36. |
| 92/20 | F.Kamareddine | Are Types for Natural Language? P. 32. |
Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.

A useful lambda notation, p. 17.

Nominalization, Predication and Type Containment, p. 40.

Bottom-up Abstract Interpretation of Logic Programs, p. 33.

A Programming Logic for For, p. 15.

A modelling method using MOVIE and SimCon/ExSpect, p. 15.

A taxonomy of keyword pattern matching algorithms, p. 50.

Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.

A continuous version of the Prisoner’s Dilemma, p. 17

Quicksort for linked lists, p. 8.

Deterministic and randomized local search, p. 78.

A congruence theorem for structured operational semantics with predicates, p. 18.

On the unavoidability of metastable behaviour, p. 29

Exercises in Multiprogramming, p. 97

A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.

Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.

Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.

Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.

Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.

Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.
| Paper Number | Authors | Title |
|--------------|---------|-------|
| 93/14        | J.C.M. Baeten, J.A. Bergstra | On Sequential Composition, Action Prefixes and Process Prefix, p. 21. |
| 93/15        | J.C.M. Baeten, J.A. Bergstra, R.N. Bol | A Real-Time Process Logic, p. 31. |
| 93/16        | H. Schepers, I. Hoorman | A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27 |
| 93/17        | D. Alstein, P. van der Stok | Hard Real-Time Reliable Multicast in the DEDOS system, p. 19. |
| 93/18        | C. Verhoef | A congruence theorem for structured operational semantics with predicates and negative premises, p. 22. |
| 93/19        | G-J. Houben | The Design of an Online Help Facility for ExSpect, p. 21. |
| 93/20        | F.S. de Boer | A Process Algebra of Concurrent Constraint Programming, p. 15. |
| 93/21        | M. Codish, D. Dams, G. Filé, M. Bruynooghe | Freeness Analysis for Logic Programs - And Correctness?, p. 24. |
| 93/22        | E. Poll | A Typechecker for Bijective Pure Type Systems, p. 28. |
| 93/23        | E. de Kogel | Relational Algebra and Equational Proofs, p. 23. |
| 93/24        | E. Poll and Paula Severi | Pure Type Systems with Definitions, p. 38. |
| 93/25        | H. Schepers and R. Gerth | A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31. |
| 93/26        | W.M.P. van der Aalst | Multi-dimensional Petri nets, p. 25. |
| 93/27        | T. Kloks and D. Kratsch | Finding all minimal separators of a graph, p. 11. |
| 93/28        | F. Kamareddine and R. Nederpelt | A Semantics for a fine λ-calculus with de Bruijn indices, p. 49. |
| 93/29        | R. Post and P. De Bra | GOLD, a Graph Oriented Language for Databases, p. 42. |
| 93/30        | J. Deogun, T. Kloks, D. Kratsch, H. Müller | On Vertex Ranking for Permutation and Other Graphs, p. 11. |
| 93/31        | W. Körver | Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40. |
| 93/32        | H. ten Eikelder and H. van Geldrop | On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17. |
| 93/33        | L. Loyens and J. Moonen | ILIAS, a sequential language for parallel matrix computations, p. 20. |
93/34 J.C.M. Baeten and J.A. Bergstra  Real Time Process Algebra with Infinitesimals, p.39.

93/35 W. Ferrer and P. Severi  Abstract Reduction and Topology, p. 28.

93/36 J.C.M. Baeten and J.A. Bergstra  Non Interleaving Process Algebra, p. 17.

93/37 J. Brunekreef J-P. Katoen R. Koymans S. Mauw  Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.

93/38 C. Verhoef  A general conservative extension theorem in process algebra, p. 17.

93/39 W.P.M. Nuijten E.H.L. Aarts D.A.A. van Erp Taalman Kip K.M. van Hee  Job Shop Scheduling by Constraint Satisfaction, p. 22.

93/40 P.D.V. van der Stok M.M,M.P.J. Claessen D. Alstein  A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.

93/41 A. Bijlsma  Temporal operators viewed as predicate transformers, p. 11.

93/42 P.M.P. Rambags  Automatic Verification of Regular Protocols in P/T Nets, p. 23.

93/43 B.W. Watson  A taxonomy of finite automata construction algorithms, p. 87.

93/44 B.W. Watson  A taxonomy of finite automata minimization algorithms, p. 23.

93/45 E.J. Luit J.M.M. Martin  A precise clock synchronization protocol, p.

93/46 T. Kloks D. Kratsch J. Spinrad  Treewidth and Pathwidth of Cocomparability graphs of Bounded Dimension, p. 14.

93/47 W. v.d. Aalst P. De Bra G.J. Houben Y. Komatzky  Browsing Semantics in the "Tower" Model, p. 19.

93/48 R. Gerth  Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.

94/01 P. America M. van der Kammen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart  The object-oriented paradigm, p. 28.
| 94/02 | F. Kamareddine         | Canonical typing and Π-conversion, p. 51.          |
|       | R.P. Nederpelt         |                                                   |
| 94/03 | L.B. Hartman           | Application of Markov Decision Processes to Search|
|       | K.M. van Hee           | Problems, p. 21.                                  |
| 94/04 | J.C.M. Baeten          | Graph Isomorphism Models for Non Interleaving Process |
|       | J.A. Bergstra          | Algebra, p. 18.                                   |
| 94/05 | P. Zhou                | Formal Specification and Compositional Verification |
|       | J. Hooman              | of an Atomic Broadcast Protocol, p. 22.            |
| 94/06 | T. Basten              | Time and the Order of Abstract Events in Distributed |
|       | T. Kunz                | Computations, p. 29.                              |
|       | J. Black               |                                                   |
|       | M. Coffin              |                                                   |
|       | D. Taylor              |                                                   |
| 94/07 | K.R. Apt               | Logic Programming and Negation: A Survey, p. 62.   |
|       | R. Bol                 |                                                   |
| 94/08 | O.S. van Roosmalen     | A Hierarchical Diagrammatic Representation of Class |
|       |                       | Structure, p. 22.                                 |
| 94/09 | J.C.M. Baeten          | Process Algebra with Partial Choice, p. 16.       |
|       | J.A. Bergstra          |                                                   |
| 94/10 | T. Verhoeff            | The testing Paradigm Applied to Network Structure.|
|       |                       | p. 31.                                            |
| 94/11 | J. Peleska             | A Comparison of Ward & Mellor's Transformation    |
|       | C. Huizing             | Schema with State- & Activitycharts, p. 30.        |
|       | C. Petersohn           |                                                   |
| 94/12 | T. Kloks               | Dominoes, p. 14.                                  |
|       | D. Kratsch             |                                                   |
|       | H. Müller              |                                                   |
| 94/13 | R. Seljée              | A New Method for Integrity Constraint checking in Deductive Databases, p. 34. |
| 94/14 | W. Peremans            | Ups and Downs of Type Theory, p. 9.               |
| 94/15 | R.J.M. Vaessens        | Job Shop Scheduling by Local Search, p. 21.       |
|       | E.H.L. Aarts           |                                                   |
|       | J.K. Lenstra           |                                                   |
| 94/16 | R.C. Backhouse         | Mathematical Induction Made Calculational, p. 36.  |
|       | H. Doombos             |                                                   |
| 94/17 | S. Mauw                | An Algebraic Semantics of Basic Message Sequence Charts, p. 9. |
|       | M.A. Reniers           |                                                   |
| 94/18 | F. Kamareddine         | Refining Reduction in the Lambda Calculus, p. 15.  |
|       | R. Nederpelt           |                                                   |
| 94/19 | B.W. Watson            | The performance of single-keyword and multiple-    |
|       |                       | keyword pattern matching algorithms, p. 46.        |
94/20 R. Bloo
F. Kamareddine
R. Nederpelt

Beyond $\beta$-Reduction in Church's $\lambda \to$, p. 22.

94/21 B.W. Watson

An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular Expressions.

94/22 B.W. Watson

The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.

94/23 S. Mauw and M.A. Reniers

An algebraic semantics of Message Sequence Charts, p. 43.

94/24 D. Dams
O. Grumberg
R. Gerth

Abstract Interpretation of Reactive Systems:
Abstractions Preserving $\forall$CFL*, $\exists$CFL* and $\text{CTL}^*$, p. 28.

94/25 T. Kloks

$K_{1,3}$-free and $W_4$-free graphs, p. 10.

94/26 R.R. Hoogerwoord

On the foundations of functional programming: a programmer's point of view, p. 54.

94/27 S. Mauw and H. Mulder

Regularity of BPA-Systems is Decidable, p. 14.

94/28 C.W.A.M. van Overveld
M. Verhoeven

Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling, p. 20.

94/29 J. Hooman

Correctness of Real Time Systems by Construction, p. 22.

94/30 J.C.M. Baeten
J.A. Bergstra
Gh. Ştefănescu

Process Algebra with Feedback, p. 22.

94/31 B.W. Watson
R.E. Watson

A Boyer-Moore type algorithm for regular expression pattern matching, p. 22.

94/32 J.J. Vereijken

Fischer's Protocol in Timed Process Algebra, p. 38.

94/33 T. Laan

A formalization of the Ramified Type Theory, p.40.

94/34 R. Bloo
F. Kamareddine
R. Nederpelt

The Barendregt Cube with Definitions and Generalised Reduction, p. 37.

94/35 J.C.M. Baeten
S. Mauw

Delayed choice: an operator for joining Message Sequence Charts, p. 15.

94/36 F. Kamareddine
R. Nederpelt

Canonical typing and $\Pi$-conversion in the Barendregt Cube, p. 19.

94/37 T. Basten
R. Bol
M. Voorhoeve

Simulating and Analyzing Railway Interlockings in ExSpect, p. 30.

94/38 A. Bijlsma
C.S. Scholten

Point-free substitution, p. 10.
| ID   | Authors                          | Title                                                                 |
|------|----------------------------------|----------------------------------------------------------------------|
| 94/39| A. Blokhuis T. Kloks            | On the equivalence covering number of splitgraphs, p. 4.             |
| 94/40| D. Alstein                      | Distributed Consensus and Hard Real-Time Systems, p. 34.            |
| 94/41| T. Kloks D. Kratsch             | Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6. |
| 94/42| I. Engelfriet J.J. Vereijken    | Concatenation of Graphs, p. 7.                                       |
| 94/43| R.C. Backhouse M. Bijsterveld   | Category Theory as Coherently Constructive Lattice M Theory: An Illustration, p. 35. |
| 94/44| E. Brinksma J. Davies R. Gerth  | Verifying Sequentially Consistent Memory, p. 160                    |
|      | S. Graf W. Janssen B. Jonsson   |                                                                       |
|      | S. Katz G. Lowe M. Poel A. Pnueli|                                                                       |
|      | C. Rump J. Zwiers               |                                                                       |
| 94/45| G.J. Houben                      | 'Tutorial voor de ExSpect-bibliotheek voor "Administratieve Logistiek", p. 43. |
| 94/46| R. Bloo F. Kamareddine R. Nederpel| The λ-cube with classes of terms modulo conversion, p. 16.        |
| 94/47| R. Bloo F. Kamareddine R. Nederpel| On Π-conversion in Type Theory, p. 12.                              |
| 94/48| Mathematics of Program Construction Group | Fixed-Point Calculus, p. 11.                                         |
| 94/49| J.C.M. Baeten J.A. Bergstra     | Process Algebra with Propositional Signals, p. 25.                  |