Uniform Logical Relations

Edwin Westbrook
Department of Computer Science
Rice University
Houston, TX 77005
Email: emw4@rice.edu

April 1, 2011

Abstract

Strong Normalization (SN) is an important property for intensional constructive type theories such as the Calculus of Inductive Constructions (CiC), the basis for the Coq theorem prover. Not only does SN imply consistency, but it also ensures that type-checking is decidable, and further, it provides a straightforward model, the term model, for a theory. Unfortunately, although SN has been proved for fragments of CiC, it is not known how to prove SN for CiC in its entirety, including eliminations for large inductive types as well as higher predicative universes. In this work, we show how to prove SN for full CiC. They key insight given here is that terms must be interpreted in a *uniform* manner, meaning that the form of the interpretation of a term must not depend on whether the term is a type. We introduce a new technique called Uniform Logical Relations, With uniformity as a guiding principle, and we show that this technique can then be used to prove SN for CiC. An important property of our technique is that it does not rely on Confluence, and thus it could potentially be used for extensions of CiC with added computation rules, such as Extensionality, for which Confluence relies on SN.

1 Introduction

Intensional constructive type theory (ICTT), as first introduced by Martin-Löf [13], is compelling because it gives a combined programming language and mathematical theory. This allows users to write programs and proofs about those programs in the same language (see e.g. [16]). Further, theories based on ICTT are defined entirely syntactically. This makes them foundational, in the sense that they can be defined with little or no appeal to models or set theory.

An important consideration for such theories is Strong Normalization (SN), which states that no well-typed term of the theory has an infinite reduction path. Stated differently, SN says that the programs of a theory have no infinite loops. SN is useful for a number of reasons. First, it ensures the consistency of a theory, as it implies that every proof has a cut-free version. Second, it ensures that type-checking is decidable. Finally, SN also gives a simple model of the theory, namely, the normal forms of that theory.

Unfortunately, it is not known how to prove SN for more powerful extensions of ICTT such as the Calculus of Inductive Constructions (CiC), the basis of the Coq theorem prover [18]. Although there have been numerous proofs of SN for fragments of CiC [8, 11, 2, 9, 12, 4], none of these approaches has been generalized to the entirety of CiC. This has lead to research into other approaches to proving the consistency

of CiC by providing set-theoretic models [3, 20, 15, 14, 19, 17]. The key difficulty is in combining two features: an impredicative universe Prop, and a cumulative type hierarchy of universes Type_i for $i \geq 0$. To handle impredicativity requires the Logical Relations technique introduced by Girard for System F [10], in which each type is interpreted as a logically defined set of terms, also called a logical relation. With a cumulative type hierarchy, however, we do not know whether a type is a sort or not unless we normalize it, and thus, under the standard approach to Logical Relations, it is unknown how to even interpret a type without normalizing it. This seems to require SN to prove SN! The problem is especially difficult in the presence of inductive types (as pointed out by e.g. [8]), since the form of a type can depend on whether a natural number term evaluates to 0 or to 1. Although Luo showed how to break this cycle in the case of ECC using a difficult and complex quasi-normalization theorem [12], ECC contains only products, not inductive types, and it is not clear that this approach generalizes to theories that do contain inductive types.

In this paper, we show how to solve this problem by using an interpretation that is *uniform*, meaning that the interpretation does not need to know whether a type is a sort.[1] We call this approach Uniform Logical Relations. The key technical development that ensures uniformity is to interpret types as sets of pairs of terms and interpretations, i.e., sets of the form $\{ \langle M,\ I\rangle \mid \ \phi(M, I) \}$ where M is a term and ϕ is a logical formula that states that I is a valid interpretation of M. This is as opposed to the usual approach of interpreting types as sets of terms $\{ M \mid \ \phi(M) \}$. Under the usual approach, the interpretation of the function type $\Pi x : A\,.\,B$ must quantify over terms of type A and reducibility candidates when A is a sort, over terms of type A and functions on reducibility candidates when A is a kind-level Π-type, and over just terms of type A when A is a type. This information about A requires the normal form of A in general. Under Uniform Logical Relations, in contrast, the interpretation of $\Pi x : A\,.\,B$ always quantifies over all pairs $\langle N,\ X\rangle \in [\![A]\!]$. This uniformity means we do not require the normal form of A.

In addition to being the first proof technique to show SN for the full generality of CiC, Uniform Logical Relations also lends a certain amount of elegance to the proof. Uniformity means that the interpretation does not have any case splits on whether a term is a type, whether it is a kind, etc. Uniform Logical Relations also requires only one main theorem, Reducibility; important lemmas required by other methods, such as the fact that the interpretations of types are valid reducibility candidates or saturated sets, become special cases of Reducibility. Further, the proof of Reducibility under Uniform Logical Relations almost directly matches the original Logical Relations proof given by Girard for System F, which is simpler than later approaches that have been applied to fragments of CiC. Another interesting aspect of Uniform Logical Relations is that Proof Irrelevance, where all terms of a given type A in Prop have equal interpretations, is in fact a necessary part of the interpretation to ensure that it is well-founded.

The remainder of this document is organized as follows. Section 2 reviews the syntax and semantics of the Calculus of Inductive Constructions (CiC). Section 3 gives the "backbone" of our interpretation by defining a cumulative hierarchy of sets in ZFC to interpret the type universes. Section 4 then uses this backbone to prove prove Strong Normalization for CiC using Uniform Logical Relations. Finally, Section 5 relates Uniform Logical Relations to previous approaches, and Section 6 concludes.

2 The Calculus of Inductive Constructions (CiC)

In this section, we give a brief overview of the Calculus of Inductive Constructions. The version we give here has some small modifications relative to the theory as presented in, for example, the Coq manual [18]. These

[1] This approach came out of the author's dissertation, which gave an interpretation of a nominal extension of CiC into CiC [21]; the interpretation for CiC in set theory given here is actually much simpler because we do not have to worry things like intensional equality and type universes.

$$
\begin{array}{lcl}
\Sigma & ::= & \cdot \mid \Sigma, c : M \mid \Sigma, a : M \\
\Gamma & ::= & \cdot \mid \Gamma, x : M \mid \Gamma, u :^{n;\vec{x}} M \\
M & ::= & \mathsf{Prop} \mid \mathsf{Type}_i \mid \Pi x{:}M\,.\,M \mid a(\vec{M};\vec{M}) \mid x \mid M\ M \mid \lambda\, x{:}M\,.\,M \mid c(\vec{M};\vec{M}) \\
 & & \mid u \mid \mathsf{fun}^{ip}\ u\ (\Gamma)\ (c\backslash\Gamma \to M \mid \ \ldots\ \mid c\backslash\Gamma \to M)
\end{array}
$$

Figure 1: Syntax of CiC

modifications are for increased conciseness of the definitions and proofs given below, but do not change the theory in a material way. Note that we do omit predicative eliminations of impredicative inductive types that are allowed in Coq, including eliminations on empty types and on some types with only one constructor, but see Section 6 for a brief discussion of how these could be handled.

A Note on Syntax: Here and in the below we use particular letters to stand for elements of particular syntactic classes; e.g., M is always used for a term. These can appear with subscripts or primes, as in M_2' or M_{fun}. We also use an arrow over a syntactic construct to indicate a sequence of zero or more of those constructs. For example, $\vec{x}$ denotes a sequence of variables. We use subscripts i and j to denote the ith and jth elements of this sequence, e.g., x_i, and we use vertical bars $|\vec{x}|$ to denote the length of this sequence. If the letter under the arrow has a prime, then so do the elements of the sequence, so for instance the elements of $\vec{M}'$ are referred to as M_i'. We also use the following compound notations: $M\ \vec{M}$ denotes the multi-arity application $M\ M_1 \ldots M_{|\vec{M}|}$; $[\vec{M}/\Gamma]$ denotes the multi-arity substitution $[M_1/x_1, \ldots, M_{|\vec{M}|}/x_{|\vec{M}|}]$, where $\vec{x}$ are the variables on the left in Γ; and $\vec{c}\backslash\vec{\Gamma} \to \vec{M}$ denotes the sequence of pattern cases $c_1\backslash\Gamma_1 \to M_1 \mid \ldots \mid c_1\backslash\Gamma_1 \to M_1$

We assume mutually disjoint sets of constructors c, type constructors a, variables x, and recursive variables u, where the latter are used for recursive calls in primitive recursive pattern-matching functions and the sets of variables and recursive variables are infinite. We also use y and z for variables.

Figure 1 gives the syntax of CiC. The first line defines the signatures Σ that map constructors c and type constructors a to types. The second line defines the contexts Γ which map variables x and recursive variables u to types. Recursive variables u are also associated with a number n and a sequence of variables $\vec{x}$; this specifies that all occurrences of u should have the form $u\ \vec{M}\ x_i$ for some i where $|\vec{M}| = n$. This is used to ensure that recursive calls in pattern-matching functions are strictly primitive recursive. In the below we write $|\Gamma|$ for the length of Γ, i.e., the number of commas.

The remainder of Figure 1 defines the terms M. We also use N, Q, and R for terms, as well as A and B for terms that are meant to denote types. The terms include the impredicative universe Prop and the predicative universes Type_i for all $i \geq 0$. These are the *sorts*. We use s to denote sorts, using s^{p} and s^{i} to denote predicative (i.e., Type_i for some i) and impredicative sorts (i.e., Prop), respectively. We also sometimes use ip for meta-variable that varies over $\{\mathrm{p}, \mathrm{i}\}$ so, for example, s_1^{ip}, s_2^{ip} denotes a pair of sorts that are both either predicative or impredicative.

The next term constructs in the figure are the function types $\Pi x{:}A\,.\,B$ and the inductive types $a(\vec{M};\vec{N})$ as terms. The latter form distinguishes between the *parameters* $\vec{M}$ and the *arguments* $\vec{N}$ of the inductive type. Next are the variables x, the applications $M\ N$, the λ-abstractions $\lambda\, x : A\,.\,M$, and the constructor applications $c(\vec{M};\vec{N})$ as terms, where the latter again distinguishes between the parameters and the arguments.

The final line lists the recursive variables u and the pattern-matching functions, where the latter have the form $\mathsf{fun}^{ip}\ u\ (\Gamma_{\mathsf{arg}})\ (\vec{c}\backslash\vec{\Gamma} \to \vec{M})$. Pattern-matching functions combine the `Fix` and `case` terms of Coq. Each compound form $c_i\backslash\Gamma_i \to M_i$ is called a *pattern case* with *pattern context* Γ_i and *body* M_i. Pattern-matching

$$\begin{array}{rcl}(\lambda\, x{:}A\,.\,M)\ N & \longrightarrow & [N/x]M \\ (M_{\mathrm{fun}} = \mathsf{fun}\ u\ (\Gamma_{\mathsf{arg}})\ (\vec{c}\backslash\vec{\Gamma} \to \vec{M}))\ \vec{N}\ c_i(\vec{Q};\vec{R}) & \longrightarrow & [\vec{N}/\Gamma_{\mathsf{arg}}, \vec{R}/\Gamma_i, M_{\mathrm{fun}}/u]M_i\end{array}$$

Figure 2: Operational Semantics of CiC

functions of this form take in $|\Gamma_{\mathsf{arg}}|+1$ arguments and pattern-match on the last argument. The last argument is called the *scrutinee* and the earlier arguments are called the *parameters*. If the scrutinee has the form $c_i(\vec{N};\vec{Q})$ then the pattern-matching function returns M_i, substituting the parameters for the variables of Γ_{arg} and substituting the arguments $\vec{Q}$ for the pattern variables listed in Γ_i. Pattern-matching functions are also annotated with either p or i, indicating whether they eliminate a predicative or impredicative inductive type.

The operational semantics of CiC is given as a higher-order rewrite system $\longrightarrow$ in Figure 2. This means that $M_1 \longrightarrow M_2$ iff M_2 can be obtained by replacing a subterm of M_1 that matches the left-hand side of a rule listed in Figure 2 be the corresponding right-hand side of the rule. The first rule performs the usual β-reduction, while the second performs the pattern-matching reduction described in the previous paragraph. In the below, we write $\longrightarrow^*$, $\longleftrightarrow$, and $\longleftrightarrow^*$ respectively for the reflexive-transitive closure, the reflexive-symmetric closure, and the reflexive-symmetric-transitive closure of $\longrightarrow$.

We do not give the well-formedness rules for signatures Σ; these may be found in the Coq manual [18]. At a high level, however, these require that Σ contain sequences that define a type constructor a and its constructors c_i of the following form:

$$a : (\Pi\Gamma_{\mathsf{p}}\,.\,\Pi\Gamma_{\mathsf{a}}\,.\,s), \ldots, c_i : (\Pi\Gamma_{\mathsf{p}}\,.\,\Pi\Gamma_{\mathsf{c}_i}[i]\,.\,a(\Gamma_{\mathsf{p}};\vec{M_c})), \ldots$$

The context Γ_{p} lists the *parameters* of the inductive type a, which must be the same for for all of the constructors. The contexts Γ_{a} and $\Gamma_{\mathsf{c}_i}[i]$ list the *arguments* of a and c_i, respectively. All of the types must be well-typed for the prefix of Σ before a, except that the $\Gamma_{\mathsf{c}_i}[i]$ may contain a *strictly positively*, meaning that types of the form $a(\vec{N};\vec{Q})$ can occur directly to the right of a colon in $\Gamma_{\mathsf{c}_i}[i]$. The signature Σ and its well-formedness are left implicit in the below.

Well-formedness for contexts is given by the judgment $\vdash \Gamma$, defined as follows:

$$\frac{}{\vdash \cdot} \quad \frac{\vdash \Gamma \quad \Gamma \vdash A : s}{\vdash \Gamma, x : A} \quad \frac{\vdash \Gamma \quad \Gamma \vdash A : s}{\vdash \Gamma, u :^{n;\vec{x}} A}$$

These rules require that each type A listed in Γ is well-typed at some sort s in the prefix of Γ before A. We implicitly assume well-formedness of all contexts used in the typing rules below.

The typing rules are given in Figure 3. The first rule is the conversion rule, which states that equal types have the same elements. Note that this only allows a single step of conversion; although the rule may be used multiple times, each use requires the destination type A' to be well-typed, ensuring that only conversions that use well-typed types can be used. The second rule is the subtyping rule, which states that if M has type A for A a *subtype* of A', written $A <: A'$, then M has type A'. Subtyping is the reflexive-transitive closure of the following three cases: $\mathsf{Prop} <: \mathsf{Type}_0$; $\mathsf{Type}_i <: \mathsf{Type}_{i+1}$; and if $B_1 <: B_2$ then $\Pi x{:}A\,.\,B_1 <: \Pi x{:}A\,.\,B_2$. Note that subtyping is guaranteed to preserve well-typedness of types. The next two rules state that $\mathsf{Prop} : \mathsf{Type}_0$ and $\mathsf{Type}_i : \mathsf{Type}_{i+1}$. The next two rules type Π-types $\Pi x : A\,.\,B$ by requiring the types of both A and B to be sorts. Also, if B is impredicative, meaning it has type Prop, then the type of $\Pi x{:}A\,.\,B$ is Prop, and otherwise the type of $\Pi x{:}A\,.\,B$ is the maximum universe of that of both A and B.

For inductive types $a(\vec{M};\vec{N})$, the parameters $\vec{M}$ and the arguments $\vec{N}$ must have the types given by the contexts Γ_{p} and Γ_{a} in Σ. This is stated with the judgment $\Gamma \vdash (\vec{M};\vec{N}) : \Gamma_{\mathsf{p}};\Gamma_{\mathsf{a}}$ which states that $|\vec{M}| = |\Gamma_{\mathsf{p}}|$,

$$\frac{\Gamma \vdash M : A \quad A \longleftrightarrow A' \quad \Gamma \vdash A' : s}{\Gamma \vdash M : A'} \qquad \frac{\Gamma \vdash M : A \quad A <: A'}{\Gamma \vdash M : A'} \qquad \frac{}{\Gamma \vdash \mathsf{Prop} : \mathsf{Type}_0} \qquad \frac{}{\Gamma \vdash \mathsf{Type}_i : \mathsf{Type}_{i+1}}$$

$$\frac{\Gamma \vdash A : \mathsf{Type}_i \quad \Gamma, x : A \vdash B : \mathsf{Type}_i}{\Gamma \vdash \Pi x{:}A \,.\, B : \mathsf{Type}_i} \qquad \frac{\Gamma \vdash A : s \quad \Gamma, x : A \vdash B : \mathsf{Prop}}{\Gamma \vdash \Pi x{:}A \,.\, B : \mathsf{Prop}} \qquad \frac{\Gamma \vdash A : s \quad \Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda\, x{:}A \,.\, M : \Pi x{:}A \,.\, B}$$

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : \Pi x{:}A \,.\, B \quad \Gamma \vdash N : A}{\Gamma \vdash M\ N : [N/x]B} \qquad \frac{a : \Pi\Gamma_\mathsf{p} \,.\, \Pi\Gamma_\mathsf{a} \,.\, s \in \Sigma \quad \Gamma \vdash (\vec{M}; \vec{N}) : \Gamma_\mathsf{p}; \Gamma_\mathsf{a}}{\Gamma \vdash a(\vec{M}; \vec{N}) : s}$$

$$\frac{c : \Pi\Gamma_\mathsf{p} \,.\, \Pi\Gamma_\mathsf{c} \,.\, a(\Gamma_\mathsf{p}; \vec{Q}) \in \Sigma \quad \Gamma \vdash (\vec{M}; \vec{N}) : \Gamma_\mathsf{p}; \Gamma_\mathsf{c}}{\Gamma \vdash c(\vec{M}; \vec{N}) : a(\vec{M}; [\vec{M}/\Gamma_\mathsf{p}, \vec{N}/\Gamma_\mathsf{c}]\vec{Q})} \qquad \frac{(u :^{|\vec{M}|;\vec{x}} \Pi\Gamma_u \,.\, B) \in \Gamma \quad \Gamma \vdash (\vec{M}, (x_i\ \vec{N})) : \Gamma_u}{\Gamma \vdash u\ \vec{M}\ (x_i\ \vec{N}) : [(\vec{M}, (x_i\ \vec{N}))/\Gamma_u]B}$$

$$\frac{\begin{array}{l}\Gamma \vdash A_{\text{fun}} : s \quad a : \Pi\Gamma_\mathsf{p} \,.\, \Pi\Gamma_\mathsf{a} \,.\, s_a^{ip} \in \Sigma \quad \textbf{elim-ok}(a \mapsto s) \quad \vec{c} = \text{ctors}_\Sigma(a) \quad \forall i.(\Gamma, \Gamma_\mathsf{arg} \vdash \Gamma_i) \\ \forall i.(\Gamma, \Gamma_\mathsf{arg}, \Gamma_i \vdash c_i(\vec{N}; \Gamma_i) : a(\vec{N}; \vec{Q})) \qquad \forall i.(\Gamma, \Gamma_\mathsf{arg}, \Gamma_i, u :^{|\Gamma_\mathsf{arg}|;\Gamma_i} \vdash M_i : [c_i(\vec{N}; \Gamma_i)/x]B)\end{array}}{\Gamma \vdash \mathsf{fun}^{ip}\ u\ (\Gamma_\mathsf{arg})\ (\vec{c} \backslash \vec{\Gamma} \to \vec{M}) : (A_{\text{fun}} = \Pi\Gamma_\mathsf{arg} \,.\, \Pi x{:}a(\vec{N}; \vec{Q}) \,.\, B)}$$

Figure 3: Typing Rules of CiC

$|\vec{N}| = |\Gamma_\mathsf{a}|$, and that, for each term Q in $\vec{M}, \vec{N}$ with type A occurring at the same position in $\Gamma_\mathsf{p}, \Gamma_\mathsf{a}$, we have $\Gamma \vdash Q : [\vec{M}/\Gamma_\mathsf{p}, \vec{N}/\Gamma_\mathsf{a}]A$.

For variables, we look the type up in the context. For applications $M\ N$, we require M to have function type $\Pi x : A \,.\, B$ and N to have type A, returning type $[N/x]B$. For λ-abstractions, we require the domain type A to be well-formed, the body M to have type B under the context extended with variable x, and we return function type $\Pi x : A \,.\, B$. For constructor applications $c(\vec{M}; \vec{N})$, we again require the parameters $\vec{M}$ and arguments $\vec{N}$ to have the appropriate types given in Γ_p and Γ_c, yielding the return type of c with the parameters and arguments substituted in. For recursive calls to u, we require the term to be of the form $u\ \vec{M}\ x_i$ where x_i is one of the variables to which u may be applied, where $|\vec{M}|$ is the number of arguments before x_i to which u must be applied, and where all of $\vec{M}$ and x_i have the types required by the type of u.

To type a pattern-matching function, we first require that its type A_{fun} (which includes the parameter types Γ_arg) has type s for some sort s. We then require that a can be eliminated at sort s, written $\textbf{elim-ok}(a \mapsto s)$; this states that, if a is impredicative then $s = \mathsf{Prop}$, except for the special case that a has at most one constructor, all of whose arguments are proofs (i.e., have a type which has type Prop), when s is allowed to be predicative. Next, we require that the patterns given exactly match the constructors of a in Σ, where the latter is written $\text{ctors}_\Sigma(a)$. Next, we require that the pattern contexts Γ_i are all well-formed. Finally, we require that each c_i applied to the given parameters and the variables listed in Γ_i has same type as the input type, and that each body M_i has as its type the result of substituting this application of c_i into the return type B.

2.1 Call-by-Name Reduction

In order to prove reducibility of $\lambda\, x{:}A \,.\, M$ below, we will need to prove that $(\lambda\, x{:}A \,.\, M)\ N$ is reducible for all reducible N. Since reducibility is proved by induction on the structure of terms, we get by assumption that

$[N/x]M$ is reducible; to proceed, we need to "undo" a step of reduction.[2] For pattern-matching functions, however, the situation is more complex: if M_{fun} is a pattern-matching function with cases M_i, and if we have by the inductive hypothesis that $[\vec{N}/\Gamma_{\mathsf{arg}}, \vec{Q}/\Gamma_i]M_i$ is reducible for all reducible $\vec{N}$ and $\vec{Q}$, then undoing a step of reduction only yields that $M_{\text{fun}}\ \vec{N}\ N$ is reducible for N of the form $c(\vec{R}; \vec{Q})$. In other words, we might need to undo multiple reductions on the scrutinee argument N. We only need to undo reductions on the path to a weak head normal form of N, however. Call-by-name (CBN) reduction gives a disciplined way to talk about exactly these sorts of reductions.

In the remainder of this section, we introduce CBN reduction and prove some properties of CBN that will be used below. More specifically, we deal with the reduction relation $\longrightarrow_{\text{n-nc}}$ which only performs CBN reductions when the immediate subterms of the redex involved are all strongly normalizing; otherwise we could have $M \longrightarrow_{\text{n}} N$ via a redex that "throws away" a subterm of M, so $\mathsf{SN}(N)$ does not imply $\mathsf{SN}(M)$. We then prove that undoing $\longrightarrow_{\text{n-nc}}$ preserves SN (Lemma 4), and also that $\longrightarrow_{\text{n-nc}}$ satisfies a weak Standardization property for reduction to weak head normal forms (Lemma 4). These both follow from Lemma 2, which shows that $\longrightarrow_{\text{n-nc}}$ weakly commutes with standard reduction. Note that none of this material should be considered surprising or novel; Lemma 2, for example, is proved as Lemma 3.5.2 by Altenkirch [2], where CBN reduction is called weak head reduction.

Definition 1 (CBN Reduction) *We say that M* call-by-name (CBN) reduces *to M', written $M \longrightarrow_{\text{n}} M'$, iff the following hold:*

1. *If M is a redex, then M' is the result of contracting M;*
2. *If $M \equiv M_{\text{fun}}\ \vec{N}\ N$ such that M is not a redex and M_{fun} is a pattern-matching function with $|\vec{N}|$ parameters, and if $N \longrightarrow_{\text{n}} N'$, then $M' \equiv M_{\text{fun}}\ \vec{N}\ N'$; and*
3. *If $M \equiv M_1\ M_2$ for M not of one of the above forms, and if $M_1 \longrightarrow_{\text{n}} M_1'$, then $M' \equiv M_1'\ M_2$.*

If none of these apply, then there is no M' such that $M \longrightarrow_{\text{n}} M'$, and M is called a weak head normal form *(WHNF). When $M \longrightarrow_{\text{n}} M'$ holds, by contracting a redex R in M, we call the immediate subterms of R the* constituents *of the reduction, and we further say that M call-by-name reduces to M'* with normalizing constituents*, written $M \longrightarrow_{\text{n-nc}} M'$, if all these constituents are strongly normalizing.*

Lemma 1 *If $M \longrightarrow_{\text{n-nc}} M'$ then $M\ N \longrightarrow_{\text{n-nc}} M'\ N$.*

Proof: If $M \longrightarrow_{\text{n-nc}} M'$ then M must be of the form $Q\ \vec{R}$ where Q is either a redex or matches case 2 of Definition 1. In either case, $M\ N$ cannot be a redex, nor can it match case 2 of Definition 1, and so by case 3 of Definition 1 we have that $M\ N \longrightarrow_{\text{n}} M'\ N$. In addition, this reduction has the same normalizing constituents as the original CBN reduction, so $M\ N \longrightarrow_{\text{n-nc}} M'\ N$. □

Lemma 2 *If $M \longrightarrow_{\text{n-nc}} N$ and $M \longrightarrow M'$, then either $M' \equiv N$ or there exists N' such that $M' \longrightarrow_{\text{n-nc}} N'$ and $N \longrightarrow^* N'$, where the latter can be strengthened to $N \longrightarrow N'$ when the reduction $M \longrightarrow M'$ contracts a redex of M that is not a subterm of a constituent of the reduction $M \longrightarrow_{\text{n-nc}} N$.*

Proof: Let R be the redex contracted by the reduction $M \longrightarrow_{\text{n-nc}} N$ and R' be that contracted by the reduction $M \longrightarrow M'$. We now consider four cases. If R and R' are the same redex, then $M' \equiv N$. If neither R nor R' is a subterm of the other, meaning that R' is not a constituent of $M \longrightarrow_{\text{n-nc}} N$, then immediately

[2] This "undoing" is used in a similar manner as the condition **CR 3** in Girard's original proof [10], and also has a counterpart in approaches based on saturated sets, such as that of Altenkirch [2].

we have that R exists in M' and R' exists as a CBN redex in N, and reducing either yields an N' such that $M' \longrightarrow_{\mathrm{n-nc}} N'$ and $N \longrightarrow N'$. It is not possible for R to be a strict subterm of R', since by definition call-by-name reduction does not contract strict subterms of redexes.

In the final case, R' is a strict subterm of R. Let Q' be the result of contracting R', and let Q be the result of replacing the subterm R' of R with Q'. By contracting Q in M' we get an N' such that $M' \longrightarrow_{\mathrm{n}} N'$. Further, since reductions inside a constituent cannot cause it to no longer be strongly normalizing, we see that $M' \longrightarrow_{\mathrm{n-nc}} N'$. In addition, by straightforward structural reasoning it is possible to conclude that the reduction $M \longrightarrow_{\mathrm{n-nc}} N$ replaces zero or more copies of R' exactly where the reduction $M' \longrightarrow_{\mathrm{n-nc}} N'$ places zero or more copies of Q', and that otherwise N and N' are the same, whereby we get that $N \longrightarrow^* N'$. □

Lemma 3 *If* $M \longrightarrow_{\mathrm{n-nc}} N$ *and* $\mathsf{SN}(N)$ *then* $\mathsf{SN}(M)$.

Proof: We prove that $\mathsf{SN}(M')$ for all M' such that $M \longrightarrow M'$, whereby the result immediately follows, and we proceed by induction on the sum of $\mathbf{size}(N)$ and of $\mathbf{size}(Q_i)$ for the constituents of the reduction $M \longrightarrow_{\mathrm{n-nc}} N$. Using Lemma 2, we have three cases: $M' \equiv N$, where the result is immediate; M' results from reduction in a constituent of the reduction $M \longrightarrow_{\mathrm{n-nc}} N$, where the result follows by the induction hypothesis on $\mathbf{size}(Q_i)$ for the constituent Q_i of the reduction; or M' results from reduction not in a constituent, where the result follows by the induction hypothesis on $\mathbf{size}(N)$. For the latter two cases, it is straightforward to see that the reduction $M \longrightarrow M'$ cannot increase $\mathbf{size}(N)$ or $\mathbf{size}(Q_i)$ for any Q_i, and that both cases reduce at least one of these. □

Lemma 4 *If* $M \longrightarrow_{\mathrm{n-nc}} N$ *and if* $M \longrightarrow^* Q$ *for weak head normal form* Q*, then* $N \longrightarrow^* Q$.

Proof: We use the weaker assumption that $M \longrightarrow_{\mathrm{n}} N$, and proceed by induction on the number of steps in $M \longrightarrow^* Q$. There is no base case for $M \equiv Q$, since $M \longrightarrow_{\mathrm{n}} N$ precludes the possibility that M is a WHNF. Thus we have $M \longrightarrow M' \longrightarrow^* Q$ for some M'. By Lemma 2 we either have $M' \equiv N$, and the result is immediate, or $M' \longrightarrow_{\mathrm{n}} N'$ and $N \longrightarrow^* N'$ for some N'. By the inductive hypothesis we then have that $N' \longrightarrow^* Q$, and so $N \longrightarrow^* Q$ by transitivity of $\longrightarrow^*$. □

3 A Cumulative Universe Hierarchy inside ZFC

In order to ensure that our interpretation is well-formed set-theoretically, we first define a basic "backbone" of our model as an ascending sequence of sets $\mathbb{S}_i$. Each $\mathbb{S}_i$ will serve as a superset of the allowed interpretations for types of sort Type_{i-1}, or of sort Prop for $i = 0$, and thus will contain sets of pairs $\langle M,\ I\rangle$ of terms and valid interpretations of those terms. As discussed below in Section 4.1, some terms will be considered "meaningless" and have a dummy interpretation $\bullet$; thus the $\mathbb{S}_i$ also include $\bullet$ for types that are meaningless. Note that $\bullet$ is also used as the interpretation of all proofs, yielding the Proof Irrelevance property which states that the interpretations of all proofs are equal.

The $\mathbb{S}_i$ are defined in Figure 4, with the help of two functions $\mathfrak{F}$ and $\mathfrak{P}$. The notation $I[F]$ in the figure indicates any set S such that $F \in^* S$, where $\in^*$ is the reflexive-transitive closure of set membership, while $I[F \cup (I_1 \mapsto I_2)]$ indicates any set obtainable from $I[F]$ by replacing 0 or more occurrences of F in the $\mathrm{TC}(S)$ (the set of all $S' \in^* S$) by the extension of F that takes I_1 to I_2. These notions can be defined in a straightforward manner using transfinite recursion, see e.g. [6]. The notation $\pi_2(S)$ indicates the set of second elements of tuples in S, while the notation $\mathfrak{F}^\omega(S)$ indicates the closure of S under 0 or more applications of $\mathfrak{F}$, which is well-defined by the Knaster-Tarski theorem as $\mathfrak{F}$ is easily seen to be monotone.

$$
\begin{array}{lcl}
\mathfrak{F}(S) & = & S \cup \{\emptyset, \bullet\} \cup \{ \langle c, \vec{I} \rangle \mid \forall i. I_i \in S \} \cup \{ I[F \cup (I_1 \mapsto I_2)] \mid I[F], I_1, I_2 \in S \} \\
\mathfrak{P}(S) & = & \mathcal{P}(\mathsf{Term} \times S) \\
\mathbb{S}_0 & = & \{\bullet\} \cup \mathfrak{P}(\{\bullet\}) \\
\mathbb{S}_{i+1} & = & \mathfrak{P}(\mathfrak{F}^\omega(\pi_2(\bigcup \mathbb{S}_i) \cup \{\mathbb{S}_i\}))
\end{array}
$$

Figure 4: The Cumulative Universe Hierarchy in ZFC

Intuitively, $\mathfrak{F}$ takes a set S of potential interpretations and adds new interpretations built from those in S. The function ensures that $\bullet$ and $\emptyset$ are in the result, where the latter is used to represent functions that are undefined on all inputs. $\mathfrak{F}$ also closes S under applications of constructors to zero or more elements of S, represented as a tuple $\langle c, \vec{I} \rangle$. Finally, $\mathfrak{F}$ extends functions already used in interpretations by adding to their domains and ranges.

To build (intpretations of) universes from sets of allowed interpretations, the $\mathfrak{P}$ function intuitively takes a set S of allowed interpretations and returns the set of all sets of pairs $\langle M, I \rangle$ for $I \in S$. We then define $\mathbb{S}_0$ as the set containing $\bullet$ (for meaningless types) as well as $\mathfrak{P}(\{\bullet\})$, the set of all sets of pairs $\langle M, \bullet \rangle$ (ensuring Proof Irrelevance for non-meaningless propositions). To construct $\mathbb{S}_{i+1}$, we first combine all interpretations already in $\mathbb{S}_i$ with the set $\mathbb{S}_i$ itself, thereby allowing each sort to be a type in the next sort. We then apply $\mathfrak{F}^\omega$ to create the constructor applications and functions over this new set, and use $\mathfrak{P}$ to form $\mathbb{S}_{i+1}$.

The following closure properties apply to $\mathfrak{F}^\omega$, using the notation $S_1 \xrightarrow{\aleph_0} S_2$ to denote the set of countable functions (i.e., whose domain is countable) from S_1 to S_2.

Lemma 5 *If $\vec{I} \in \pi_2(\bigcup \mathbb{S}_i)$, $S' \subseteq \pi_2(\bigcup \mathbb{S}_i)$, $F \in \pi_2(\bigcup \mathbb{S}_i) \xrightarrow{\aleph_0} \pi_2(\bigcup \mathbb{S}_i)$, and either $i = 0$ or $c : A : \mathsf{Prop}$ (i.e., c is impredicative), then the following are in $\pi_2(\bigcup \mathbb{S}_i)$ when they are defined:*

1. $c^\bullet(\vec{I}) = \textbf{\textit{if}}\ c : A : \mathsf{Prop}\ \textbf{\textit{then}}\ \bullet\ \textbf{\textit{else}}\ \langle c, \vec{I} \rangle$
2. $I_1 \mathbin{@^\bullet} I_2 \triangleq \textbf{\textit{if}}\ I_1 = \bullet\ \textbf{\textit{then}}\ \bullet\ \textbf{\textit{else}}\ I_1(I_2)$
3. $\lambda^\bullet(x \in S').\ F(x) \triangleq \textbf{\textit{if}}\ \forall x. F(x) = \bullet\ \textbf{\textit{then}}\ \bullet\ \textbf{\textit{else}}\ F$

Proof: All three cases are immediate when $i = 0$, as the only element of $\pi_2(\bigcup \mathbb{S}_0)$ is $\bullet$, so we show the result for $\mathbb{S}_{i+1}$ by proving that the given definitions are all in $\mathfrak{F}^\omega(S)$ for $S = \pi_2(\bigcup \mathbb{S}_i) \cup \{\mathbb{S}_i\}$. For part 1, if c is impredicative the result is also immediate. Otherwise, if there is some finite n such that $\vec{I} \in \mathfrak{F}^n(S)$ then $\langle c, \vec{I} \rangle \in \mathfrak{F}^{n+1}(S) \subseteq \mathfrak{F}^\omega(S)$. Otherwise, one or more of the I_i contain functions over $\mathfrak{F}^\omega(S)$ in their transitive closures that are "built up" by the final clause in the definition of $\mathfrak{F}$; but the same functions are in the transitive closure of $\langle c, \vec{I} \rangle$ and can be built up in the same way. For part 2, the result is immediate if $I_1 = \bullet$. Otherwise I_1 must be a function that is built up using the final clause in the definition of $\mathfrak{F}$ (in some $\mathbb{S}_j$ for $j < i$) and thus any $x \in \mathrm{Ran}(I_1)$ must satisfy $x \in \mathfrak{F}^n(S) \subseteq \mathfrak{F}^\omega(S)$. For part 3, the result is immediate if $\forall x. F(x) = \bullet$ or $F \in S$. Otherwise consider the sequence of steps of application of $\mathfrak{F}$ used to build the elements of $\mathrm{Dom}(F) \cup \mathrm{Ran}(F)$. Since these sequences of steps are all countable and since $\mathrm{Dom}(F) \cup \mathrm{Ran}(F)$ countable, it is straightforward to build up F with a countable sequence of steps of application of $\mathfrak{F}$ that interleaves the countably many countable sequences used to build $\mathrm{Dom}(F) \cup \mathrm{Ran}(F)$. □

We now consider some of the (interpretations of) types yielded by the above definitions. Note that if $T \subseteq \mathsf{Term}$ and $S \subseteq \mathfrak{F}^\omega(\pi_2(\bigcup \mathbb{S}_i) \cup \{\mathbb{S}_i\})$ then any "type" that associates terms in T with interpretations in S is in $\mathbb{S}_{i+1}$. (We can only use $S = \{\bullet\}$ for $\mathbb{S}_0$.) As an example, we can use part 3 of Lemma 5 to form, inside

$\mathbb{S}_{i+1}$, the interpretations of dependent function types such as

$$\Pi x : \mathsf{nat} \,.\, (\mathsf{fun}\ (0 \to \mathsf{nat} \mid \mathsf{succ}\ y \to (\Pi z : \mathsf{nat} \,.\, \mathsf{nat})))\ x$$

This type can be interpreted as the set of all functions whose elements map 0 to a natural number and $\mathsf{succ}\ y$ to a function on natural numbers.

To build the interpretations of inductive types, we can use the Knaster-Tarski fixed-point theorem to build up sets of interpretations that are closed under constructor applications. Let $\vec{S}$ be any sequence of sets and let $\phi(F, \vec{X})$ be any first-order formula that defines a set of pairs $S \subseteq \mathbb{S}_i$ whenever each $X_j \in S_j$ and $F\vec{S} \to \mathcal{P}(\mathbb{S}_i)$. Further, let ϕ be monotonic in F viewed as a set, i.e., $F \subseteq F'$ implies $\phi(F, \vec{X}) \subseteq \phi(F', \vec{X})$. We then have that ϕ has a unique least-fixed point $\mu F(\vec{X}).\phi(F, \vec{X})$ that defines a function from $\vec{S}$ to subsets of $\mathbb{S}_i$.

As an example, we can use $\phi(F) = \{\langle \mathsf{z},\ \langle \mathsf{z} \rangle \rangle\} \cup \{\ \langle \mathsf{s}\ M,\ (\langle \mathsf{s},\ I \rangle) \rangle \mid\ \langle M,\ I \rangle \in F\ \}$ (which defines a nullary function, i.e., a set) to give interpretations to the natural number terms. (Note that the interpretations of inductive types given below actually includes more terms M.) Part 1 of Lemma 5 shows that $\mathbb{S}_1$ is closed under $\phi(F)$, so the result is a subset of $\mathbb{S}_1$. Parameterized types, such as the lists of type A, can be defined as a function over sets which are themselves (interpretations of) types contained in $\mathbb{S}_i$, while dependent types, such as the lists of type A with length n, can be defined as functions over types and elements of types contained in $\mathbb{S}_i$. We can also accommodate inductive types a with constructors that take function arguments of type $\Pi x : A \,.\, a$, such as the type of ordinals (see, e.g., [5]), using part 3 of Lemma 5 to show that functions from a subset of $\mathbb{S}_i$ into $F(\vec{X})$ are always in $\pi_2(\bigcup \mathbb{S}_i)$.

By the Axiom of Regularity, we can define primitive recursion over such sets: the notation

$$\mathsf{primrec}_F\ (\vec{X})\ (\langle c_1,\ \vec{I} \rangle \to \phi_1(F, \vec{X}, \vec{I}) \mid \ldots \mid \langle c_n,\ \vec{I} \rangle \to \phi_n(F, \vec{X}, \vec{I}) \mid I \to \phi_0(\vec{X}, I))$$

denotes a function that takes $|\vec{X}|+1$ arguments and tests whether the last argument is of the form $\langle c_i,\ \vec{I} \rangle$. If so, the function returns ϕ_i applied to $F, \vec{X}, \vec{I}$, where F is the whole function and ϕ_i is any formula that defines a function in ZFC which only applies F to either $I_i(S_1)(S_2)\ldots(S_k)$ for some (possibly empty) sequence sets $\vec{S}$. If the test fails, the function returns $\phi_0(\vec{X}, I)$, which again is a formula that defines a function in ZFC, but this time it cannot make recursive calls to F. By the Axiom of Regularity, we know that $\in^*$ is well-founded, and since $\ldots I_i(S_1)(S_2) \in^* I_i(S_1) \in^* I_i \in^* \langle c,\ \vec{I} \rangle$ it is straightforward to see that this function performs well-founded recursion and so can be defined by transfinite recursion (see e.g. [6]). In the below, we also use $\mathsf{primrec}^\bullet_F$ in place of $\mathsf{primrec}^\bullet_F[]$, which returns $\bullet$ if the $\mathsf{primrec}^\bullet_F[]$ version returns $\bullet$ for all inputs. The latter version can be seen to be in $\mathbb{S}_i$ by part 3 of Lemma 5 whenever the domain is countable and when $\mathbb{S}_i$ is closed under the functions defined by the ϕ_i.

The Axiom of Regularity also implies that we cannot form interpretations of inductive types a that are not strictly positive, however. This is because a function cannot have any set that transitively contains itself in either its domain or range, since $\mathrm{Dom}(F), \mathrm{Ran}(F) \subseteq F$ in the usual representation of functions in ZFC. Thus it is not possible for the interpretation of a to contain $\langle c,\ F \rangle$ for any F whose domain or range contains all of a. Of course, a function in the interpretation of $\Pi x : A \,.\, a$, where a is the co-domain of F, need not contain the whole interpretation of a in its range, and so constructor c of a can take arguments of type $\Pi x : A \,.\, a$ whose interpretations do not contain themselves. A function in the interpretation of $\Pi x : a \,.\, A$, however, must include all of a in its domain in order to be total, and so such functions cannot be used in the interpretation of a. A similar argument applies to the function type $\Pi x : (\Pi y : a \,.\, A) \,.\, B$, since although a occurs positively in this type, functions (in ZF) in the interpretation of this type must contain all of $\Pi y : a \,.\, A$, elements of which must contain all of a. Viewed a different way, if $\phi(F)$ defines a set containing, for example,

functions with domain F, then $\phi(F)$ is not monotone, since such functions with domain F are not included in functions with domain F' when $F \subsetneq F'$.

4 Strong Normalization for CiC

In this section, we give a Uniform Logical Relations (ULR) interpretation $[\![\cdot]\!]$ of the terms of CiC and use this to prove SN for the well-typed terms of CiC. As discussed above, types are interpreted as sets of pairs $\langle M,\ I\rangle$, where intuitively a pair $\langle M,\ I\rangle$ is in the interpretation of a type if M satisfies the logical condition of the type and if I is a valid interpretation of M with respect to that type. SN is proved via the Reducibility Theorem, which states that $\langle M,\ [\![M]\!]\rangle$ is in $[\![A]\!]$ whenever M has type A and A itself satisfies $\langle A,\ [\![A]\!]\rangle$ is in $[\![s]\!]$ for some sort s; SN follows since all interpretations in $[\![s]\!]$ must be *reducibility candidates*, which among other things require that they contain only strongly normalizing terms.

We work in general with untyped terms, as this makes the Interpretation Weakening property much easier to formulate (see Section 4.1 below). This requires a "dummy" interpretation $\bullet$ for "meaningless" terms which have no reasonable interpretation, including, for example, untyped free variables. The object $\bullet$ is also used as the interpretation for all proofs, i.e., for all terms M of type A where A has type Prop. This has the nice side effect that $[\![\cdot]\!]$ satisfies the well-known Proof Irrelevance property. The main reason for this Proof Irrelevance is to allow $[\![\mathsf{Prop}]\!]$ to be defined before $[\![\mathsf{Type}_i]\!]$, thus allowing the universe of our interpretation to be well-founded; otherwise $[\![\mathsf{Prop}]\!]$ would have to contain functions whose domain includes $[\![\mathsf{Prop}]\!]$.

The remainder of this section is organized as follows. Section 4.1 defines the meaningless terms precisely, and shows they are closed under application. Section 4.2 then gives a ULR interpretation for CiC that maps terms into this hierarchy, and the Reducibility Theorem is proved in Section 4.3.

4.1 Meaningless Terms

As discussed above, since we are working with untyped terms we will naturally get some "meaningless" terms that are ill-formed, such as $\mathsf{Prop}\ M$. Free variables with no assigned interpretations are also considered meaningless. As in Girard's original proof, free variables are useful in proving Reducibility for λ-abstractions and pattern-matching functions, since proving SN of the body requires passing in *some* term as the argument(s), and not all types are inhabited by ground terms. In the below, we use the notation $\perp$ for a free variable; such a variable is always available, as the set of variables is infinite.

We now formalize the notion of meaningless terms:

Definition 2 (Meaningless Terms) *A* meaningless term *is a ground WHNF of the form x or of the form $M\ \vec{N}$ with $|\vec{N}|$ greater than 0 and also greater than the number of parameters of M if M is a pattern-matching function.*

The following lemma shows that the strongly normalizing meaningless terms are closed under application to strongly normalizing arguments:

Lemma 6 *If $\mathsf{SN}(M)$ and $\mathsf{SN}(N)$ for M being a meaningless term then $\mathsf{SN}(M\ N)$ and $M\ N$ is a meaningless term.*

Proof: If $M = x$, then $x\ N$ is a WHNF and thus a meaningless term. Otherwise, since M cannot be a λ-abstraction and cannot be an application $M_{\text{fun}}\ \vec{N}$ of a pattern-matching function M_{fun} with $|\vec{N}|$ equal to

the number of parameters of M_{fun}, $M\ N$ cannot match either case 1 or 2 of the definition of CBN reduction, and so, since M is a WHNF, so is $M\ N$.

To show $\mathsf{SN}(M\ N)$, we reason by induction on $\mathbf{size}(M) + \mathbf{size}(N)$ and show that all possible reductions on $M\ N$ lead to a strongly normalizing term, whereby the result follows. Since $M\ N$ cannot be a redex, we only have that $M\ N \longrightarrow M'\ N$ for $M \longrightarrow M'$ or $M\ N \longrightarrow M\ N'$ for $N \longrightarrow N'$. In both cases we have by the induction hypothesis that $M\ N$ rewrites to a strongly normalizing term, where the first case requires the straightforward fact that reduction preserves WHNFs. □

4.2 A ULR Interpretation for CiC

We turn now to the interpretation of CiC into our cumulative universe hierarchy $\mathbb{S}_i$. In the below, we use the notation $\lceil I \rceil$ to map I to the *set denoted by* I, defined as follows: if $I = \bullet$, $\lceil I \rceil$ yields the set $\{\, \langle M,\ \bullet\rangle \mid \mathsf{SN}(M) \,\}$; otherwise $\lceil I \rceil = I$. We can now define the reducibility candidates, which constitute the well-formed sets denoted by interpretations of types:

Definition 3 (Reducibility Candidate) *We say that I is a* reducibility candidate*, written* $\mathbf{CR}(I)$*, iff $S = \lceil I \rceil$ is a set of pairs $\langle M,\ I\rangle$ such that:*

(CR 1): $\mathsf{SN}(M)$*;*

(CR 2): *If $M \longrightarrow M'$ then $\langle M',\ I\rangle \in S$;*

(CR 3): *If $M' \longrightarrow_{\text{n-nc}} M$ then $\langle M',\ I\rangle \in S$; and*

(CR 4): *If $\mathsf{SN}(M')$ and M' is a meaningless term then $\langle M',\ \bullet\rangle \in S$.*

It is straightforward to see that $\bullet$ defines a valid reducibility candidate:

Lemma 7 $\mathbf{CR}(\bullet)$.

Proof: **CR 1** and **CR 4** are immediate. **CR 2** follows because $\mathsf{SN}(M)$ and $M \longrightarrow M'$ implies $\mathsf{SN}(M')$. **CR 3** follows by Lemma 3. □

We now define the interpretation $[\![\cdot]\!]^{\Delta}$ in Figure 5. This uses the *translations contexts* Δ, which map variables, recursive variables, and inductive types to their interpretations and are defined syntactically as follows:

$$\Delta ::= \cdot \mid \Delta, x \mapsto I \mid \Delta, u \mapsto I \mid \Delta, a \mapsto I$$

The interpretation $[\![M]\!]^{\Delta}$ is defined by primary induction on the shortest prefix of Σ that contains all constructors c and type constructors a used in M, and by secondary induction on the structure of M. The primary induction hypothesis is used to define the interpretations of inductive types, as all inductive types used in the definition of inductive type a and its constructors, other than a itself, are guaranteed to come before a in Σ. Otherwise $[\![\cdot]\!]^{\Delta}$ mostly works by induction on the structure of M. Note that we use the notation $\{\!|A|\!\}^{\Delta} = \lceil [\![A]\!]^{\Delta} \rceil$ for the set denoted by the interpretation of A.

The interpretation of the sorts is the set of all pairs $\langle A,\ I\rangle$ such that A is a set and I is a reducibility candidate in the appropriate $\mathbb{S}_i$. Since this does not depend on Δ we sometimes write $\{\!|s|\!\}$ for $\{\!|s|\!\}^{\Delta}$. The interpretation of function types $\Pi x\!:\!A\,.\,B$ is the set of all pairs $\langle M,\ F\rangle$ such that, for all $\langle N,\ I\rangle$ in $\{\!|A|\!\}^{\Delta}$ we have that applying M to N and applying F to I yields a pair in the interpretation of B.

Inductive type constructors a are interpreted as the least function whose range is a set of pairs $\langle M,\ I\rangle$ with $\mathsf{SN}(M)$ that satisfies the following: if M rewrites to a constructor term $c(\vec{Q};\vec{R})$ for some constructor c

$$
\begin{array}{lcl}
[\![s]\!]^{\Delta} & = & \{\, \langle A,\ I\rangle \mid \mathsf{SN}(A) \wedge \mathbf{CR}(I) \wedge I \in (\mathbf{if}\ s = \mathsf{Type}_i\ \mathbf{then}\ \mathbb{S}_{i+1}\ \mathbf{else}\ \mathbb{S}_0)\,\} \\
[\![\Pi x\!:\!A\,.\,B]\!]^{\Delta} & = & \{\, \langle M,\ (\lambda^{\bullet} p.\ F\ @^{\bullet}\ p)\rangle \mid \forall(\langle N,\ I\rangle \in \{\!|A|\!\}^{\Delta}).\ \langle M\ N,\ F\ @^{\bullet}\ I\rangle \in \{\!|B|\!\}^{\Delta, x\mapsto I}\,\} \\
[\![a(\vec{M};\vec{N})]\!]^{\Delta} & = & \mathbf{if}\ a \mapsto F \in \Delta\ \mathbf{then}\ F([\![\vec{M}]\!]^{\Delta}, [\![\vec{N}]\!]^{\Delta})\ \mathbf{else}\ [\![a]\!]([\![\vec{M}]\!]^{\Delta}, [\![\vec{N}]\!]^{\Delta}) \\
[\![x]\!]^{\Delta_1, x\mapsto I, \Delta_2} & = & I \\
[\![u]\!]^{\Delta_1, u\mapsto I, \Delta_2} & = & I \\
[\![M\ N]\!]^{\Delta} & = & [\![M]\!]^{\Delta}\ @^{\bullet}\ [\![N]\!]^{\Delta} \\
[\![\lambda\, x\!:\!A\,.\,M]\!]^{\Delta} & = & \lambda^{\bullet}(I \in \pi_2(\{\!|A|\!\}^{\Delta}).\ [\![M]\!]^{\Delta, x\mapsto I} \\
[\![c(\vec{M};\vec{N})]\!]^{\Delta} & = & c^{\bullet}([\![\vec{N}]\!]^{\Delta})
\end{array}
$$

$$
\begin{array}{l}
[\![M_{\mathrm{fun}} = \mathsf{fun}^{\mathrm{p}}\ u\ (\Gamma_{\mathsf{arg}})\ (\vec{c}\backslash\vec{\Gamma} \to \vec{M})]\!]^{\Delta} \\
\qquad = \ \mathsf{primrec}^{\bullet}_{F}\ (\vec{I} \in \pi_2(\{\!|\Gamma_{\mathsf{arg}}|\!\}^{\Delta}))\ (\langle c_1,\ \vec{I'}\rangle \to [\![M_1]\!]^{\Delta, \Gamma_{\mathsf{arg}}\mapsto\vec{I}, \Gamma_1\mapsto\vec{I'}, u\mapsto F} \mid \ldots \mid I \to \bullet)
\end{array}
$$

$$
\begin{array}{l}
[\![M_{\mathrm{fun}} = \mathsf{fun}^{\mathrm{i}}\ u\ (\Gamma_{\mathsf{arg}})\ (\vec{c}\backslash\vec{\Gamma} \to \vec{M})]\!]^{\Delta} \\
\qquad = \ \lambda^{\bullet}(\langle \vec{N},\ \vec{I}\rangle \in \{\!|\Gamma_{\mathsf{arg}}, a(\vec{Q};\vec{R})|\!\}^{\Delta}).\ \bigsqcap_{\vec{I'} \in \pi_2(\{\!|\Gamma_i, A_{\mathrm{fun}}|\!\}^{\Delta})} [\![M_i]\!]^{\Delta, (\Gamma_{\mathsf{arg}}, \Gamma_i, u)\mapsto\vec{I'}}
\end{array}
$$

$$
\begin{array}{l}
[\![a]\!] = \mu F(\vec{X},\vec{Y}).\{\, \langle M,\ I\rangle \mid \mathsf{SN}(M) \wedge \\
\qquad\qquad (\forall\vec{Q}\forall\vec{R}.\ M \longrightarrow^{*} c(\vec{Q};\vec{R})\ \mathbf{implies}\ c \in \mathrm{ctors}_{\Sigma}(a) \wedge \\
\qquad\qquad\quad \exists\vec{I}.\ \langle \vec{R},\ \vec{I}\rangle \in \{\!|\Gamma_{\mathsf{c}}|\!\}^{a\mapsto F, \Gamma_{\mathrm{p}}\mapsto\vec{X}} \wedge I = c^{\bullet}(\vec{I}) \wedge [\![\vec{M_c}]\!]^{\Gamma_{\mathrm{p}}\mapsto\vec{X}, \Gamma_{\mathrm{c}}\mapsto\vec{I}} = \vec{Y}) \wedge \\
\qquad\qquad (\neg(\exists c.\exists\vec{Q}\exists\vec{R}.\ M \longrightarrow^{*} c(\vec{Q};\vec{R}))\ \mathbf{implies}\ I = \bullet)\,\}
\end{array}
$$

Figure 5: A Uniform Logical Relations Interpretation for CiC

of a, then c must be a constructor of a, the arguments $\vec{R}$ must have some interpretations $\vec{I}$ that make them reducible, I must be $c^{\bullet}(\vec{I})$, and the interpretations of the arguments $\vec{M_c}$ in the return type of c must equal the arguments $\vec{Y}$; otherwise $I = \bullet$. We then interpret $a(\vec{M};\vec{N})$ by passing the interpretations of $\vec{M}, \vec{N}$ to the interpretation of a.

The interpretations of variables and recursive variables look those variables up in Δ. The interpretation of an application applies (using $I_1\ @^{\bullet}\ I_2$) the interpretation of the function to that of the argument. The interpretation of a λ-abstraction forms a function (using the $\lambda^{\bullet}(I \in S).\ F(x)$ notation) that takes in an interpretation for its argument and returns the interpretation of the body. The interpretation of a constructor term $c(\vec{M};\vec{N})$ applies the constructor (using the $c^{\bullet}(\vec{I})$ notation) to the interpretations of the arguments $\vec{N}$. The interpretation of a predicative pattern-matching function uses primitive recursion on the interpretation of the scrutinee, which is guaranteed to be well-defined for well-typed pattern-matching functions since the recursive calls to u in such a pattern-matching function will always be to applications of pattern variables to zero or more arguments. This translates to recursive calls on F in the interpretation that always pass interpretations which are lower in the $\in^*$ ordering than the scrutinee input. Recall that, for applications, functions (including primitive recursion), and constructor applications, the result is the dummy object $\bullet$ when the function is $\bullet$, the body of the function is constantly $\bullet$, or the constructor is impredicative, respectively.

The final case is that for impredicative pattern-matching functions. Because of Proof Irrelevance, the scrutinee for such a function will always have interpretation $\bullet$, and thus we cannot perform primitive recursion on it. Instead, the interpretation of an impredicative pattern-matching function throws away its input, and instead considers *all possible* interpretations of immediate subterms of the scrutinee and of the pattern-matching function itself. If all possible interpretations of the pattern cases of the function using all possible interpretations of the immediate subterms of the scrutinee are all equal to some result I, then the interpretation of the impredicative pattern-matching function returns I. Otherwise the interpretation is

undefined. This is written with the notation $\bigsqcap S$, where $\bigsqcap\{I\} = I$, $\bigsqcap\{\} = \bullet$, and $\bigsqcap S$ is undefined otherwise.

The reason this approach works is due to Proof Irrelevance. If the return type is a proposition, then all pattern cases are proofs and so must all have the same interpretation $\bullet$. Otherwise the scrutinee type must be an empty type, so there are no pattern cases (and $\bullet$ is returned), or the scrutinee type must be a singleton type whose constructor only takes proof arguments, and there is only one possible interpretation, again $\bullet$, for those arguments. Note that one ramification of this approach to interpreting impredicative pattern-matching functions is that $[\![M]\!]^\Delta$ is not always defined for ill-typed M.

The following structural properties can be proved by straightforward induction on the definition of $[\![\cdot]\!]^\Delta$:

Lemma 8 (Interpretation Weakening) $[\![M]\!]^{\Delta_1,\Delta_2,\Delta_3} = [\![M]\!]^{\Delta_1,\Delta_3}$ *for any* M *and* Δ_i *where both of these are defined.*

Lemma 9 (Interpretation Substitution) $[\![M]\!]^{\Delta,x\mapsto[\![N]\!]^\Delta,\Delta'} = [\![[N/x]M]\!]^{\Delta,\Delta'}$ *or both of these are undefined for any* M*,* N*,* Δ*, and* Δ'*.*

4.3 Proving Reducibility

In order to prove Reducibility, we first need to define the notion of a well-formed translation context Δ that only maps variables to valid interpretations. More specifically, we define the judgment $\Gamma \vdash (\Delta;\sigma)$ to indicate that the translation context Δ and substitution σ are well-formed with respect to Γ. For each $x : A \in \Gamma$, the judgment requires that Δ have the form $\Delta_1, x \mapsto I, \Delta_2$, and that the pair $\langle \sigma(x),\ I\rangle$ is in the set $\{\!|A|\!\}^{\Delta_1}$. For each $u :^{n;\vec{x}} \Pi\Gamma_{\mathsf{arg}}\,.\,\Pi x\!:\!A\,.\,B$ in Γ, the judgment requires that Δ have the form $\Delta_1, u \mapsto F, \Delta_2$, and that, for all sequences of pairs $\langle \vec{N},\ \vec{I}\rangle$ in $\{\!|\Gamma_{\mathsf{arg}}|\!\}^{\Delta_1}$, for all $\langle \sigma(x_j),\ I\rangle \in \{\!|A|\!\}^{\Delta_1}$ where Δ_1 maps x_j to I, and for all $\langle \vec{Q},\ \vec{I'}\rangle \in \{\!|\Gamma'|\!\}^{\Delta_1}$ for $x_j : \Pi\Gamma'\,.\,a(\vec{R};\vec{R'}) \in \Gamma$, we have that $\langle \sigma(u)\ \vec{N}\ (\sigma(x_j)\ \vec{Q}),\ F\ @^\bullet\ (\vec{I}, I_j(\vec{I'}))\rangle$ is in $\{\!|B|\!\}^\Delta$. Finally, for each $a \mapsto F$ in Δ, the judgment requires that F result from 0 or more applications of the formula used in the least fixed-point definition of $[\![a]\!]$.

We can now immediately prove Proof Irrelevance. Note that well-formedness of Δ and σ are needed, for example, when B is a free variable:

Lemma 10 (Proof Irrelevance) *Let* $\Gamma \vdash (\Delta;\sigma)$ *and* $\langle \sigma B,\ [\![B]\!]^\Delta\rangle \in \{\!|\mathsf{Prop}|\!\}$ *hold. If* $\langle M,\ I\rangle \in \{\!|B|\!\}^\Delta$ *then* $I = \bullet$*.*

The following lemmas now prove each of the cases of the Reducibility Theorem. An interesting aspect of this proof is that we do not have to prove $\mathbf{CR}(\{\!|A|\!\}^\Delta)$ for each A as a separate theorem; this result is part of Reducibility for A, i.e., it is implied by $\langle A,\ [\![A]\!]^\Delta\rangle \in \{\!|s|\!\}^\Delta$ for $A : s$. In the below, we say that term M is reducible at type A relative to $(\Delta;\sigma)$ iff $\Gamma \vdash (\Delta;\sigma)$ and the pair $\langle \sigma(M),\ [\![M]\!]^\Delta\rangle$ is in $\{\!|A|\!\}^\Delta$. We often omit A, Δ, and/or σ when these are clear from context. In addition, we say that $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ is reducible relative to $(\Delta;\sigma)$ iff $\Gamma \vdash (\Delta;\sigma)$ and, for all Δ_i and σ_i that map variables x_j for $j < i$ to terms $\langle N_j,\ I_j\rangle \in \{\!|A_j|\!\}^{\Delta_i}$ we have that $\langle (\sigma,\sigma_i)(A_i),\ [\![A_i]\!]^{\Delta,\Delta_i}\rangle \in \{\!|s_i|\!\}$ for some s_i. Again, we often omit Δ and σ when they are clear from context. Finally, we often appeal to properties **CR 1** through **CR 4** of a type A, where technically what is meant is the properties of the set $\{\!|A|\!\}^\Delta$ for the appropriate Δ.

Lemma 11 *If* $M \longrightarrow N$ *then* $[\![M]\!]^\Delta = [\![N]\!]^\Delta$ *whenever both of these are defined.*

Proof: By straightforward induction on M. □

Lemma 12 *If* $A_1 <: A_2$ *then* $\{\!|A_1|\!\}^\Delta \subseteq \{\!|A_2|\!\}^\Delta$*.*

Proof: By straightforward induction, using the fact that $\{\!|s_1|\!\}^\Delta \subseteq \{\!|s_2|\!\}^\Delta$ whenever $s_1 <: s_2$ for the base case. □

Lemma 13 *If* $\cdot \vdash s_1 : s_2$ *then* $\langle s_1,\ [\![s_1]\!]^\Delta\rangle \in \{\!|s_2|\!\}$ *for all* Δ.

Proof: $\mathsf{SN}(s_1)$ is immediate, as is $\langle s_1,\ [\![s_1]\!]^\Delta\rangle \in \mathbb{S}_i$ where $s_2 = \mathsf{Type}_i$ (s_2 cannot be Prop). All that remains is to prove $\mathbf{CR}([\![s_1]\!]^\Delta)$:

(CR 1): Immediate.

(CR 2): If $\mathsf{SN}(A)$ and $A \longrightarrow A'$, then $\mathsf{SN}(A')$; the remaining conditions are immediate.

(CR 3): For **CR 3**, if $\mathsf{SN}(A)$ and $A' \longrightarrow_{\mathrm{n-nc}} A$ then $\mathsf{SN}(A')$ holds by Lemma 3; the remaining conditions are immediate.

(CR 4): If A is a meaningless term and $\mathsf{SN}(A)$, then $\mathbf{CR}(\bullet)$ holds by Lemma 7 and $\bullet \in \mathbb{S}_0 \subseteq \mathbb{S}_i$ is immediate.

□

Lemma 14 *Let* $s_B <: s$ *and either* $s_B = \mathsf{Prop}$ *or* $s_A <: s$. *Further let* $\Gamma \vdash (\Delta;\sigma)$. *If* $\langle \sigma(A),\ [\![A]\!]^\Delta\rangle \in \{\!|s_A|\!\}^\Delta$ *and for all* $\langle N,\ I\rangle \in \{\!|A|\!\}^\Delta$ *we have* $\langle \sigma([N/x]B),\ [\![B]\!]^{\Delta,x\mapsto I}\rangle \in \{\!|s_B|\!\}$ *then* $\langle \sigma(\Pi x\!:\!A\,.\,B),\ [\![\Pi x\!:\!A\,.\,B]\!]^\Delta\rangle \in \{\!|s|\!\}$.

Proof: We have $\mathsf{SN}(\sigma(A))$ by definition of $\{\!|s_A|\!\}$. By **CR 4** for A we have $\langle \bot,\ \bullet\rangle \in \{\!|A|\!\}^\Delta$, so by assumption and by definition of $\{\!|s_B|\!\}$ we have and $\mathsf{SN}([\sigma,\bot/x]B)$, and thus we have $\mathsf{SN}(B)$. Hence we have $\mathsf{SN}(\Pi x\!:\!A\,.\,B)$.

We now show $[\![\Pi x\!:\!A\,.\,B]\!]^\Delta \in \mathbb{S}_i$ for $s = \mathsf{Type}_i$ or for $i = 0$ and $s = \mathsf{Prop}$. If $s_B = \mathsf{Prop}$ then, since by assumption B is reducible at Prop, we must have that $F\ @^\bullet\ \langle N,\ I\rangle = \bullet$ for all $\langle M,\ F\rangle \in [\![\Pi x\!:\!A\,.\,B]\!]^\Delta$. Thus $\lambda^\bullet p.\ F\ @^\bullet\ p = \bullet$ and so $F = \bullet$ by definition, and we have $[\![\Pi x\!:\!A\,.\,B]\!]^\Delta \in \mathbb{S}_0 \subseteq \mathbb{S}_i$. Otherwise by assumption we have both $s_A <: s$ and $s_B <: s$, and so by assumption and the definition of $\{\!|s_A|\!\}$ and $\{\!|s_B|\!\}$ we have $\{\!|A|\!\}^\Delta \in \mathbb{S}_j$ and $\{\!|B|\!\}^{\Delta,x\mapsto I} \in \mathbb{S}_k$ for all $\langle N,\ I\rangle \in \{\!|A|\!\}^\Delta$, where $j,k \le i$. Hence for all $\langle M,\ F\rangle \in \{\!|\Pi x\!:\!A\,.\,B|\!\}^\Delta$ we have that F is a function whose domain and range is in $\mathbb{S}_i$, and the result is immediate.

Finally, we show that $\mathbf{CR}([\![\Pi x\!:\!A\,.\,B]\!]^\Delta)$ holds:

(CR 1): By **CR 4**, we have $\langle \bot,\ \bullet\rangle \in \{\!|A|\!\}^\Delta$, so if $\langle M,\ F\rangle \in [\![\Pi x\!:\!A\,.\,B]\!]^\Delta$ then we have $\langle M\ \bot,\ F\ @^\bullet\ \bullet\rangle \in \{\!|B|\!\}^{\Delta,x\mapsto\bullet}$. By **CR 1** for B we have $\mathsf{SN}(M\ \bot)$ and hence $\mathsf{SN}(M)$.

(CR 2): If $\langle M,\ F\rangle \in \{\!|\Pi x : A\,.\,B|\!\}^\Delta$ and $M \longrightarrow M'$ then for all $\langle N,\ I\rangle \in \{\!|A|\!\}^\Delta$ we have that $\langle M\ N,\ F\ @^\bullet\ \langle N,\ I\rangle\rangle \in [\![B]\!]^{\Delta,x\mapsto I}$. By **CR 2** for B we have $\langle M'\ N,\ F\ @^\bullet\ \langle N,\ I\rangle\rangle \in [\![B]\!]^{\Delta,x\mapsto I}$, so $\langle M',\ F\rangle \in [\![\Pi x\!:\!A\,.\,B]\!]^\Delta$.

(CR 3): If $M' \longrightarrow_{\mathrm{n-nc}} M$ and $\langle M,\ F\rangle \in [\![\Pi x : A\,.\,B]\!]^\Delta$, then for all $\langle N,\ I\rangle \in \{\!|A|\!\}^\Delta$ we have that $\langle M\ N,\ F\ @^\bullet\ I\rangle \in \{\!|B|\!\}^{\Delta,x\mapsto I}$. It also follows that $M'\ N \longrightarrow_{\mathrm{n-nc}} M\ N$ by Lemma 1, and thus by **CR 3** for B we have $\langle M'\ N,\ F\ @^\bullet\ I\rangle \in \{\!|B|\!\}^{\Delta,x\mapsto I}$, and the desired result is immediate.

(CR 4): Let M' be a meaningless term such that $\mathsf{SN}(M')$. For all $\langle N,\ I\rangle \in \{\!|A|\!\}^\Delta$ we have $\mathsf{SN}(N)$ by **CR 1** for A, and so $M'\ N$ is a meaningless term with $\mathsf{SN}(M'\ N)$ by Lemma 6. Thus $\langle M'\ N,\ \bullet\rangle \in \{\!|B|\!\}^{\Delta,x\mapsto I}$ holds by **CR 4** for B. Since $\bullet = \bullet\ @^\bullet\ I$, the result follows.

□

Lemma 15 *Let* $a : \Pi\Gamma_\mathsf{p}\,.\,\Pi\Gamma_\mathsf{a}\,.\,s$, *and let the context* $\Gamma_\mathsf{p},\Gamma_\mathsf{a}$, *as well as all contexts* $\Gamma_\mathsf{p},\Gamma_\mathsf{c}$ *for constructors* c *of* a, *be reducible relative to* $\cdot$. *If* $\langle \sigma(\vec{M},\vec{N}),\ [\![\vec{M},\vec{N}]\!]^\Delta\rangle \in \{\!|\Gamma_\mathsf{p},\Gamma_\mathsf{a}|\!\}^\Delta$ *then* $\langle \sigma(a(\vec{M};\vec{N})),\ [\![a(\vec{M};\vec{N})]\!]^\Delta\rangle \in [\![s]\!]^\Delta$.

Proof: By **CR 1** for $\Gamma_{\mathsf{p}}, \Gamma_{\mathsf{a}}$ we have $\mathsf{SN}(\sigma(M_i))$ and $\mathsf{SN}(\sigma(N_j))$ for all i and j, and so $\mathsf{SN}(\sigma(a(\vec{M};\vec{N})))$. Also, if $s = \mathsf{Prop}$, then we immediately have $I = \bullet$ for all $\langle M,\ I\rangle \in \{\!|a(\vec{M};\vec{N})|\!\}^{\Delta}$, so $[\![a(\vec{M};\vec{N})]\!]^{\Delta} \in \mathbb{S}_0$. Otherwise $s = \mathsf{Type}_i$, and the well-formedness rules for inductive types (see Section 2) require that all of the types in Γ_{c} for any constructor c of a to have type s. Thus if $I = \langle c,\ \vec{I}\rangle$ we have that each $I_j \in S_j$ for some $S_j \in \mathbb{S}_i$, and it then follows that $[\![a(\vec{M};\vec{N})]\!]^{\Delta} \in \mathbb{S}_i$.

What remains is to prove $\mathbf{CR}(\{\!|a(\vec{M};\vec{N})|\!\}^{\Delta})$:

(CR 1): Immediate.

(CR 2): Straightforward, as reduction preserves SN and normal forms and because, if $M' \longrightarrow^* c(\vec{Q};\vec{R})$ for $M \longrightarrow M'$ then $M \longrightarrow^* c(\vec{Q};\vec{R})$.

(CR 3): Assume that $M' \longrightarrow_{\mathrm{n-nc}} M$ and that $\langle M,\ I\rangle \in \{\!|a(\vec{M};\vec{N})|\!\}^{\Delta}$. By Lemma 3, we have $\mathsf{SN}(M')$. By Lemma 4, we have that $M' \longrightarrow^* c(\vec{Q};\vec{R})$ implies $M \longrightarrow^* c(\vec{Q};\vec{R})$, so the condition

$$\exists \vec{I} \exists \vec{I'}.\ \langle (\vec{Q},\vec{R}),\ (\vec{I},\vec{I'})\rangle \in \{\!|\Gamma_{\mathsf{c}}|\!\}^{\Delta})$$

holds because $\langle M,\ I\rangle \in \{\!|a(\vec{M};\vec{N})|\!\}^{\Delta}$. The remaining conditions are straightforward because M' has the same normal form as M.

(CR 4): Immediate because any meaningless term M' has no normal form of the form $c(\vec{Q};\vec{R})$, and so if $\mathsf{SN}(M')$ then $\langle M',\ \bullet\rangle$ is in $\{\!|a(\vec{M};\vec{N})|\!\}^{\Delta}$.

□

Lemma 16 *If $\Gamma \vdash (\Delta;\sigma)$ and $[\![x]\!]^{\Delta}$ is defined then $\langle \sigma(x),\ [\![x]\!]^{\Delta}\rangle \in \{\!|A|\!\}^{\Delta}$.*

Proof: The assumption $\Gamma \vdash (\Delta;\sigma)$ implies $\langle \sigma(x),\ [\![x]\!]^{\Delta}\rangle \in \{\!|A|\!\}^{\Delta_1}$, and $\{\!|A|\!\}^{\Delta_1} = \{\!|A|\!\}^{\Delta}$ by Interpretation Weakening (Lemma 8). □

Lemma 17 *If $\Gamma \vdash (\Delta;\sigma)$ for $u :^{|\vec{N}|;\vec{x}} \Pi\Gamma_u \,.\, B \in \Gamma$, and if $\vec{N}, x_i$ are reducible at the types listed in Γ_u, then $\langle \sigma(u\ \vec{N}\ x_i),\ [\![u\ \vec{N}\ x_i]\!]^{\Delta}\rangle \in \{\!|[(\vec{N},x)/\Gamma_u]B|\!\}^{\Delta}$.*

Proof: The assumption that $\Gamma \vdash (\Delta;\sigma)$ and that the arguments $\vec{N}, x_i$ are reducible implies that $u\ \vec{N}\ x_i$ is reducible relative to Δ_1, as in the proof of Lemma 16. The result again follows by Interpretation Weakening (Lemma 8). □

Lemma 18 *If $\langle \sigma(M),\ [\![M]\!]^{\Delta}\rangle \in \{\!|\Pi x : A \,.\, B|\!\}^{\Delta}$ and $\langle \sigma(N),\ [\![N]\!]^{\Delta}\rangle \in \{\!|A|\!\}^{\Delta}$ then $\langle \sigma(M\ N),\ [\![M\ N]\!]^{\Delta}\rangle \in \{\!|[N/x]B|\!\}^{\Delta}$.*

Proof: By hypothesis we immediately have $\langle \sigma(M\ N),\ [\![M\ N]\!]^{\Delta}\rangle \in \{\!|B|\!\}^{\Delta, x\mapsto [\![N]\!]^{\Delta}}$, and the result then follows by Interpretation Substitution (Lemma 9). □

Lemma 19 *Let A and B be reducible at types s_A and s_B, respectively. If, for every $\langle N,\ I\rangle \in \{\!|A|\!\}^{\Delta}$, we have $\langle [\sigma,N/x](B),\ [\![B]\!]^{\Delta,x\mapsto I}\rangle \in \{\!|s|\!\}$ and also $\langle [\sigma,N/x]M,\ [\![M]\!]^{\Delta,x\mapsto I}\rangle \in \{\!|[\sigma,N/x]B|\!\}^{\Delta,x\mapsto I}$, then $\langle \sigma(\lambda\, x : A \,.\, M),\ [\![\lambda\, x : A \,.\, M]\!]^{\Delta}\rangle \in \{\!|\Pi x : A \,.\, B|\!\}^{\Delta}$.*

Proof: We must show that, for all $\langle N,\ I\rangle \in \{\!|A|\!\}^{\Delta}$, we have $\langle \sigma(\lambda\, x{:}A\,.\,M)\ N,\ [\![M]\!]^{\Delta,x\mapsto I}\rangle \in [\![B]\!]^{\Delta,x\mapsto I}$. By the definition of $\{\!|s_A|\!\}$ we have $\mathsf{SN}(A)$, and by **CR 4** for A we have $\langle \bot,\ \bullet\rangle \in \{\!|A|\!\}^{\Delta}$, so by assumption and by **CR 1** for B we have $\mathsf{SN}([\sigma,\bot/x]M)$ and thus $\mathsf{SN}(\sigma(M))$. It is straightforward to see that $\sigma(\lambda\, x{:}A\,.\,M)$ is the same as $\lambda\, x : (\sigma(A))\,.\,(\sigma(M))$ and that $[\sigma,N/x]M$ is the same as $[N/x](\sigma(M))$, and thus we have $\sigma(\lambda\, x{:}A\,.\,M)\ N \longrightarrow_{\mathrm{n-nc}} [\sigma,N/x]M$. Hence, by **CR 3** for B, we have the desired result. □

Lemma 20 *Let the type $a(\vec{M};\vec{Q})$ and the context $\Gamma_{\mathsf{p}},\Gamma_{\mathsf{c}}$ of arguments to constructor c be reducible. If $\langle \sigma(\vec{M},\vec{N}),\ [\![\vec{M},\vec{N}]\!]^{\Delta}\rangle \in \{\!|\Gamma_{\mathsf{p}},\Gamma_{\mathsf{c}}|\!\}^{\Delta}$ then $c(\vec{M};\vec{N})$ is reducible at type $a(\vec{M};\vec{Q})$.*

Proof: By **CR 1** for the types in Γ_{c}, we have that $\mathsf{SN}(\sigma(M_i))$ and $\mathsf{SN}(\sigma(N_j))$ for all i and j. Thus $\mathsf{SN}(\sigma(c(\vec{M};\vec{N})))$. For the third condition in the interpretation of a, if $\sigma(c(\vec{M};\vec{N})) \longrightarrow^{*} c(\vec{M}';\vec{N}')$ then we must have $\sigma(M_i) \longrightarrow^{*} M_i'$ and $\sigma(N_j) \longrightarrow^{*} N_j'$ for all i and j. By assumption and by **CR 2** for the A_i we then have that $\langle(\vec{M}',\vec{N}'),\ [\![\vec{M},\vec{N}]\!]^{\Delta}\rangle \in \{\!|\Gamma_{\mathsf{c}}|\!\}^{\Delta}$. Further, if a is a predicative inductive type then we cannot have $c : \mathsf{Prop}$, and we have $[\![c(\vec{M};\vec{N})]\!]^{\Delta} = \langle c,\ [\![\vec{N}]\!]^{\Delta}\rangle$. In contrast, if a is an impredicative inductive type then $c : \mathsf{Prop}$, so $[\![c(\vec{M};\vec{N})]\!]^{\Delta} = \bullet$. It is not possible for $\mathsf{NF}(\sigma(c(\vec{M};\vec{N})))$ to not start with c. □

Lemma 21 *Let $M_{\mathrm{fun}} = \mathsf{fun}^{ip}\ u\ (\Gamma_{\mathsf{arg}})\ (\vec{c}\backslash\vec{\Gamma}\to\vec{M})$ have type A_{fun}, and let the context $\Gamma_{\mathsf{arg}}, y : a(\vec{Q};\vec{R}), z : B$ be reducible with respect to Δ. If $\Gamma \vdash (\Delta;\sigma)$, and if*

$$\langle \sigma_i(M_i),\ [\![M_i]\!]^{\Delta_i}\rangle \in \{\!|B|\!\}^{\Delta_i, x\mapsto [\![c_i(\vec{Q};\Gamma_i)]\!]^{\Delta_i}}$$

holds for all Δ_i and σ_i such that $\Gamma_i' \vdash (\Delta_i;\sigma_i)$, where Γ_i' is

$$\Gamma, \Gamma_{\mathsf{arg}}, \Gamma_i, u :^{|\Gamma_{\mathsf{arg}}|;\Gamma_i} A_{\mathrm{fun}},$$

then $\langle \sigma(M_{\mathrm{fun}}),\ [\![M_{\mathrm{fun}}]\!]^{\Delta}\rangle \in \{\!|A|\!\}^{\Delta}$.

Proof: For all $\langle(\vec{N},N),\ (\vec{I},I)\rangle \in \{\!|\Gamma_{\mathsf{arg}}, x : a(\vec{Q};\vec{R})|\!\}^{\Delta}$, we must show that $\langle M,\ I_M\rangle$ is in $\{\!|B|\!\}^{\Delta,\Gamma_{\mathsf{arg}}\mapsto\vec{I},x\mapsto I}$ where $M \equiv \sigma(M_{\mathrm{fun}})\ \vec{N}\ N$ and $I_M \equiv [\![M_{\mathrm{fun}}]\!]^{\Delta}\ @^{\bullet}\ (\vec{I},I)$. By **CR 4** for the types in Γ_{arg} and Γ_i we can form $\Delta_{\bullet,i}$ and $\sigma_{\bullet,i}$ such that that extend Δ and σ, respectively, by mapping u and the variables of the contexts Γ_{arg} and Γ_i to $\bot$ in $\sigma_{\bullet,i}$ and $\bullet$ in $\Delta_{\bullet,i}$. Thus we have $\langle \sigma_{\bullet,i}(M_i),\ [\![M_i]\!]^{\Delta_{\bullet,i}}\rangle \in \{\!|B|\!\}^{\Delta_{\bullet,i},x\mapsto\bullet}$, so by **CR 1** for B we have $\mathsf{SN}(\sigma_{\bullet,i}(M_i))$ and hence $\mathsf{SN}(\sigma(M_i))$. By **CR 1** for Γ_{arg} and $a(\vec{Q};\vec{R})$ we also have $\mathsf{SN}(\vec{N})$ and $\mathsf{SN}(N)$. We now proceed by primary induction on $\mathsf{NF}(N)$ and by secondary induction on $\mathbf{size}(N)$.

For the step case of the secondary induction, if $N \longrightarrow_{\mathrm{n-nc}} N'$ then $\mathbf{size}(N') < \mathbf{size}(N)$, so by the inductive hypothesis we have that $\langle \sigma(M_{\mathrm{fun}})\ \vec{N}\ N',\ [\![M_{\mathrm{fun}}]\!]^{\Delta}\ @^{\bullet}\ \langle(\vec{N},N),\ (\vec{I},I)\rangle\rangle$ is in the required set, and the result follows by **CR 3** for B.

For the base case of the secondary induction, if N is a WHNF then it is straightforward to see that it is either of the form $c_i(\vec{Q};\vec{R}')$ or it does not reduce to a term of this form. In the latter case, the definition of $\{\!|\Delta|\!\}^{\Delta}a(\vec{Q};\vec{R})$ ensures that $I = \bullet$, and so the interpretation of M_{fun} return $I_M = \bullet$. In this case we also have that $\sigma(M_{\mathrm{fun}})\ \vec{N}\ N$ is a WHNF, and so is a meaningless term. The desired result then follows by **CR 4** for B.

In the former case, the definition of $\{\!|a(\vec{Q};\vec{R})|\!\}$ ensures that $\langle\vec{R}',\ \vec{I}'\rangle \in \{\!|\Gamma_{\mathsf{c}_i}|\!\}^{\Delta}$ for some $\vec{I}'$, since $N \longrightarrow^{*} N$. Thus we can form Δ' and σ' extend Δ and σ by mapping Γ_{arg} to $\vec{I}$ and $\vec{N}$, respectively, by mapping Γ_i to $\vec{I}'$ and $\vec{R}'$, respectively, and by mapping u to $\sigma(M_{\mathrm{fun}})$ and $[\![M_{\mathrm{fun}}]\!]^{\Delta}$, respectively. We can conclude $\Gamma_i' \vdash (\Delta';\sigma')$ for Γ_i' as above, using the primary induction hypothesis and the fact that $\mathsf{NF}(R_i')$ is a strict subterm of

$\mathsf{NF}(N)$ to handle the mapping in Δ' for u. Thus by assumption we have $\langle \sigma'(M_i),\ [\![M_i]\!]^{\Delta'} \rangle \in \{\!|B|\!\}^{\Delta', x \mapsto I_x}$ where $I_x = [\![c_i(\vec{Q}; \Gamma_i)]\!]^{\Delta'}$.

If we know that $[\![M_i]\!]^{\Delta'} = I_M$ then the result follows by **CR 3** for B, as $\sigma(M) \longrightarrow_{\mathrm{n-nc}} \sigma'(M_i)$. For predicaitve pattern-matching functions, this is immediate by the definition of $[\![M_{\mathrm{fun}}]\!]^{\Delta}$ and by Interpretation Substitution (Lemma 9) as $I = \langle c_i,\ \vec{I'} \rangle$. For impredicative pattern-matching functions with propositional return type, we know by Proof Irrelevance (Lemma 10) that $[\![M_j]\!]^{\Delta'} = \bullet$ for all j, and thus $I_M = \bullet$ as well. Finally, for impredicative pattern-matching functions with predicative return type, we must have that a has exactly one constructor that takes only proof arguments (since there are no constructors c_i for empty inductive types). Thus, by Proof Irrelevance, each I'_j has only one possible value, $I'_j = \bullet$. This means there is only one possible well-formed extension Δ' of Δ and we have that I_M is well-defined and $I_M = [\![M_j]\!]^{\Delta'}$. □

Theorem 1 (Reducibility) *Let* $|\Delta| \vdash M : A$ *and* $|\Delta| \vdash A : s$ *for some* Δ *and* σ *with* $\Gamma \vdash (\Delta; \sigma)$. *If* $\langle \sigma(A),\ [\![A]\!]^{\Delta} \rangle \in \{\!|s|\!\}$ *then* $\langle \sigma(M),\ [\![M]\!]^{\Delta} \rangle \in \{\!|A|\!\}^{\Delta}$.

Proof: By induction on the derivation of $|\Delta| \vdash M : A$, using Lemmas 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21. □

Corollary 1 *(Strong Normalization) If* $\Gamma \vdash M : A$ *then* $\mathsf{SN}(M)$.

5 Related Work

There have been numerous proofs of SN for various forms of both the Calculus of Inductive Constructions (CiC) and the Calculus of Constructions without inductive types (CC). As discussed above, none has been extended to full CiC. Many of these proofs are based on the *Candidates de Reducibilité* of Girard [10], or on a modified version called *saturated sets*. Under these approaches, each type is interpreted as a set of terms that satisfies some closure properties (such as properties **CR 1** through **CR 4** above), which include strong normalization of the elements. The proof then proves the Reducibility Theorem, which states that each term is in the interpretation of its type, and SN follows.

Using this approach, Luo [12] proves SN for the Extended Calculus of Constructions (ECC), a version of CC with an impredicative universe (Prop), a hierarchy of infinitely many predicative universes (Type_i), and product types. This seems to be the only SN proof for a calculus with infinitely many universes, probably due to the non-uniformity problem discussed in the Introduction. Luo's proof defines an interpretation for each type construct, and then proves Reducibility. The interpretation is not defined for all terms, however, only types that are in weak head normal form. Thus Luo requires a complex and involved Quasi-Normalization theorem, which states that all terms can be reduced to a weak head normal form.

Coquand and Gallier [4] adapt the saturated sets approach to work on typed sets of terms, yielding a proof of SN for CC with one universe. They call their interpretation a Kripke structure, as each type is interpreted with respect to a particular world (similar to the Δ in our approach). The interpretation of a type contains pairs $\langle \Delta',\ M \rangle$ of interpretation contexts Δ' that extend Δ and terms M such that M is well-typed with respect to (the context associated with) Δ'. Again, because of non-uniformity it is unclear how to extend this approach to multiple universes. It is also interesting to consider how sets of typed terms would make our approach more difficult. Specifically, Interpretation Weakening (Lemma 8) would be much more complex even to state; for types A, we would have $[\![A]\!]^{\Delta'} \subseteq [\![A]\!]^{\Delta}$ when Δ' extends Δ, since there are fewer Δ'' that extend the former than the latter. It is unclear how to specify this relationship on arbitrary terms in a uniform way.

Geuvers [8] extends the saturated sets approach by defining two interpretations, one for terms and one for types. The interpretation for terms maps a term to a substitution instance of that term, and Reducibility then requires only that the instance be in the interpretation of a related substitution instance of the type of the term. Geuvers then shows that this approach can be used to support CC with strong products and inductive kinds, but comments that it will probably not work with small inductive types like the natural numbers.

Altenkirch [2] adapts the notion of saturated sets to be a semantic notion, using the saturated sets as the basis for a denotational idea called a *CC-structure* and thus building a model of CC. Soundness of this model then implies SN of CC. Altenkirch then shows that this argument can be extended to handle inductive types. In a similar manner, Goguen [11] proves SN by providing a typed operational semantics of terminating computations of a given type. He then shows that this semantics is sound for the syntactic type theory, i.e., that all well-typed terms have a corresponding terminating computation. This result is proved for CiC with with two universes, one predicative and one impredicative.

Finally, Geuvers and Nederhof [9] prove SN for CC without universes by mapping terms in this language into System F_ω. The proof for SN of System F_ω uses saturated sets but is much simpler, because types are trivially strongly normalizing. It is not clear, however, that this approach could be adapted easily (without proving SN already) to extensions of CC.

6 Conclusion

In this work, we have shown how to adapt the Logical Relations proof technique to prove strong normalization (SN) for the full Calculus of Inductive Constructions (CiC) with a cumulative predicative type hierarchy. The guiding principle that has allowed us to handle full CiC is *uniformity*, meaning that our interpretation does not depend on whether a term is a type. One benefit of this is that our interpretation does not depend on confluence. The key technical development that enables uniformity is to interpret types not as sets of terms but as sets of pairs $\langle M,\ I\rangle$ of terms M and valid interpretations I of M. Closure conditions on the sets associated with types, for example that the sets be saturated sets, are handled simply by including only such sets as valid interpretations of types in the interpretations $[\![\mathsf{Prop}]\!]$ and $[\![\mathsf{Type}_i]\!]$ of universes. Surprisingly, the proof is actually very similar to the original proof given by Girard, with the proof of the closure conditions on interpretations of types being folded into the proof of Reducibility, leading to a simple and elegant proof. In fact, the proof given by Girard can be seen as a special case of Uniform Logical Relations.

As a final consideration, the interpretation we give can easily be extended to support a transfinite hierarchy of universes. That is, we could support universes Type_α for any ordinal α. Of course, to define such a theory syntactically requires a notation of well-founded ordinals, such as the well-known notation based on ϕ-functions for ordinals less than Γ_0 (see e.g. [7]). Thus not only is Zermelo-Frankel set theory (ZF) proof-theoretically stronger than CiC, but, if we view CiC as an ordinal operation that assigns to each α the proof-theoretic ordinal of CiC with up to α universes, then the ordinal of ZF is also larger then this function applied to α (as well as infinitely many fixed points of this function).[3] See e.g. Coquand et al. [5] for more on proof-theoretic ordinals of type theory.

It has been shown by Aczel [1], however, that *constructive* ZF (CZF), without the axiom of the excluded middle, can be encoded in CiC, and so the proof-theoretic strength of CZF is no more than that of CiC. Also, if we add excluded middle for each sort s (EM_s) to CiC, then the work of Werner [19] shows that ZF

[3] ZFC is known to have the same proof-theoretic strength as ZF.

can be encoded, where excluded middle for sort s has the following type:[4]

$$\Pi A\!:\!s\,.\,A +_s (A \to \mathsf{False})$$

Here False is the empty inductive type and $+_i$ denotes the inductive type for disjunction in sort s. In fact, Werner shows that ZF with i inaccessible cardinals (ZF_i) can be encoded in CiC with $EM_{\mathsf{Type}_{i+2}}$, where an *inaccessible cardinal* in ZF is essentially a set that is so big that it includes a whole model of ZF (and thus ZF_{i+1} is guaranteed to be proof-theoretically stronger than ZF_i). Note that, although it is possible to interpret EM_{Prop} as $\bullet$ in our construction (because of Proof Irrelevance), it is not possible to interpret EM_{Type_i} since, for some types A, neither A nor its negation is inhabited. This is good, because, by the above and by Gödel's Second Incompleteness Theorem, defining an interpretation that did allow this in ZF is only possible if ZF is inconsistent. In this light, we can see that (predicative) excluded middle is in fact a very powerful axiom.

Acknowledgments

The author would like to acknowledge Chris Casinghino and Peter Hancock for helpful discussions, as well as the anonymous reviewers who caught a number of bugs and omissions in a previous version of this paper.

References

[1] Peter Aczel. The type theoretic interpretation of constructive set theory. In *Logic Colloqium '77*, 1977.

[2] Thorsten Altenkirch. *Constructions, Inductive Types and Strong Normalization*. PhD thesis, University of Edinburgh, 1993.

[3] Bruno Barras. Sets in coq, coq in sets. *Journal of Formalized Reasoning*, 3(1):29–48, 2010.

[4] Thierry Coquand and Jean Gallier. A proof of strong normalization for the theory of constructions using a kripke-like interpretation. Technical Report MS-CIS-90-44, University of Pennsylvania, 1990.

[5] Thierry Coquand, Peter Hancock, and Anton Setzer. Ordinals in type theory, 1997. invited talk at Computer Science Logic (CSL '97).

[6] Herbert B. Enderton. *Elements of Set Theory*. Academic Press, 1977.

[7] Jean Gallier. What's so special about Kruskal's theorem and the ordinal Γ_0? A survey of some results in proof theory. *Annals of Pure and Applied Logic*, 53(3):199–260, 1991.

[8] Herman Geuvers. A short and flexible proof of strong normalization for the calculus of constructions. In *International Workshop on Types for Proofs and Programs*, pages 14–38, 1995.

[9] Herman Geuvers and Mark-Jan Nederhof. Modular proof of strong normalization for the calculus of constructions. *Journal of Functional Programming*, 1(2):155–189, 1991.

[10] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[4] Werner actually uses EM_{Prop} along with the Type-Theoretical Description Axioms at each sort s ($TTDA_s$), but the author has verified in the Coq proof assistant that EM_s implies $TTDA_s$.

[11] Healfdene Goguen. *A Typed Operational Semantics for Type Theory*. PhD thesis, University of Edinburgh, 1994.

[12] Zhaohui Luo. *An Extended Calculus of Constructions*. PhD thesis, University of Edinburgh, 1990.

[13] Per Martin-Löf. An intuitionistic theory of types. In G Sambin and J Smith, editors, *Twenty-Five Years of Constructive Type Theory*. Oxford University Press, 1998.

[14] Alexandre Miquel. A model for impredicative type systems, universes, intersection types and subtyping. In *15th Annual IEEE Symposium on Logic in Computer Science (LICS)*, 2000.

[15] Alexandre Miquel and Benjamin Werner. The not so simple proof-irrelevant model of cc. In *The 2002 international conference on Types for proofs and programs (TYPES)*, pages 240–258, 2002.

[16] B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory*. Oxford University Press, 1990.

[17] M. Stefanova and H. Geuvers. A simple model construction for the calculus of constructions. In *The 1996 International Conference on Types for Proofs and Programs (TYPES)*, pages 5–8, 1996.

[18] The Coq Development Team. *The Coq Proof Assistant Reference Manual, Version V8.3*, 2010. http://coq.inria.fr/doc/.

[19] Benjamin Werner. Sets in types, types in sets. In *Third International Symposium on Theoretical Aspects of Computer Software (TACS)*, pages 530–346, 1997.

[20] Benjamin Werner. On the strength of proof-irrelevant type theories. *Logical Methods in Computer Science*, 4, 2008.

[21] Edwin Westbrook. *Higher-Order Encodings with Constructors*. PhD thesis, Washington University in Saint Louis, 2008.

[22] Edwin Westbrook. Uniform logical relations. Technical Report TR11-01, Rice University, 2011.