

Consistency of the Predicative Calculus of Cumulative Inductive Constructions (pCuIC)

Amin Timany, Matthieu Sozeau

▶ **To cite this version:**

Amin Timany, Matthieu Sozeau. Consistency of the Predicative Calculus of Cumulative Inductive Constructions (pCuIC). [Research Report] RR-9105, KU Leuven, Belgium; Inria Paris. 2017, pp.30. hal-01615123v2

HAL Id: hal-01615123

https://inria.hal.science/hal-01615123v2

Submitted on 23 Jan 2018 (v2), last revised 13 May 2020 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



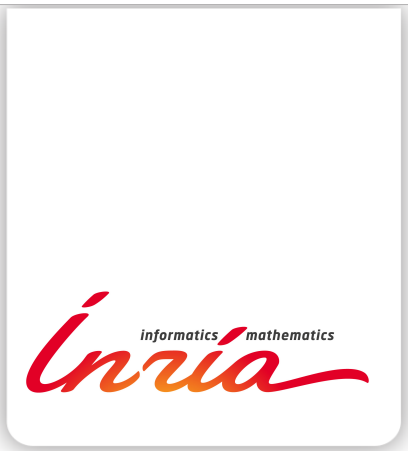
Consistency of the Predicative Calculus of Cumulative Inductive Constructions (pCuIC)

Amin Timany
imec-Distrinet, KU Leuven, Belgium
amin.timany@cs.kuleuven.be

Matthieu Sozeau
Inria Pi.R2 & IRIF, Université Paris 7 Denis Diderot
matthieu.sozeau@inria.fr

RESEARCH REPORT
N° 9105
October 2017
Project-Team Pi.R2

ISSN 0249-6399 ISRN INRIA/RR--9105--FR+ENG



Consistency of the Predicative Calculus of Cumulative Inductive Constructions (pCuIC)

Amin Timany*
imec-Distrinet, KU Leuven, Belgium
amin.timany@cs.kuleuven.be

Matthieu Sozeau
Inria Pi.R2 & IRIF, Université Paris 7 Denis Diderot
matthieu.sozeau@inria.fr

Project-Team Pi.R2

Research Report n° 9105 — version 2 — initial version October 2017 —
revised version January 2018 — 31 pages

Abstract: In order to avoid well-know paradoxes associated with self-referential definitions, higher-order dependent type theories stratify the theory using a countably infinite hierarchy of universes (also known as sorts), $\mathtt{Type}_0 : \mathtt{Type}_1 : \cdots$. Such type systems are called cumulative if for any type A we have that $A : \mathtt{Type}_i$ implies $A : \mathtt{Type}_{i+1}$. The predicative calculus of inductive constructions (pCIC) which forms the basis of the Coq proof assistant, is one such system.
In this paper we present and establish the soundness of the predicative calculus of cumulative inductive constructions (pCuIC) which extends the cumulativity relation to inductive types.

Key-words: logic, metatheory, Coq, calculus of constructions, set theory

* This research was partly carried out while I was visiting Inria Paris and Université Paris Diderot and partly while I was visiting Aarhus university.

RESEARCH CENTRE
PARIS

2 rue Simone Iff - CS 42112
75589 Paris Cedex 12

Cohérence du Calcul Prédicatif des Constructions Inductives Cumulatives

Résumé : Les théories des types d'ordre supérieur sont stratifiées afin d'éviter les paradoxes bien connus associés aux définitions circulaires. Elles utilisent une hiérarchie dénombrable d'univers (aussi appelé sortes), $\texttt{Type}_0 : \texttt{Type}_1 : \cdots$. Ces systèmes de types sont appelés cumulatifs si pour tout type A on a $A : \texttt{Type}_i$ implique $A : \texttt{Type}_{i+1}$. Le calcul prédicatif des constructions inductives (pCIC), qui forme la base de l'assistant de preuve Coq, est un tel système.

Dans cet article, nous présentons une extension du calcul, dont nous prouvons la cohérence relative vis à vis de la théorie des ensembles. Ce nouveau calcul étend la relation de cumulativité aux types inductifs.

Mots-clés : logique, métathéorie, Coq, calcul des constructions, théorie des ensembles

1 Introduction

We construct a model for pCuIC based inspired by the model of Lee and Werner [2011] establishing consistency of pCuIC. In loc. cit. authors present a model of pCIC without inductive types in the sort `Prop`. Similarly, the model that we shall construct also does not feature inductive types in the sort `Prop`. We note, however, that most inductive types in `Prop` can be encoded using their Church encoding. For instance, the type `False` and conjunction of two predicates can be defined as follows:

```
Definition conj (P Q : Prop) :=
  ∀ (R : Prop), (P → Q → R) → R.

Definition False := ∀ (P : Prop), P.
```

The model of Lee and Werner [2011] does not support the sort `Set`, neither the predicative version nor the impredicative one. In this paper, we treat the predicative sort `Set` simply as a shorthand for $\texttt{Type}_0$. Note that in Coq *it does not hold that* `Prop` : `Set`. However, in our model the interpretation of `Prop` is an element of the interpretation of $\texttt{Type}_0$ but this does not preclude our model from being sound for the typing rules of the system. In addition, the model of Lee and Werner [2011] does support η-expansion yet this is not included in their rules for judgemental equality. We do include η-expansion in our system.

The main difference between our system and the one by Lee and Werner [2011], apart from cumulativity for inductive types which is a new contribution to pCIC, is that in Lee and Werner [2011] the system features case analysis and fixpoints (recursive functions on inductive types) for inductive types while ours features eliminators. It is well-known that all functions written using fixpoints and case analysis on inductive types are representable using eliminators. However, the former is closer to actual implementation in Coq. On the other hand, the occurrences of recursive calls in recursive functions are may be hidden under some computations. That is, in order to obtain the position and arguments of recursive calls, which are crucial for constructing the model of recursive functions, in general requires normalizing the body of the recursive function. It so appears that the intention of Lee and Werner [2011] is to construct a set theoretic model for Coq, assuming the normalization property which already implies soundness of their system. This, however, is not possible when constructing a model that is intended to establish the soundness of the system. Indeed, our model, similarly to the model of Lee and Werner [2011], does not imply normalization of well-typed terms. Therefore, when modeling recursive functions, i.e., fixpoints in Coq, we cannot assume that we can normalize the body of the fixpoint. This is the main reason why we opted for formalizing our system with eliminators instead.

1.1 Contributions

Timany and Jacobs [2015] give an account of then work-in-progress on extending pCIC with a single cumulativity rule for cumulativity of inductive types. The authors show a rather restricted subsystem of the system that they present to be sound. This subsystem roughly corresponds to the fragment where terms of cumulative inductive types do not appear as dependent arguments in other terms. The proof given in Timany and Jacobs [2015] is done by giving a syntactic translation from that subsystem to pCIC. In this paper, we extend and complete the work that was initiated by Timany and Jacobs [2015].

In particular, in this work, we consider a more general version of the cumulativity rule for inductive types. Adding to this, we also consider related rules for judgemental equality of inductive types which are given rise to by the mutual cumulativity relation and also judgemental equality of the terms constructors of types in the cumulativity relation.

In this work, we present the system pCuIC in its full details and prove that the system as presented to be sound. We do this by constructing a set-theoretic model in ZFC, together with the axiom that there are countably many uncountable strongly inaccessible cardinals, inspired by the model of Lee and Werner [2011]. The cumulativity of inductive types as presented in this paper is now supported in Coq 8.7 [The Coq Development Team, 2017].

Notations for different equalities and the like:

$a \triangleq b$	a is defined as b
$a = b$	equality of mathematical objects, e.g., sets, sequences, etc.
$t \equiv t'$	Syntactically identical terms
$\Gamma \vdash t \simeq t' : A$	Judgemental equality, i.e., t and t' are judgementally equal terms of type A under context Γ

2 pCuIC

The terms and contexts of the language pCuIC is as follows:

$$
\begin{aligned}
& x, y, z, \ldots && \text{(Variables)}\\
& s ::= \mathtt{Prop}, \mathtt{Type}_0, \mathtt{Type}_1, \ldots && \text{(Sorts)}\\
t, u, \ldots, M, N, \ldots, & \\
A, B, \cdots ::= {} & x \mid s \mid \mathbf{\Pi} x : A.\, B \mid \boldsymbol{\lambda} x : A.\, t \mid \\
& \mathbf{let}\, x := t : A\, \mathbf{in}\, u \mid M\ N \mid && \text{(Terms)}\\
& \mathcal{D}.x \mid \mathbf{Elim}(t; \mathcal{D}.d_i; Q_{d_1}, \ldots, Q_{d_n})\, \{f_{c_1}, \ldots, f_{c_m}\} & \\
\mathcal{D} ::= {} & \mathbf{Ind}_n\, \{\Delta_I := \Delta_C\} && \text{(Inductive blocks)}\\
\Delta ::= {} & \cdot \mid \Delta, x : A && \text{(Declarations)}\\
\Gamma ::= {} & \cdot \mid \Gamma, x : A, \mid \Gamma, x := t : A \mid \Gamma, \mathcal{D} && \text{(Contexts)}
\end{aligned}
$$

Note that although by abuse of notation we write $x : A \in \Gamma$, $x := t : A \in \Gamma$ or $x : A \in \Delta$, contexts and declarations *are not* sets. In particular, the order in declarations is important, this is even the case for declarations where there can be no dependence among the elements. We write $\Delta(x)$ to refer to A whenever $x : A \in \Delta$.

Here, $\mathbf{Ind}_n\, \{\Delta_I := \Delta_C\}$ is a block of mutual inductive definitions where n is the number of parameters, the declarations in Δ_I are the inductive types of the block and declarations in Δ_C are constructors of the block. The term $\mathbf{Ind}_n\, \{\Delta_I := \Delta_C\}\, .x$ is an inductive definition whenever $x \in \Delta_I$ and a constructor whenever $x \in \Delta_C$. The term $\mathbf{Elim}(t; \mathcal{D}.d_k; Q_{d_1}, \ldots, Q_{d_n})\, \{f_{c_1}, \ldots, f_{c_m}\}$ is the elimination of t (a term of the inductive type $\mathcal{D}.d_k$), Q_{d_i}'s are the motives of elimination, i.e., the result of the elimination will have type $Q_{d_k}\ \vec{a}\ t$ whenever the term being eliminated, t, has type $\mathcal{D}.d_k\ \vec{a}$. The term f_{c_i} in the eliminator above is a *case-eliminator* corresponding to the case where t is constructed using constructor $\mathcal{D}.c_i$.

We write $\mathsf{len}(\vec{s})$, $\mathsf{len}(p)$, $\mathsf{dom}(f)$, $\mathsf{dom}(\Delta)$ and $\mathsf{dom}(\Gamma)$ respectively for the length of a sequence $\vec{s}$, length of a tuple p, the domain of a partial map f, domain of a declaration Δ or domain of a context Γ. Notice crucially that the inductive types and constructors of an inductive block are *not* part of the domain of the context that they appear in. We write nil for the empty sequence. We write $\mathit{Inds}(\Gamma)$ for the sequence of inductive types in the context Γ. We write tuples as $\langle a_1; a_2; \ldots; a_n \rangle$.

Definition 2.1 (Free variables)**.**

Free variables of terms

$$
\begin{aligned}
FV(\texttt{Prop}) &\triangleq \emptyset \\
FV(\texttt{Type}_i) &\triangleq \emptyset \\
FV(z) &\triangleq \{z\} \\
FV(\boldsymbol{\lambda} y : A.\, u) &\triangleq FV(A) \cup (FV(u) \setminus \{y\}) \\
FV(\textbf{let}\, y := u : A\, \textbf{in}\, v) &\triangleq FV(A) \cup FV(u) \cup (FV(v) \setminus \{y\}) \\
FV(u\ v) &\triangleq FV(u) \cup FV(v) \\
FV(\mathcal{D}.z) &\triangleq FV(\mathcal{D}) \\
FV(\textbf{Elim}(u; \mathcal{D}.d_i; \vec{Q}) \left\{ \vec{f} \right\}) &\triangleq FV(u) \cup FV(\mathcal{D}) \cup FV(\vec{Q}) \cup FV(\vec{f})
\end{aligned}
$$

Free variables of inductive blocks

$$
FV(\textbf{Ind}_n \{\Delta_I := \Delta_C\}) \triangleq FV(\Delta_I) \cup FV(\Delta_C)
$$

Free variables of sequences of terms

$$
\begin{aligned}
FV(\mathsf{nil}) &\triangleq \mathsf{nil} \\
FV(v, \vec{v}) &\triangleq FV(v) \cup FV(\vec{v})
\end{aligned}
$$

Free variables of declarations

$$
\begin{aligned}
FV(\cdot) &\triangleq \emptyset \\
FV(y : A, \Delta) &\triangleq FV(A) \cup FV(\Delta)
\end{aligned}
$$

We define simultaneous substitution for terms as follows:

Definition 2.2 (Simultaneous substitution)**.** *We assume that $\vec{x}$ is a sequence of distinct variables. In this definition for the sake of simplicity we use y for all bound variables.*

Substitution for terms

$$
\begin{aligned}
\texttt{Prop}\, [\vec{x}/\vec{t}] &\triangleq \texttt{Prop} \\
\texttt{Type}_i\, [\vec{x}/\vec{t}] &\triangleq \texttt{Type}_i \\
z\, [\vec{x}/\vec{t}] &\triangleq t_i \ \textit{if}\ x_i = z \\
z\, [\vec{x}/\vec{t}] &\triangleq z \ \textit{if}\ \forall i.\ x_i \neq z \\
(\boldsymbol{\lambda} y : A.\, u)\, [\vec{x}/\vec{t}] &\triangleq \boldsymbol{\lambda} y : A\, [\vec{x}/\vec{t}]\, .\, u\, [\vec{x}'/\vec{t}'] \\
(\textbf{let}\, y := u : A\, \textbf{in}\, v)\, [\vec{x}/\vec{t}] &\triangleq \textbf{let}\, y := u\, [\vec{x}/\vec{t}] : A\, [\vec{x}/\vec{t}]\ \textbf{in}\, v\, [\vec{x}'/\vec{t}'] \\
(u\ v)\, [\vec{x}/\vec{t}] &\triangleq u\, [\vec{x}/\vec{t}]\ \ v\, [\vec{x}/\vec{t}] \\
\textbf{Ind}_n \{\Delta_I := \Delta_C\}\, .z\, [\vec{x}/\vec{t}] &\triangleq \textbf{Ind}_n \{\Delta_I\, [\vec{x}/\vec{t}] := \Delta_C\, [\vec{x}/\vec{t}]\}\, .z \\
\textbf{Elim}(u; \mathcal{D}.d_i; \vec{Q}) \left\{ \vec{f} \right\}\, [\vec{x}/\vec{t}] &\triangleq \textbf{Elim}(u\, [\vec{x}/\vec{t}]\, ; .\mathcal{D}.d_i\, [\vec{x}/\vec{t}]\, ; \vec{Q}\, [\vec{x}/\vec{t}]) \left\{ \vec{f}\ [\vec{x}/\vec{t}] \right\}
\end{aligned}
$$

Substitution for inductive blocks

$$\mathsf{Ind}_n \{\Delta_I := \Delta_C\} \left[\vec{x}/\vec{t}\right] \triangleq \mathsf{Ind}_n \left\{\Delta_I \left[\vec{x}/\vec{t}\right] := \Delta_C \left[\vec{x}/\vec{t}\right]\right\}$$

Substitution for sequences

$$\begin{aligned} \mathsf{nil} \left[\vec{x}/\vec{t}\right] &\triangleq \mathsf{nil} \\ v, \vec{v} \left[\vec{x}/\vec{t}\right] &\triangleq v \left[\vec{x}/\vec{t}\right], \vec{v} \left[\vec{x}/\vec{t}\right] \end{aligned}$$

Substitution for declarations

$$\begin{aligned} \cdot \left[\vec{x}/\vec{t}\right] &\triangleq \cdot \\ y : A, \Delta \left[\vec{x}/\vec{t}\right] &\triangleq y : A \left[\vec{x}/\vec{t}\right], \Delta \left[\vec{x}/\vec{t}\right] \end{aligned}$$

Substitution for context

$$\begin{aligned} \cdot \left[\vec{x}/\vec{t}\right] &\triangleq \cdot \\ y : A, \Delta \left[\vec{x}/\vec{t}\right] &\triangleq y : A \left[\vec{x}/\vec{t}\right], \Delta \left[\vec{x'}/\vec{t'}\right] \end{aligned}$$

where $\vec{x'} = \vec{x}$ *and* $\vec{t'} = \vec{t}$ *if* y *does not appear in* $\vec{x}$ *and is as follows whenever* $x_i = y$*:*

$$\vec{x'}; \vec{t'} = x_0, \ldots, x_{i-1}, \ldots, x_n; t_0, \ldots, t_{i-1}, \ldots, t_n$$

2.1 Basic constructions

Figure 1 shows typing rules for the basic constructions, i.e., well-formedness of contexts ($\mathcal{WF}(\Gamma)$), sorts, let bindings, dependent products (also referred to as dependent functions), lambda abstractions and applications. It also contains the rules for the judgemental equality for these constructions. In this figure, the relation $\mathcal{R}_s$ indicates the sort that a (dependent) product type belongs to. The sort `Prop` is *impredicative* and therefore any product type with codomain in `Prop` also belongs to `Prop`.

2.2 Inductive types and their eliminators

The typing rules for inductive types, their constructors and eliminators are depicted in Figure 2.

Well-formedness of inductive types The first rule in this figure is the well-formedness of inductive types. It states that in order to have that the context Γ is well-formed after adding the mutual inductive block $\mathsf{Ind}_n \{\Delta_I := \Delta_C\}$, i.e., $\mathcal{WF}(\Gamma, \mathsf{Ind}_n \{\Delta_I := \Delta_C\})$, all inductive types in the block as well as all constructors need to be well-typed terms. In addition, we must have that the well-formedness side condition $\mathcal{I}_n(\Gamma, \Delta_I, \Delta_C)$ holds. The well-formedness side condition $\mathcal{I}_n(\Gamma, \Delta_I, \Delta_C)$ holds whenever:

- All variables in Δ_I and Δ_C are distinct.

$$\textsc{WF-ctx-empty}\ \mathcal{WF}(\cdot) \qquad \textsc{WF-ctx-hyp}\ \frac{\Gamma \vdash A : s \qquad x \notin \mathsf{dom}(\Gamma)}{\mathcal{WF}(\Gamma, x : A)} \qquad \textsc{WF-ctx-def}\ \frac{\Gamma \vdash t : A \qquad x \notin \mathsf{dom}(\Gamma)}{\mathcal{WF}(\Gamma, (x := t : A))}$$

$$\textsc{Prop}\ \frac{\mathcal{WF}(\Gamma)}{\Gamma \vdash \mathtt{Prop} : \mathtt{Type}_i} \qquad \textsc{Hierarchy}\ \frac{\mathcal{WF}(\Gamma) \qquad i < j}{\Gamma \vdash \mathtt{Type}_i : \mathtt{Type}_j} \qquad \textsc{Var}\ \frac{\mathcal{WF}(\Gamma) \qquad x : A \in \Gamma \quad \text{or} \quad (x := t : A) \in \Gamma}{\Gamma \vdash x : A}$$

$$\textsc{Let}\ \frac{\Gamma, (x := t : A) \vdash u : B}{\Gamma \vdash \mathbf{let}\, x := t : A\, \mathbf{in}\, u : B\,[t/x]}$$

$$\textsc{Let-eq}\ \frac{\Gamma \vdash A \simeq A' : s \qquad \Gamma \vdash t \simeq t' : A \qquad \Gamma, (x := t : A) \vdash u \simeq u' : B}{\Gamma \vdash \mathbf{let}\, x := t : A\, \mathbf{in}\, u \simeq \mathbf{let}\, x := t' : A'\, \mathbf{in}\, u' : B\,[t/x]}$$

$$\textsc{Prod}\ \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad \mathcal{R}_s(s_1, s_2, s_3)}{\Gamma \vdash \mathbf{\Pi} x : A.\, B : s_3}$$

$$\textsc{Prod-eq}\ \frac{\Gamma \vdash A \simeq A' : s_1 \qquad \Gamma, x : A \vdash B \simeq B' : s_2 \qquad \mathcal{R}_s(s_1, s_2, s_3)}{\Gamma \vdash \mathbf{\Pi} x : A.\, B \simeq \mathbf{\Pi} x : A'.\, B' : s_3}$$

$$\textsc{Lam}\ \frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \mathbf{\Pi} x : A.\, B : s}{\Gamma \vdash \boldsymbol{\lambda} x : A.\, M : \mathbf{\Pi} x : A.\, B}$$

$$\textsc{Lam-eq}\ \frac{\Gamma \vdash A \simeq A' : s_1 \qquad \Gamma, x : A \vdash M \simeq M' : B \qquad \Gamma \vdash \mathbf{\Pi} x : A.\, B : s_2}{\Gamma \vdash \boldsymbol{\lambda} x : A.\, M \simeq \boldsymbol{\lambda} x : A'.\, M' : \mathbf{\Pi} x : A.\, B}$$

$$\textsc{App}\ \frac{\Gamma \vdash M : \mathbf{\Pi} x : A.\, B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B\,[M/x]} \qquad \textsc{App-eq}\ \frac{\Gamma \vdash M \simeq M' : \mathbf{\Pi} x : A.\, B \qquad \Gamma \vdash N \simeq N' : A}{\Gamma \vdash M\ N \simeq M'\ N' : B\,[M/x]}$$

$$\textsc{Predicativity}\ \mathcal{R}_s(\mathtt{Type}_i, \mathtt{Type}_j, \mathtt{Type}_{\mathsf{max}\{i,j\}}) \qquad \textsc{Predicativity'}\ \mathcal{R}_s(\mathtt{Prop}, \mathtt{Type}_i, \mathtt{Type}_i) \qquad \textsc{Impredicativity}\ \mathcal{R}_s(s, \mathtt{Prop}, \mathtt{Prop})$$

Figure 1: Basic construction

$$\textsc{Ind-WF}\quad \frac{\mathcal{I}_n(\Gamma, \Delta_I, \Delta_C) \qquad \Gamma \vdash A : s_d \text{ for all } (d : A) \in \Delta_I \qquad \Gamma, \Delta_I \vdash T : s_d \text{ for all } (c : A) \in \Delta_C \text{ if } c \in \mathsf{Constrs}(\Delta_C, d)}{\mathcal{WF}(\Gamma, \mathbf{Ind}_n \{\Delta_I := \Delta_C\})}$$

$$\textsc{Ind-type}\quad \frac{\mathcal{WF}(\Gamma) \qquad \mathcal{D} \equiv \mathbf{Ind}_n \{\Delta_I := \Delta_C\} \in \Gamma \qquad d_i \in \mathsf{dom}(\Delta_I)}{\Gamma \vdash \mathcal{D}.d_i : \Delta_I(d_i)}$$

$$\textsc{Ind-constr}\quad \frac{\mathcal{WF}(\Gamma) \qquad \mathcal{D} \equiv \mathbf{Ind}_n \{\Delta_I := \Delta_C\} \in \Gamma \qquad c \in \mathsf{dom}(\Delta_C)}{\Gamma \vdash \mathcal{D}.c : \Delta_C(c)\left[\vec{d}/\overrightarrow{\Delta_I.d}\right]}$$

$$\textsc{Ind-Elim}\quad \frac{\begin{array}{c}\mathcal{WF}(\Gamma) \qquad \mathcal{D} \equiv \mathbf{Ind}_n \{\Delta_I := \Delta_C\} \in \Gamma \\ \mathsf{dom}(\Delta_I) = \{d_1, \ldots, d_l\} \qquad \mathsf{dom}(\Delta_C) = \{c_1, \ldots, c_{l'}\} \\ \Gamma \vdash Q_{d_i} : \mathbf{\Pi} \vec{x} : \vec{A}.\, (d_i\ \vec{x}) \to s' \text{ where } \Delta_I(d_i) \equiv \mathbf{\Pi} \vec{x} : \vec{A}.\, s \text{ for all } 1 \le i \le l \\ \Gamma \vdash t : \mathcal{D}.d_k\ \vec{m} \qquad \Gamma \vdash f_{c_i} : \xi_{\mathcal{D}}^{\vec{Q}}(c_i, \Delta_C(c_i)) \text{ for all } 1 \le i \le l'\end{array}}{\Gamma \vdash \mathbf{Elim}(t; \mathcal{D}.d_k; Q_{d_1}, \ldots, Q_{d_l})\left\{f_{c_1}, \ldots, f_{c_{l'}}\right\} : Q_{d_k}\ \vec{m}\ t}$$

$$\textsc{Ind-Elim-eq}\quad \frac{\begin{array}{c}\mathcal{WF}(\Gamma) \qquad \mathcal{D} \equiv \mathbf{Ind}_n \{\Delta_I := \Delta_C\} \in \Gamma \\ \mathsf{dom}(\Delta_I) = \{d_1, \ldots, d_l\} \qquad \mathsf{dom}(\Delta_C) = \{c_1, \ldots, c_{l'}\} \\ \Gamma \vdash Q_{d_i} \simeq Q'_{d_i} : \mathbf{\Pi} \vec{x} : \vec{A}.\, (d_i\ \vec{x}) \to s' \text{ where } \Delta_I(d_i) \equiv \mathbf{\Pi} \vec{x} : \vec{A}.\, s \text{ for all } 1 \le i \le l \\ \Gamma \vdash t \simeq t' : \mathcal{D}.d_k\ \vec{m} \qquad \Gamma \vdash f_{c_i} \simeq f'_{c_i} : \xi_{\mathcal{D}}^{\vec{Q}}(c_i, \Delta_C(c_i)) \text{ for all } 1 \le i \le l'\end{array}}{\begin{array}{c}\Gamma \vdash \mathbf{Elim}(t; \mathcal{D}.d_k; Q_{d_1}, \ldots, Q_{d_l})\left\{f_{c_1}, \ldots, f_{c_{l'}}\right\} \simeq \\ \mathbf{Elim}(t'; \mathcal{D}.d_k; Q'_{d_1}, \ldots, Q'_{d_l})\left\{f'_{c_1}, \ldots, f'_{c_{l'}}\right\} : Q_{d_k}\ \vec{m}\ t\end{array}}$$

Figure 2: Inductive types and eliminators

- The first n arguments of all inductive types and constructors in the block are the parameters. In other words, there is a sequence of terms $\vec{P}$ such that $\mathsf{len}(\vec{P}) = n$ and for all $x : T \in \Delta_I, \Delta_C$ we have $T \equiv \mathbf{\Pi} \vec{p} : \vec{P}. U$ for some U.

- Parameters are parametric. In other words, for all $c : T \in \Delta_C$ we have $T \equiv \mathbf{\Pi} \vec{p} : \vec{P}. \mathbf{\Pi} \vec{x} : \vec{B}. d\ \vec{p}\ \vec{v}$. That is, every constructor constructs a term of some inductive type in the block where values applied for parameter arguments of the inductive type are parameter arguments of the constructor.

- Every inductive type is just a type (an element of a universe) that depends on a number of arguments beginning with parameters. The non-parameter arguments are called the arity of the inductive type. In other words, for all $d \in \mathsf{dom}(\Delta_I)$ we have $\Delta_I(d) \equiv \mathbf{\Pi} \vec{x} : \vec{P}. \mathbf{\Pi} \vec{y} : \vec{A}. s$ where $\vec{A}$ is called the arity and $s \neq \texttt{Prop}$ is the sort of the inductive type.

- Every constructor is a constructs terms of an inductive type in the block. In other words, for all $c \in \mathsf{dom}(\Delta_C)$ we have $c \in \mathsf{Constrs}(\Delta_C, d)$ for some $d \in \Delta_I$ where $\mathsf{Constrs}(\Delta_C, d)$ is the set of constructors in Δ_C that construct terms of the inductive type d.

$$\mathsf{Constrs}(\Delta_C, d) \triangleq \left\{ c \in \mathsf{dom}(\Delta_C) \mid \Delta_C(c) \equiv \mathbf{\Pi} \vec{p} : \vec{P}. \mathbf{\Pi} \vec{x} : \vec{U}. d\ \vec{u} \right\}$$

- All inductive types in the block appear only strictly positively in constructors. In other words, for all $c \in \mathsf{dom}(\Delta_C)$, $\mathsf{Pos}_{\mathsf{dom}(\Delta_I)}(\Delta_C(c))$ where $\mathsf{Pos}_S(t)$ is determined by the following rules:

$$\frac{u \text{ does not appear in } \vec{A} \text{ or } \vec{t} \text{ for all } u \in S \qquad B \in S}{\mathsf{Pos}^{\dagger}_S(\mathbf{\Pi} \vec{x} : \vec{A}. B\ \vec{t})}$$

$$\frac{u \text{ does not appear in } \vec{t} \text{ for all } u \in S \qquad B \in S}{\mathsf{Pos}_S(B\ \vec{t})} \qquad \frac{\mathsf{Pos}^{\dagger}_S(A) \qquad \mathsf{Pos}_S(B)}{\mathsf{Pos}_S(A \to B)}$$

$$\frac{u \text{ does not appear in } \vec{A} \text{ for all } u \in S \qquad \mathsf{Pos}_S(B)}{\mathsf{Pos}_S(\mathbf{\Pi} \vec{x} : \vec{A}. B)}$$

Inductive types and constructs Rules Ind-Type and Ind-constr indicate, respectively, the type of inductive types and constructs in a block of mutually inductive types. The type of an inductive type of a block is exactly the same as declared in the block. The type of constructors of a block is determined by the type declared in the block except that inductive types in block are replaced by their proper (global) names.

Example 2.3. *The following is the definition natural numbers inpCuIC.*

$$\begin{aligned} \mathcal{N} &\triangleq \mathbf{Ind}_0\{nat : \texttt{Type}_0 := zero : nat, succ : nat \to nat\} \\ nat &\triangleq \mathcal{N}.nat \\ zero &\triangleq \mathcal{N}.zero \\ succ &\triangleq \mathcal{N}.succ \end{aligned}$$

Example 2.4. *The following is the definition of universe polymorphic lists (for level i) in pCuIC.*

$$\begin{aligned}
\mathcal{L}_i &\triangleq \mathbf{Ind}_1\{list : \mathbf{\Pi} A : \mathtt{Type}_i.\, \mathtt{Type}_i := nil : \mathbf{\Pi} A : \mathtt{Type}_i.\, list\ A,\\
&\qquad cons : \mathbf{\Pi} A : \mathtt{Type}_i.\, A \to list\ A \to list\ A\}\\
list_i &\triangleq \mathcal{L}_i.list\\
nil_i &\triangleq \mathcal{L}_i.nil\\
cons_i &\triangleq \mathcal{L}_i.cons
\end{aligned}$$

Example 2.5. *Definition of a finitely branching tree as a mutual inductive block in pCuIC.*

$$\begin{aligned}
&\mathbf{Ind}_0\{FTree : \mathtt{Type}_0, Forest : \mathtt{Type}_0 :=\\
&\qquad leaf : FTree, node : Forset \to FTree,\\
&\qquad Fnil : Forest, Fcons : FTree \to Forest \to Forest\}
\end{aligned}$$

Eliminators The term $\mathbf{Elim}(t; \mathcal{D}.d_k; Q_{d_1}, \ldots, Q_{d_l})\left\{f_{c_1}, \ldots, f_{c_{l'}}\right\}$ is the elimination of the term t (which is of type d_k (in some inductive block $\mathcal{D}$) where the result of elimination of inductive types in the block, i.e., motives of eliminations, is given by $\vec{Q}$ and $\vec{f}$ are functions for elimination of terms constructed using particular constructors. The term, above, has type $Q_{d_k}\ \vec{m}\ t$ if t has type $d_k\ \vec{m}$.

Each case-eliminator f_{c_i} is the recipe for generating a term of the appropriate type (according to the corresponding motive) out of arguments of the constructor c_i under the assumption that all (mutually) recursive arguments are already appropriately eliminated. This is perhaps best seen in the rule Iota below which describes the judgemental equality corresponding to the (intended) operational behavior of eliminators.

The function $\xi^{\vec{Q}}_{\mathcal{D}}(c_i, \Delta_C(c_i))$ ascribes a type to the case-eliminator f_{c_i} in the manner explained above. That is, $\xi^{\vec{Q}}_{\mathcal{D}}(c_i, \Delta_C(c_i))$ is a function type that given arguments of the constructor c_i (and their eliminated version if they are (mutually) recursive arguments) produces a term of the appropriate type according to the motives. It is formally defined as follows by recursion on derivation of $\mathsf{Pos}_{\mathsf{dom}(\Delta_I)}(\Delta_C(c_i))$:

If $P \equiv \mathbf{\Pi}\vec{x} : \vec{A}.\, d_i\ \vec{m}$ and we have $\mathsf{Pos}^{\dagger}_{\mathsf{dom}\Delta_I}(P)$ and $\mathsf{Pos}_{\mathsf{dom}\Delta_I}(B)$

$$\xi^{\vec{Q}}_{\mathcal{D}}(t, P \to B) \triangleq \mathbf{\Pi} p : P.\, (\mathbf{\Pi}\vec{x} : \vec{A}.\, Q_{d_i}\ m\ (p\ \vec{x})) \to \xi^{\vec{Q}}_{\mathcal{D}}(t\ p, B)$$

Otherwise,

$$\xi^{\vec{Q}}_{\mathcal{D}}(t, \mathbf{\Pi} x : A.\, B) \triangleq \mathbf{\Pi} x : A.\, \xi^{\vec{Q}}_{\mathcal{D}}(t\ x, B)$$

Otherwise, if $d \in \mathsf{dom}(\Delta_I)$

$$\xi^{\vec{Q}}_{\mathcal{D}}(t, d\ \vec{m}) \triangleq Q_d\ \vec{m}\ t$$

Example 2.6. *The following is the definition of the recursive function corresponding to the principle of mathematical induction on natural numbers in pCuIC.*

$$\begin{aligned}
nat_ind \triangleq \boldsymbol{\lambda} P : nat \to \mathtt{Prop}.\, \boldsymbol{\lambda} o : P\ zero.\, \boldsymbol{\lambda} stp : \mathbf{\Pi} x : nat.\ P\ x \to P\ (succ\ x).\\
\boldsymbol{\lambda} n : nat.\, \mathbf{Elim}(n; nat; P)\,\{o, stp\}
\end{aligned}$$

The term nat_ind above has the type

$$\mathbf{\Pi} P : nat \to \texttt{Prop}.\ (P\ Z) \to (\mathbf{\Pi} x : nat.\ P\ x \to P\ (S\ x)) \to \mathbf{\Pi} n : nat.\ P\ n$$

Example 2.7. *The following is the definition the recursive function to add two natural numbers in pCuIC.*

$$add \triangleq \boldsymbol{\lambda} n : nat.\, \boldsymbol{\lambda} m : nat.\, \mathbf{Elim}(n; nat; \boldsymbol{\lambda}_ : nat.\ nat)\ \{m, \boldsymbol{\lambda} x : nat.\ succ\ x\}$$

Example 2.8. *The following is the definition the polymorphic and universe polymorphic recursive function to compute the length of a list in pCuIC.*

$$\begin{aligned} length \triangleq\ & \boldsymbol{\lambda} A : \texttt{Type}_i.\, \boldsymbol{\lambda} l : list_i\ A.\, \mathbf{Elim}(l; list; \boldsymbol{\lambda}_ : list_i.\ nat) \\ & \{\boldsymbol{\lambda} B : \texttt{Type}_i.\ zero, \boldsymbol{\lambda} B : \texttt{Type}_i.\, \boldsymbol{\lambda} x : B.\, \boldsymbol{\lambda} y : list_i\ B.\, \boldsymbol{\lambda} z : nat.\ succ\ z\} \end{aligned}$$

Remark 2.9. *From the definition of eliminators, it might appear that they limit expressivity of our system. It might so appear that they only allow us to define recursive functions that are paramtrically polymorphic in the parameters of the inductive block. This, however is not the case, as existence of inductive types allows for breaking this paramtricity. In order to demonstrate this, the next example defines a function that sums up a list of natural numbers, where list is the polymorphic type of lists defined above.*

It is necessary to formulate eliminators such that motives and case-eliminators include parameters. This is due to the way inductive types are defined. In particular, a constructor is only required to produce a term of the inductive type with the parameters the same as the whole block. When an inductive type (including the one that the constructor in question is producing) appears as an argument of the constructor it needs not be with the same parameters as that of the inductive block. This is also the case in the practical implementation of inductive types in Coq. An example of such an inductive type is the following definition of lists where one can instead of an element of the list store some code *in the list representing that element as a* list *of natural numbers.*

$$\begin{aligned} \mathbf{Ind}_1\{ & Clist : \mathbf{\Pi} A : \texttt{Type}_i.\ \texttt{Type}_i := Cnil : \mathbf{\Pi} A : \texttt{Type}_i.\ Clist\ A, \\ & Ccons : \mathbf{\Pi} A : \texttt{Type}_i.\ A \to Clist\ A \to Clist\ A \\ & Ccons' : \mathbf{\Pi} A : \texttt{Type}_i.\ Clist\ nat \to Clist\ A \to Clist\ A\} \end{aligned}$$

Example 2.10. *The following is the definition a function that sums up a list of natural numbers in pCuIC. Notice that this is a non-paramtrically-polymorphic recursive function defined in our system for a polymorphic definition of lists.*

We first define an inductive type that can determine when a type is the type of natural numbers.

$$\begin{aligned} \mathcal{N} &\triangleq \mathbf{Ind}_0\{isNat : \mathbf{\Pi} A : \texttt{Type}_0.\ \texttt{Type}_0 := IN : isNat\ nat\} \\ isNat &\triangleq \mathcal{N}.isNat \\ IN &\triangleq \mathcal{N}.IN \end{aligned}$$

The elimination of this type allows us to write a function that given a type A and an element of the type $isNat\ A$ we can construct a function from A to the natural numbers.

$$\begin{aligned} toNat \triangleq\ & \boldsymbol{\lambda} A : \texttt{Type}_0.\, \boldsymbol{\lambda} i : isNar\ A.\, \mathbf{Elim}(i; isNat; \boldsymbol{\lambda} B : \texttt{Type}_0.\, \boldsymbol{\lambda}_ : isNat\ B.\, B \to nat) \\ & \{\boldsymbol{\lambda} x : nat.\, x\} \end{aligned}$$

$$\textsc{Eq-ref}\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t \simeq t : A} \qquad \textsc{Eq-sym}\ \frac{\Gamma \vdash t \simeq t' : A}{\Gamma \vdash t' \simeq t : A} \qquad \textsc{Eq-trans}\ \frac{\Gamma \vdash t \simeq t' : A \qquad \Gamma \vdash t \simeq t'' : A}{\Gamma \vdash t \simeq t'' : A}$$

$$\textsc{Beta}\ \frac{\Gamma, x : A \vdash M : B \qquad \Gamma, x : A \vdash B : s \qquad \Gamma \vdash N : A}{\Gamma \vdash (\boldsymbol{\lambda} x : A.\, M)\ N \simeq M\,[N/x] : B\,[N/x]} \qquad \textsc{Delta}\ \frac{\mathcal{WF}(\Gamma) \qquad x := t : A \in \Gamma}{\Gamma \vdash x \simeq t : A}$$

$$\textsc{Eta}\ \frac{\Gamma \vdash t : \boldsymbol{\Pi} x : A.\, B}{\Gamma \vdash t \simeq \boldsymbol{\lambda} x : A.\, t\ x : \boldsymbol{\Pi} x : A.\, B} \qquad \textsc{Zeta}\ \frac{\Gamma \vdash t : A \qquad \Gamma, x := t : A \vdash u : B}{\Gamma \vdash (\mathbf{let}\, x := t : A\, \mathbf{in}\, u) \simeq u\,[t/x] : B\,[t/x]}$$

$$\textsc{Iota}\ \frac{\begin{array}{c}\mathcal{WF}(\Gamma) \qquad \mathcal{D} \equiv \mathbf{Ind}_n\,\{\Delta_I := \Delta_C\} \in \Gamma \qquad \Gamma \vdash \mathcal{D}' \preceq^{\dagger} \mathcal{D} \\ \mathsf{dom}(\Delta_I) = \{d_1, \ldots, d_l\} \qquad \mathsf{dom}(\Delta_C) = \{c_1, \ldots, c_{l'}\} \\ \Gamma \vdash Q_{d_i} : \boldsymbol{\Pi} \vec{x} : \vec{A}.\, (d_i\ \vec{x}) \to s' \text{ where } \Delta_I(d_i) \equiv \boldsymbol{\Pi} \vec{x} : \vec{A}.\, s \text{ for all } 1 \le i \le l \\ \Gamma \vdash \mathcal{D}'.c_j\ \vec{a} : \mathcal{D}.d_k\ \vec{m} \qquad 1 \le j \le l' \qquad \Gamma \vdash f_{c_i} : \xi_{\mathcal{D}}^{\vec{Q}}(\mathcal{D}.c_i, \Delta_C(c_i)) \text{ for all } 1 \le i \le l'\end{array}}{\begin{array}{c}\Gamma \vdash \mathbf{Elim}(\mathcal{D}'.c_j\ \vec{a}; \mathcal{D}.d_k; Q_{d_1}, \ldots, Q_{d_l})\,\{f_{c_1}, \ldots, f_{c_{l'}}\} \simeq \\ \mu_{\mathcal{D}}^{\vec{Q};\vec{f}}(f_{c_j}; \vec{a}; \Delta_C(c_j)) : Q_{d_k}\ \vec{m}\ (\mathcal{D}'.c_j\ \vec{a})\end{array}}$$

Figure 3: The main judgemental equality rules

Now we can use these constructions to define the basis for our sum function.

$$\begin{aligned} sum_el' \triangleq\ & \boldsymbol{\lambda} A : \mathtt{Type}_0.\, \boldsymbol{\lambda} A : list_0\ A.\, \mathbf{Elim}(l; list_0; \boldsymbol{\lambda} B : \mathtt{Type}_0.\, isNar\ A \to nat) \\ & \{\boldsymbol{\lambda} B : \mathtt{Type}_0.\, \boldsymbol{\lambda} _ : isNar\ A.\, zero, \\ & \boldsymbol{\lambda} B : \mathtt{Type}_0.\, \boldsymbol{\lambda} x : B.\, \boldsymbol{\lambda} y : list\ B.\, \boldsymbol{\lambda} z : isNat\ B \to nat.\, \boldsymbol{\lambda} i : isNar\ A. \\ & add\ (toNat\ ix)\ (z\ i)\} \end{aligned}$$

Using this helper function we can define the non-polymorphic *function that given a list of natural numbers computes the sum of all its elements.*

$$sum_el \triangleq \boldsymbol{\lambda} l : list_0\ nat.\, sum_el'\ nat\ l\ IN$$

2.3 Judgemental equality

The main typing rules for judgemental equality, except for those related to cumulativity, are presented in Figure 3.

The rules Eq-ref, Eq-sym and Eq-trans make the judgemental equality an equivalence relation. The rule Beta corresponds to the operational rule for β-reduction in lambda calculus. The rules Delta and Zeta correspond to the operation of expansion of global definitions (defined in the context) and let-bindings, respectively. The rule Eta corresponds to η-expansion for (dependent) functions.

The rule Iota corresponds to the operation of reduction of recursive functions and case analysis (in systems featuring these, e.g., Coq itself) and that of eliminators as in our case. Notice that this rule only applies if the term on which elimination is being performed is a constructor

applied to some terms[1]. This rule specifies that, as expected, the eliminator, applied to a term that is constructed out of a constructor (of the corresponding type or any of its sub-types, see Remark 2.11 below) applied to some terms, is equivalent to the corresponding case-eliminator applied arguments of the constructor, after (mutually) recursive arguments are appropriately eliminated. This is exactly what the recursor $\mu_{\mathcal{D}}^{\vec{Q};\vec{f}}$ does, applying arguments of the constructor to the corresponding case-eliminator after eliminating (mutually) recursive arguments as necessary. Recursors are defined as follows by recursion on the derivation of $\mathsf{Pos}_{\mathsf{dom}(\Delta_I)}(\Delta_C(c_i))$:

If $P \equiv \mathbf{\Pi} \vec{x} : \vec{A}.\, d_i\ \vec{m}$ and we have $\mathsf{Pos}^{\dagger}_{\mathsf{dom}\Delta_I}(P)$ and $\mathsf{Pos}_{\mathsf{dom}\Delta_I}(B)$

$$\mu_{\mathcal{D}}^{\vec{Q};\vec{f}}(t; b, \vec{a}; P \to B) \triangleq \mu_{\mathcal{D}}^{\vec{Q};\vec{f}}\left(t\ b\ \left(\boldsymbol{\lambda} \vec{x} : \vec{A}.\, \mathbf{Elim}(b\ \vec{x}; \mathcal{D}.d_i; \vec{Q})\left\{\vec{f}\right\}\right); \vec{a}; B\right)$$

Otherwise,

$$\mu_{\mathcal{D}}^{\vec{Q};\vec{f}}(t; b, \vec{a}; \mathbf{\Pi} x : A.\, B) \triangleq \mu_{\mathcal{D}}^{\vec{Q};\vec{f}}(t\ b; \vec{a}; B)$$

Otherwise, if $d \in \mathsf{dom}(\Delta_I)$

$$\mu_{\mathcal{D}}^{\vec{Q};\vec{f}}(t; \mathsf{nil}; d\ \vec{m}) \triangleq t$$

Remark 2.11 (IOTA and subtyping of cumulativity for inductive tpyes)**.** *We, in addition, stipulate here that the eliminators can eliminate terms constructed using the corresponding constructor of any inductive type that is a sub-type of the inductive type for which the elimination is specified. Note that in the Coq implementation of cumulativity for inductive types, cumulativity is only considered for different instances of the same inductive type at different universe levels. That is, only for two instances of the same universe polymorphic inductive type. In those settings, the inductive types being sup-types of one another are instance of the same inductive types and the eliminators, and in general the operational semantics, simply ignore the universes in terms. Here, we have to consider this side condition to achieve a similar result.*

2.4 Cumulativity

Figures 4 and 5 show the rules for cumulativity in pCuIC. Rules in Figure 5 pertain to cumulativity of inductive types. These are novel additions to the predicative calculus of constructions (pCIC). The main purpose of the model constructed in this paper is to prove these rules correct.

The rule IND-LEQ specifies when we say a blocks of inductive types $\mathcal{D}$ is included in another block $\mathcal{D}'$. Intuitively, this holds when corresponding arguments of the inductive type (only the arity and not the parameters) of corresponding inductive types have the appropriate subtyping relation and also for all corresponding arguments of corresponding constructors, except for those that are parameters of the inductive types. Excluding parameters has interesting consequences. For instance, let us consider the following example that shows the polymorphic definition of lists of elements of a type A : $\mathtt{Type}_i$ are independent of the level of the type, i. Notice that cumulativity, implies that A also has type $\mathtt{Type}_j$ for any $i \leq j$.

Example 2.12. *Consider the universe polymorphic definition of lists from Example 2.3. This definitions allows us to use rules* INT-LEQ *and* C-IND *to conclude that* $\mathcal{L}_i, \mathcal{L}_j, \Gamma \vdash List_i\ A \preceq List_j\ A$

[1]This restriction is indeed necessary in Coq so as to guarantee strong normalization.

$$\textsc{Prop-in-Type}\quad \Gamma \vdash \texttt{Prop} \preceq \texttt{Type}_i$$

$$\textsc{Cum-Type}\quad \frac{i \leq j}{\Gamma \vdash \texttt{Type}_i \preceq \texttt{Type}_j}$$

$$\textsc{Cum-trans}\quad \frac{\Gamma \vdash M \preceq N \qquad \Gamma \vdash N \preceq O}{\Gamma \vdash M \preceq O}$$

$$\textsc{Cum-weaken}\quad \frac{\Gamma \vdash M \preceq N \qquad \Gamma \vdash O : s \qquad x \notin \mathsf{dom}(\Gamma)}{\Gamma, x : O \vdash M \preceq N}$$

$$\textsc{Cum-Prod}\quad \frac{\Gamma \vdash A_1 \simeq B_1 : s \qquad \Gamma, x : A_1 \vdash A_2 \preceq B_2}{\Gamma \vdash \mathbf{\Pi} x : A_1 . A_2 \preceq \mathbf{\Pi} x : B_1 . B_2}$$

$$\textsc{Cum-eq-L}\quad \frac{\Gamma \vdash M \simeq N : s \qquad \Gamma \vdash N \preceq O}{\Gamma \vdash M \preceq O}$$

$$\textsc{Cum-eq-R}\quad \frac{\Gamma \vdash M \preceq N \qquad \Gamma \vdash N \simeq O : s}{\Gamma \vdash M \preceq O}$$

$$\textsc{Cum}\quad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \preceq B}{\Gamma \vdash t : B}$$

$$\textsc{Cum-eq}\quad \frac{\Gamma \vdash t \simeq t' : A \qquad \Gamma \vdash A \preceq B}{\Gamma \vdash t \simeq t' : B}$$

$$\textsc{Eq-Cum}\quad \frac{\Gamma \vdash M \simeq M' : s}{\Gamma \vdash M \preceq M'}$$

Figure 4: Cumulativity

$$\textsc{Ind-leq}\quad \frac{\begin{array}{c}\mathcal{D} \equiv \mathbf{Ind}_n \{\Delta_I := \Delta_C\} \in \Gamma \qquad \mathcal{D}' \equiv \mathbf{Ind}_n \{\Delta'_I := \Delta'_C\} \in \Gamma \\ \mathsf{dom}(\Delta_I) = \mathsf{dom}(\Delta'_I) \qquad \mathsf{dom}(\Delta_C) = \mathsf{dom}(\Delta'_C) \\ \Big[\Delta_I(d) \equiv \vec{p} : \vec{P}.\mathbf{\Pi}\vec{z} : \vec{V}.\, s \qquad \Delta'_I(d) \equiv \vec{p} : \vec{P'}.\mathbf{\Pi}\vec{z} : \vec{V'}.\, s' \qquad \Gamma, \vec{p} : \vec{P} \vdash \vec{V} \preceq \vec{V'} \\ \Big(\Delta_C(c) \equiv \mathbf{\Pi}\vec{p} : \vec{P}.\mathbf{\Pi}\vec{x} : \vec{U}.\, d\ \vec{u} \qquad \Delta'_C(c) \equiv \mathbf{\Pi}\vec{p} : \vec{P'}.\mathbf{\Pi}\vec{x} : \vec{U'}.\, d\ \vec{u'} \\ \Gamma, \vec{p} : \vec{P} \vdash \vec{U} \preceq \vec{U'} \qquad \Gamma, \vec{p} : \vec{P}, \vec{x} : \vec{U} \vdash \vec{u} \simeq \vec{u'} : \vec{P'}, \vec{V'} \\ \text{for } c \in \mathsf{Constrs}(\Delta_C, d)\Big) \qquad \text{for } d \in \mathsf{dom}(\Delta_I)\Big]\end{array}}{\Gamma \vdash \mathcal{D} \preceq^{\dagger} \mathcal{D}'}$$

$$\textsc{C-Ind}\quad \frac{\begin{array}{c}\mathcal{D} \equiv \mathbf{Ind}_n \{\Delta_I := \Delta_C\} \qquad \mathcal{D}' \equiv \mathbf{Ind}_n \{\Delta'_I := \Delta'_C\} \qquad \Gamma \vdash \mathcal{D} \preceq^{\dagger} \mathcal{D}' \\ \Gamma \vdash \mathcal{D}.d\ \vec{a} : s \qquad \Gamma \vdash \mathcal{D}'.d\ \vec{a} : s'\end{array}}{\Gamma \vdash \mathcal{D}.d\ \vec{a} \preceq \mathcal{D}'.d\ \vec{a}}$$

$$\textsc{Ind-Eq}\quad \frac{\Gamma \vdash \mathcal{D}.d\ \vec{a} \preceq \mathcal{D}'.d\ \vec{a} \qquad \Gamma \vdash \mathcal{D}'.d\ \vec{a} \preceq \mathcal{D}.d\ \vec{a} \qquad \Gamma \vdash \mathcal{D}.d\ \vec{a} : s \qquad \Gamma \vdash \mathcal{D}'.d\ \vec{a} : s}{\Gamma \vdash \mathcal{D}.d\ \vec{a} \simeq \mathcal{D}'.d\ \vec{a} : s}$$

$$\textsc{Constr-Eq-L}\quad \frac{\Gamma \vdash \mathcal{D}'.d\ \vec{a} \preceq \mathcal{D}.d\ \vec{a} \qquad \Gamma \vdash \mathcal{D}.c\ \vec{m} : \mathcal{D}.d\ \vec{a} \qquad \Gamma \vdash \mathcal{D}'.c\ \vec{m} : \mathcal{D}'.d\ \vec{a}}{\Gamma \vdash \mathcal{D}.c\ \vec{m} \simeq \mathcal{D}'.c\ \vec{m} : \mathcal{D}.d\ \vec{a}}$$

$$\textsc{Constr-Eq-R}\quad \frac{\Gamma \vdash \mathcal{D}.d\ \vec{a} \preceq \mathcal{D}'.d\ \vec{a} \qquad \Gamma \vdash \mathcal{D}.c\ \vec{m} : \mathcal{D}.d\ \vec{a} \qquad \Gamma \vdash \mathcal{D}'.c\ \vec{m} : \mathcal{D}'.d\ \vec{a}}{\Gamma \vdash \mathcal{D}.c\ \vec{m} \simeq \mathcal{D}'.c\ \vec{m} : \mathcal{D}'.d\ \vec{a}}$$

Figure 5: Cumulativity and judgemental equality for inductive types

for all types A regardless of i and j which would not be the case if we were to compare parameters. This result in turn allows us to use the rule Ind-Eq *to conclude that* $\mathcal{L}_i, \mathcal{L}_j, \Gamma \vdash List_i\ A \simeq List_j\ A : \mathtt{Type}_i$ *and use the rule* Constr-Eq-L *to conclude* $\mathcal{L}_i, \mathcal{L}_j, \Gamma \vdash nil_i\ A \simeq nil_j\ A : List_i\ A$ *both regardless of i and j. Similarly, we can conclude that* $\mathcal{L}_i, \mathcal{L}_j, \Gamma \vdash cons_i\ A\ a\ l \simeq nil_j\ A\ a'\ l' : List_i\ A$ *whenever* $\mathcal{L}_i, \mathcal{L}_j, \Gamma \vdash l \simeq l' : List_i\ A\ \mathcal{L}_i, \mathcal{L}_j, \Gamma \vdash a \simeq\ a' : A$.

Notice that rules C-Ind, Ind-Eq, Constr-Eq-L and Constr-Eq-R are only applicable if the inductive type or the constructor in the latter case are fully applied. This is mainly because subtyping only makes sense for types (elements of a sort).

3 Set-theoretic background

In this section we shortly explain the set-theoretic axioms and constructions that form the basis of our model. We assume that the reader is familiar with the ZFC set theory. This is very similar to the theory that Lee and Werner [2011] use as the basis for their model. In particular, we use Zermelo-Fraenkel set theory with the axiom of choice (ZFC) together with an axiom that there is a countably infinite strictly increasing hierarchy of uncountable strongly inaccessible cardinals. In particular, we assume that we have a hierarchy of *strongly inaccessible* cardinals $\kappa_0, \kappa_1, \kappa_2, \ldots$ where $\kappa_0 > \omega$.

3.1 Von Neumann cumulative hierarchy and models of ZFC

The von Neumann cumulative hierarchy is a sequence of sets (indexed by ordinal numbers) that is cumulative. That is, each set in the hierarchy is a subset of the all sets after it. These sets are also referred to as von Neumann universes. This hierarchy is defined as follows for ordinal number α:

$$\mathcal{V}_\alpha \triangleq \bigcup_{\beta \in \alpha} \mathcal{P}\left(\mathcal{V}_\beta\right)$$

It is well-known [Drake, 1974] that whenever α is a strongly inaccessible cardinal number of a cardinality strictly greater than ω, as is the case for $\kappa_0, \kappa_2, \ldots$, $\mathcal{V}_\alpha$ is a model for ZFC. The von Neumann universe $\mathcal{V}_\omega$ satisfies all axioms of ZFC except for the axiom of infinity. This is due to the fact that all sets in $\mathcal{V}_\omega$ are *finite*. This is why we have assumed that our hierarchy of strngly inaccessible cardinals starts at one strictly greater than ω. In particular, if A and B are two sets in $\mathcal{V}_\kappa$ then $B^A \in \mathcal{V}_\kappa$ where B^A is the set of all functions from B to A. This allows us to use von Neumann universes to model the predicative pCuIC universes. For more details about strongly inaccessible cardinals and von Neumann universes refer to Drake [1974].

3.2 Rule sets and fixpoints: inductive constructions in set theory

Following Lee and Werner [2011], who follow Dybjer [1991] and Aczel [1999], we use inductive definitions (in set theory) constructed through rule sets to model inductive types. Here, we give a very short account of rule sets for inductive definitions. For further details refer to Aczel [1977].

A pair (A, a) is a *rule* based on a set U where $A \subseteq U$ is the set of premises and $a \in U$ is the conclusion. We usually write $\frac{A}{a}$ for a rule (A, a). A *rule set* based on U is a set Φ of rules based on U. We say a set $X \subseteq U$ is Φ-*closed*, $closed_\Phi(X)$ for a U based rule set Φ if we have:

$$closed_\Phi(X) \triangleq \forall \frac{A}{a} \in \Phi.\ A \subseteq X \Rightarrow a \in X$$

The operator $\mathcal{O}_\Phi$ corresponding to a rule set Φ is the operation of collecting all conclusions for a set whose premises are available in that set. That is,

$$\mathcal{O}_\Phi(X) \triangleq \left\{ a \middle| \frac{A}{a} \in \Phi \wedge A \subseteq X \right\}$$

Hence, a set X is Φ-closed if $\mathcal{O}_\Phi(X) \subseteq X$. Notice that $\mathcal{O}_\Phi$ is a monotone function on $\mathcal{P}(U)$ which is a complete lattice. Therefore, for any U based rule set Φ, the operator $\mathcal{O}_\Phi$ has a least fixpoint, $\mathcal{I}(\Phi) \subseteq U$:

$$\mathcal{I}(\Phi) \triangleq \bigcap \{X \subseteq U | closed_\Phi(X)\}$$

We define by transfinite recursion a sequence, indexed by ordinal numbers $\mathcal{O}_\Phi^\alpha$ for an ordinal number α:

$$\mathcal{O}_\Phi^\alpha \triangleq \bigcup_{\beta \in \alpha} \left(\mathcal{O}_\Phi^\beta \cup \mathcal{O}_\Phi(\mathcal{O}_\Phi^\beta) \right)$$

Obviously, for $\beta \leq \alpha$ we have $\mathcal{O}_\Phi^\beta \subseteq \mathcal{O}_\Phi^\alpha$.

Theorem 3.1 (Aczel [1977])**.** *For any rule set Φ there exists an ordinal number $ClOrd(\Phi)$ called the* closing ordinal *of Φ such that it is the smallest ordinal number for which we have $\mathcal{I}(\Phi) = \mathcal{O}_\Phi^{ClOrd(\Phi)}$ is the least fixpoint of $\mathcal{O}_\Phi$. In other words, $\mathcal{O}_\Phi^{ClOrd(\Phi)+1} = \mathcal{O}_\Phi(\mathcal{O}_\Phi^{ClOrd(\Phi)}) = \mathcal{O}_\Phi^{ClOrd(\Phi)}$.*

Lemma 3.2 (Aczel [1977])**.** *Let Φ be a rule set based on some set U and β a* regular *cardinal such that for every rule $\frac{A}{a} \in \Phi$ we have that $|A| < \beta$ then*

$$ClOrd(\Phi) \leq \beta$$

In other words, $\mathcal{O}_\Phi^\beta$ is the least fixpoint of $\mathcal{O}_\Phi$.

3.3 Fixpoints of large functions

A set theoretic constructions is called *large* with respect to a set theoretic (von Neumann) universe if it does not belong to that universe. As we shall see, the functions that we will consider for interpreting of inductive types (operators of certain rule sets) are indeed large. That is, they map subsets of the universe to subsets of the universe. As a result, the fixpoint of these functions might not have a fixpoint within the universe in question as the universe with the subset relation on it is not a complete lattice. The following lemmas show that under certain conditions, the fixpoint of the function induced by rule sets does exist in the desired universe. We will use this lemma to show that the interpretations of inductive types do indeed fall in the universe that they are supposed to.

Lemma 3.3. *Let Φ be a rule set based on the set-theoretic universe $\mathcal{V}$ and $\alpha \in \mathcal{V}$ be a cardinal number such that for all $\frac{A}{a} \in \Phi$ we have $|A| \leq \alpha$. Then,*

$$ClOrd(\Phi) \in \mathcal{V}$$

Proof. By Lemma 3.2, it suffices to show that there is a regular cardinal $\beta \in \mathcal{V}$ such that $|A| < \beta$ for any $\frac{A}{a} \in \Phi$. Take β to be $\aleph_{\alpha+1}$. By the fact that $\mathcal{V}$ is a model of ZFC we know that $\aleph_{\alpha+1} \in \mathcal{V}$. By the fact that $\alpha < \aleph_{\alpha+1}$ we know that $|A| < \aleph_{\alpha+1}$ for any $\frac{A}{a} \in \Phi$. It is well known that under axioms of ZFC (this crucially uses axiom of choice) $\aleph_{\alpha+1}$ is a regular cardinal for any ordinal number α – see Drake [1974] for a proof. This concludes our proof. □

Lemma 3.4. *Let Φ be a rule set based on the set-theoretic universe $\mathcal{V}$ and $\alpha \in \mathcal{V}$ be a cardinal number such that for all $\frac{A}{a} \in \Phi$ we have $|A| \leq \alpha$. Then,*

$$\mathcal{I}(\Phi) \in \mathcal{V}$$

Proof. By Lemma 3.2 we know that $\mathcal{I}(\Phi) = \mathcal{O}_{\Phi}^{ClOrd(\Phi)}$. We know that $\mathcal{O}_{\Phi}^{ClOrd(\Phi)} \in \mathcal{V}$ as it is constructed by transfinite recursion up to $ClOrd(\Phi)$ and that we crucially know that $ClOrd(\Phi) \in \mathcal{V}$ by Lemma 3.3. More precisely, this can be shown using transfinite induction up to $ClOrd(\Phi)$ showing that every stage belongs to $\mathcal{V}$. Notice that it is crucial for an ordinal to belong to the universe in order for transfinite induction to be valid. □

3.4 The use of axiom of choice

The only place in this work that we make use of axiom of choice is in the proof of Lemma 3.3. We use this axiom to show the following statement which we could have alternatively taken as a (possibly) weaker axiom.

> In any von Neumann universe $\mathcal{V}$ for any cardinal number α there is a *regular* cardinal β such that $\alpha < \beta$.

Note that his statement is independent of ZF and certain axiom, e.g., choice as we have taken here, needs to be postulated. This is due to the well-known fact proven by Gitik [1980] that under the assumption of existence of strongly compact cardinals, any uncountable cardinal is singular!

3.5 Modeling the impredicative sort Prop: trace encoding

One of the challenges in constructing a model for a system like pCuIC is treatment of an impredicative proof-irrelevant sort Prop. This can be done by simply modeling Prop as the set $\{\emptyset, \{\emptyset\}\}$ where provable propositions are modeled as $\{\emptyset\}$ and non-provable propositions as $\emptyset$. This however, will only work where we don't have the cumulativity relation between Prop and $\mathtt{Type}_i$. In presence of such cumulativity relations, such a naïve treatment of Prop breaks interpretation of the (dependent) function spaces as sets of functions. The following example should make the issue plain.

Example 3.5 (Werner [1997])**.** *Let's consider the interpretation of the term $I \equiv \boldsymbol{\lambda}(P : \mathtt{Type}_0).\, P \to P$. In this case, the semantics of $[\![I\ True]\!]$ will be $\{\emptyset\}$ or $\{\emptyset\}^{\{\emptyset\}}$ depending on whether $True : \mathtt{Prop}$ or $True : \mathtt{Type}_0$ is considered, respectively. And hence we should have $\{\emptyset\} = \{\emptyset\}^{\{\emptyset\}}$ which is not the case, even though the two sets are isomorphic (bijective).*

In order to circumvent this issue, we follow Lee and Werner [2011], who in turn follow Aczel [1999], in using the method known as *trace encoding* for representation of functions.

Definition 3.6 (Trace encoding)**.** *The following two functions,* Lam *and* App*, are used for trace encoding and application of trace encoded functions respectively.*

$$\mathsf{Lam}(f) \triangleq \bigcup_{(x,y)\in f} (\{x\} \times y)$$
$$\mathsf{App}(f, x) \triangleq \{y | (x, y) \in f\}$$

Lemma 3.7. *Let $f : X \to Y$ be a set theoretic function then for any $x \in X$ we have*

$$\mathsf{App}(\mathsf{Lam}(f), x) = f(x)$$

Note that using the trace encoding technique the problem mentioned in Example 3.5 is not present anymore. That is, we have:

$$\left\{\mathsf{Lam}(f)\middle| f \in \{\emptyset\}^{\{\emptyset\}}\right\} = \{\emptyset\}$$

Lemma 3.8 (Aczel [1999]). *Let A be a set and assume the set $B(x) \subseteq 1$ for $x \in A$.*

1. $\{\mathsf{Lam}(f)|f \in \Pi x \in A.\, B(x)\} \subseteq 1$
2. $\{\mathsf{Lam}(f)|f \in \Pi x \in A.\, B(x)\} = 1$ *iff* $\forall x \in A.\ B(x) = 1$

4 Set-theoretic model and consistency of pCuIC

We construct a model for pCuIC by interpreting predicative universes using von Neumann universes and `Prop` as $\{0,1\} = \{\emptyset, \{\emptyset\}\}$. We use the trace encoding technique presented earlier for (dependent) function types. We will construct the interpretation of inductive types and their eliminators using rule sets for inductive definitions in set theory. We shall first define a *size* function on terms, typing contexts, and pairs of a context and a term which we write as $size(\Gamma \vdash t)$. We will then define the interpretation of typing contexts and terms (in appropriate context) by well-founded recursion on their size.

Definition 4.1. *We define a function size on terms, contexts, declarations and pairs consisting of a context and a term (which we write as $size(\Gamma \vdash t)$) mutually recursively as follows:*

Size for typing contexts and declarations

$$\begin{aligned}
size(\cdot) &\triangleq 1/2\\
size(\Gamma, x : A) &\triangleq size(\Gamma) + size(A)\\
size(\Gamma, x := t : A) &\triangleq size(\Gamma) + size(t) + size(A)\\
size(\Gamma, \mathbf{Ind}_n\,\{\Delta_I := \Delta_C\}) &\triangleq size(\Gamma) + size(\mathbf{Ind}_n\,\{\Delta_I := \Delta_C\})
\end{aligned}$$

Size for term

$$\begin{aligned}
size(\texttt{Prop}) &\triangleq 1\\
size(\texttt{Type}_i) &\triangleq 1\\
size(x) &\triangleq 1\\
size(\mathbf{\Pi} x : A.\, B) &\triangleq size(A) + size(B) + 1\\
size(\boldsymbol{\lambda} x : A.\, t) &\triangleq size(A) + size(t) + 1\\
size(t\ u) &\triangleq size(t) + size(u) + 1\\
size(\mathbf{let}\, x := t : A\, \mathbf{in}\, u) &\triangleq size(t) + size(u) + size(A) + 1\\
size(\mathcal{D}.z) &\triangleq size(\mathcal{D})\\
size(\mathbf{Elim}(t; \mathcal{D}.d_i; \vec{Q})\left\{\vec{f}\right\}) &\triangleq size(t) + size(\mathcal{D}) + \sum_i size(Q_i) + \sum_i size(f_i) + 1
\end{aligned}$$

Size for pairs consisting of a context and a term

$$size(\mathbf{Ind}_n\,\{\Delta_I := \Delta_C\}) \triangleq \sum_{d \in \mathsf{dom}(\Delta_I)} size(\Delta_I(d)) + \sum_{c \in \mathsf{dom}(\Delta_C)} size(\Delta_C(c)) + 1$$

Size for judgements

$$size(\Gamma \vdash t) \triangleq size(\Gamma) + size(t) - 1/2$$
$$size(\Gamma \vdash \mathbf{Ind}_n \{\Delta_I := \Delta_C\}) \triangleq size(\Gamma) + size(\mathbf{Ind}_n \{\Delta_I := \Delta_C\}) - 1/2$$

In the definition above, which is similar to that by Lee and Werner [2011] and Miquel and Werner [2003], the $\pm 1/2$ is add to make sure $size(\Gamma \vdash t) < size(\Gamma, x : t)$ and that $size(\Gamma) < size(\Gamma \vdash t)$.

Definition 4.2 (The model)**.** *We define the interpretations for contexts and terms by well-founded recursion on their sizes.*

Interpretation of contexts

$$\begin{aligned}
[\![\cdot]\!] &\triangleq \{\mathsf{nil}\} \\
[\![\Gamma, x : A]\!] &\triangleq \left\{ \gamma, a \middle| \gamma \in [\![\Gamma]\!] \wedge [\![\Gamma \vdash A]\!]_\gamma \downarrow \wedge a \in [\![\Gamma \vdash A]\!]_\gamma \right\} \\
[\![\Gamma, x := t : A]\!] &\triangleq \left\{ \gamma, a \middle| \gamma \in [\![\Gamma]\!] \wedge [\![\Gamma \vdash A]\!]_\gamma \downarrow \wedge [\![\Gamma \vdash t]\!]_\gamma \downarrow \wedge a = [\![\Gamma \vdash t]\!]_\gamma \in [\![\Gamma \vdash A]\!]_\gamma \right\} \\
[\![\Gamma, \mathbf{Ind}_n \{\Delta_I := \Delta_C\}]\!] &\triangleq [\![\Gamma]\!] \text{ if } [\![\Gamma \vdash \mathbf{Ind}_n \{\Delta_I := \Delta_C\}]\!]_\gamma \downarrow \text{ for all } \gamma \in [\![\Gamma]\!]
\end{aligned}$$

Above, we assume that $x \notin \mathsf{dom}(\Gamma)$, otherwise, both $[\![\Gamma, x : A]\!]$ and $[\![\Gamma, x := t : A]\!]$ are undefined.

Interpretation of terms

$$\begin{aligned}
[\![\Gamma \vdash \texttt{Prop}]\!]_\gamma &\triangleq \{\emptyset, \{\emptyset\}\} \\
[\![\Gamma \vdash \texttt{Type}_i]\!]_\gamma &\triangleq \mathcal{V}_{\kappa_i} \\
[\![\Gamma \vdash x]\!]_{\vec{a}} &\triangleq a_{\mathsf{len}(\Gamma_1)-l} \qquad \text{if } \Gamma = \Gamma_1, x : A, \Gamma_2 \text{ and } x \notin \mathsf{dom}(\Gamma_1) \cup \mathsf{dom}(\Gamma_2) \\
&\qquad \text{and } l = \mathsf{len}(\mathit{Inds}(\Gamma_1)) \\
[\![\Gamma \vdash \mathbf{\Pi} x : A.\, B]\!]_\gamma &\triangleq \left\{ \mathsf{Lam}(f) \middle| f : \Pi a \in [\![\Gamma \vdash A]\!]_\gamma .\ [\![\Gamma, x : A \vdash B]\!]_{\gamma,a} \right\} \\
[\![\Gamma \vdash \boldsymbol{\lambda} x : A.\, t]\!]_\gamma &\triangleq \mathsf{Lam}\left(\left\{ (a, [\![\Gamma, x : A \vdash t]\!]_{\gamma,a}) \middle| a \in [\![\Gamma \vdash A]\!]_\gamma \right\} \right) \\
[\![\Gamma \vdash t\ u]\!]_\gamma &\triangleq \mathsf{App}([\![\Gamma \vdash t]\!]_\gamma, [\![\Gamma \vdash u]\!]_\gamma) \\
[\![\Gamma \vdash \mathbf{let}\, x := t : A\, \mathbf{in}\, u]\!]_\gamma &\triangleq [\![\Gamma, x := t : A \vdash u]\!]_{\gamma, [\![\Gamma \vdash u]\!]_\gamma}
\end{aligned}$$

Interpretation of inductive types, constructors and eliminators is defined below

4.1 Modeling inductive types

We define the interpretation for inductive blocks by constructing a rule set which will interpret the whole inductive block. We will define interpretation of individual inductive types and constructors of the block based on the fixpoint of this rule set.

Definition 4.3 (Interpretation of the inductive types)**.** *Let $\mathcal{D} \equiv \mathsf{Ind}_n \{\Delta_I := \Delta_C\}$ be an arbitrary but fixed inductive block such that $\Delta_I = d_0 : A_0, \dots d_l : A_l$ and $\Delta_C = c_0 : T_0, \dots c_{l'} : T_{l'}$. Where $A_i \equiv \mathbf{\Pi} \vec{p} : \vec{P}. \mathbf{\Pi} \vec{x} : \vec{B_i}. s$ for some sequence of types $\vec{B}$ and some sort s. Here, $\vec{P}$ are parameters of the inductive block. The type of constructors are of the following form: $T_k \equiv \mathbf{\Pi} \vec{p} : \vec{P}. \mathbf{\Pi} \vec{x} : \vec{V_k}. d_{i_k}\ \vec{p}\ \vec{t_k}$ for some $\vec{t_k}$. Notice that $\vec{V_k}$ is a sequence itself. That is, it is of the form $\vec{V_k} \equiv V_{k,1}, \dots, V_{k,\mathsf{len}(\vec{V_K})}$. That is, each constructor c_k, takes a number of arguments, first parameters $\vec{P}$ and then some more $\vec{V_k}$. The strict positivity condition implies that for any non-parameter argument $V_{k,i}$ of a constructor c_k, either $d \in \mathsf{dom}(\Delta_I)$ does not appear in $V_{k,i}$ or we have $V_{k,i} \equiv \mathbf{\Pi} \vec{x} : \overrightarrow{W_{k,i}}. d_{I_{k,i}}\ \vec{q}\ \vec{t}$ where $\mathsf{len}(\vec{q}) = \mathsf{len}(\vec{p})$. That is, each argument of a constructor where an inductive type (of the same block) appears, is a (dependent) function with codomain being that inductive type. In this case, no inductive type of the block appears the domain(s) of the function, $\overrightarrow{W_{k,i}}$. Notice that the codomain of the function is an inductive type in the block but not necessarily of the same family as the one being defined – the parameters applied are $\vec{q}$ instead of $\vec{p}$! We write $rec(V_{k,i})$ if some inductive of the block appears in $V_{k,i}$ in which case it will be of the form just described.*

$$[\![\Gamma \vdash \mathcal{D}]\!]_\gamma \triangleq \mathcal{I}(\Phi^\gamma_{\Gamma,\mathcal{D}})$$

$$\Phi^\gamma_{\Gamma,\mathcal{D}} \triangleq \bigcup_{d_i \in \mathsf{dom}(\Delta_I)} \bigcup_{c_k \in \mathsf{Constrs}(d_i)} \left\{ \frac{\Psi_{d_i,c_k}}{\psi_{d_i,c_k}} \;\middle|\; \begin{array}{l} \vec{a} \in [\![\Gamma \vdash \vec{P}]\!]_\gamma, \\ \vec{m} \in [\![\Gamma, \vec{p} : \vec{P}, \vec{d'} : \vec{s} \vdash \vec{V'_k}]\!]_{\gamma,\vec{a},\vec{b}} \end{array} \right\}$$

Where, $\vec{d'}$ is the sequence of $d'_{I_{k,j}}$'s occurring within $\vec{V'_k}$ below and s_i is the sort of the inductive type d_i, $\vec{b} \in [\![\Gamma, \vec{p} : \vec{P} \vdash \vec{s}]\!]_{\gamma,\vec{a}}$ and,

$$\psi_{d_i,c_k} \triangleq \left\langle i; \vec{a}; [\![\Gamma, \vec{p} : \vec{P}, \vec{d'} : \vec{s}, \vec{x} : \vec{V'_k} \vdash \vec{t_k}]\!]_{\gamma,\vec{a},\vec{b},\vec{m}}; \langle k; \vec{a}, \vec{m}\rangle \right\rangle$$

$$\Psi_{d_i,c_k} \triangleq \bigcup_{rec(V_{k,j})} \left\{ \left\langle \begin{array}{l} \langle I_{k,j}; \\ [\![\Gamma_{k,j}, \vec{y} : \overrightarrow{W_{k,j}} \vdash \vec{q}]\!]_{\gamma,\vec{a}.\vec{b},\vec{e},\vec{u}}; \\ [\![\Gamma_{k,j}, \vec{y} : \overrightarrow{W_{k,j}} \vdash \vec{t}]\!]_{\gamma,\vec{a},\vec{b},\vec{e},\vec{u}}; \\ \overrightarrow{\mathsf{App}}(m_{I_{k,j}}, \vec{u}) \end{array} \right\rangle \;\middle|\; \vec{u} \in [\![\Gamma_{k,j} \vdash \overrightarrow{W_{k,j}}]\!]_{\gamma,\vec{a},\vec{b},\vec{e}} \right\}$$

$$\Gamma_{k,j} \triangleq \Gamma, \vec{p} : \vec{P}, \vec{d'} : \vec{s}, x_1 : V'_{k,1}, \dots, x_{j-1} : V'_{k,j-1}$$

$$\vec{e} \triangleq m_1, \dots m_{j-1}$$

For Ψ_k we assume that $V_{k,j} \equiv \mathbf{\Pi} \vec{y} : \overrightarrow{W_{k,j}}. d_{I_{k,j}}\ \vec{q}\ \vec{t}$.

The types $V'_{k,j}$ are defined based on $V_{k,j}$ as follows:

$$V'_{k,j} \triangleq \begin{cases} \mathbf{\Pi} \vec{x} : \overrightarrow{W_{k,i}}. d'_{I_{k,j}} & \text{if } rec(V_{k,j}) \text{ and } V_{k,i} \equiv \mathbf{\Pi} \vec{x} : \overrightarrow{W_{k,i}}. d_{I_{k,j}}\ \vec{q}\ \vec{t} \\ V_{k,j} & \text{otherwise} \end{cases}$$

We define the interpretation of the inductive types in the block as follows:

$$\begin{aligned}\left[\!\left[\Gamma \vdash \mathcal{D}.d_i\right]\!\right]_\gamma &\triangleq \overrightarrow{\mathsf{Lam}}(f_{d_i})\\ f_{d_i}(\vec{a},\vec{t}) &\triangleq \left\{\langle k;\vec{a},\vec{m}\rangle \middle| \langle i;\vec{a};\vec{t};\langle k;\vec{a},\vec{m}\rangle\rangle \in \left[\!\left[\Gamma \vdash \mathcal{D}\right]\!\right]_\gamma\right\}\\ &\quad \textit{for } \vec{a},\vec{t} \in \left[\!\left[\Gamma \vdash \vec{P},\vec{B_i}\right]\!\right]_\gamma\end{aligned}$$

We define the interpretation of the constructors in the block as follows:

$$\begin{aligned}\left[\!\left[\Gamma \vdash \mathcal{D}.c_k\right]\!\right]_\gamma &\triangleq \overrightarrow{\mathsf{Lam}}(g_{c_k})\\ g_{c_k}(\vec{a},\vec{m}) &\triangleq \langle k;\vec{a},\vec{m}\rangle \ \textit{ for } \vec{a},\vec{m} \in \left[\!\left[\Gamma \vdash \vec{P},\vec{V_k}\right]\!\right]_\gamma\end{aligned}$$

Let us first discuss the intuitive construction of Definition 4.3 above. In this definition the most important part is the interpretation of the inductive block. We have defined this as the fixpoint of the rule sets corresponding to constructors of the inductive type. This rule set basically spells out the following. Take, some set $d'_{I_{k,j}}$ in the universe as a candidate for the interpretation of j^{th} occurrence of inductive type $d_{I_{k,j}}$ in the k^{th} constructor. This candidate, $d'_{I_{k,j}}$, is taken to be a candidate element of the inductive type $d_{I_{k,j}}$ in case $\overrightarrow{W_{k,i}} = \mathsf{nil}$, i.e., intuitively, if the recursive occurrence *embeds* an element of $d_{I_{k,j}}$ in the type being constructed. On the other hand, if the recursive occurrence of $d_{I_{k,j}}$ is, intuitively, *embedding* a function with codomain $d_{I_{k,j}}$ into the type being constructed then $d'_{I_{k,j}}$ is to be understood as the codomain of the function being embedded by the constructor. In each of these two cases, we need to make sure the candidate element and or function is *correctly chosen*, i.e., we need to make sure that the element or the *range* of the function chosen is indeed in the interpretation of the inductive type. This is where the rule sets come to play, so to speak. The premise set Ψ_k makes sure that all candidate recursive occurrences are indeed correctly chosen. This is done by basically making sure that for any of the arguments of the function being embedded (here an element is treated as a function with no arguments!) the result of applying the candidate function to the arguments is indeed in the interpretation of the corresponding inductive type. Do notice that $\overrightarrow{\mathsf{App}}(a,\mathsf{nil}) = a$.

Example 4.4 (Rule set for construction of natural numbers)**.**

$$\Phi^{\gamma}_{\Gamma,\mathcal{N}} = \left\{\frac{\emptyset}{\langle 0;\mathsf{nil};\mathsf{nil};\langle 0;\mathsf{nil}\rangle\rangle}\right\} \cup \left\{\frac{\{\langle 0;\mathsf{nil};\mathsf{nil};a\rangle\}}{\langle 0;\mathsf{nil};\mathsf{nil};\langle 1;a\rangle\rangle}\middle| a \in \mathcal{V}\kappa_0\right\}$$

Example 4.5 (Rule set for construction of lists)**.**

$$\Phi^{\gamma}_{\Gamma,\mathcal{L}_i} = \left\{\frac{\emptyset}{\langle 0;A;\mathsf{nil};\langle 0;A\rangle\rangle}\right\} \cup \left\{\frac{\{\langle 0;A;\mathsf{nil},b\rangle\}}{\langle 0;A;\mathsf{nil};\langle 1;A,a,b\rangle\rangle}\middle| b \in \mathcal{V}\kappa_0 \wedge a \in \left[\!\left[\Gamma \vdash A\right]\!\right]_\gamma\right\}$$

Lemma 4.6. *Values for arguments of arities are uniquely determined by the values for arguments each constructor (including values for parameters) in* $\mathcal{O}^{\alpha}_{\Phi^{\gamma}_{\Gamma,\mathcal{D}}}$ *and in particular in* $\left[\!\left[\Gamma \vdash \mathcal{D}\right]\!\right]_\gamma$*. That is, if*

$$\langle i;\vec{a};\vec{t};\langle k;\vec{a},\vec{m}\rangle\rangle, \left\langle i;\vec{a};\vec{t'};\langle k;\vec{a},\vec{m}\rangle\right\rangle \in \mathcal{O}^{\alpha}_{\Phi^{\gamma}_{\Gamma,\mathcal{D}}}$$

then, $\vec{t} = \vec{t'}$*. Analogously for* $\left[\!\left[\Gamma \vdash \mathcal{D}\right]\!\right]_\gamma$ *we have that if*

$$\langle i;\vec{a};\vec{t};\langle k;\vec{a},\vec{m}\rangle\rangle, \left\langle i;\vec{a};\vec{t'};\langle k;\vec{a},\vec{m}\rangle\right\rangle \in \left[\!\left[\Gamma \vdash \mathcal{D}\right]\!\right]_\gamma$$

then, $\vec{t} = \vec{t'}$*.*

Proof. This immediately follows from the fact that for any two rules

$$\frac{X}{\langle i; \vec{a}; \vec{t}; \langle k; \vec{a}, \vec{m}\rangle\rangle} \quad \text{and} \quad \frac{X}{\left\langle i; \vec{a}; \vec{t'}; \langle k; \vec{a}, \vec{m}\rangle\right\rangle}$$

in $\Phi^{\gamma}_{\Gamma,\mathcal{D}}$ we have $\vec{t} = \vec{t'}$. □

We shall show that the interpretation of the inductive types in a block are each in the universe corresponding to their sort. Notice that whenever two inductive types appear in one another the syntactic criteria for typing enforce that they are both have the same sort.[2] Therefore, we assume without loss of generality that all inductive types in the block have the same sort. In case it is not the case, it must be there are some inductive types in the block that are not necessarily mutually inductive with the rest of the block. Hence, those inductive types (and their interpretations) can be considered prior to considering the block as a whole. Therefore, in the following theorem, we assume, without loss of generality that the all of the inductive types of a block are of the same sort.

Lemma 4.7. *Let $\mathcal{D} \equiv \mathsf{Ind}_n \{\Delta_I := \Delta_C\}$ be a block of inductive types with all inductive types of sort* `Type`$_i$*. Furthermore, let us assume that all the terms (and particularly types) appearing in the body of the block are well defined under the context Γ and environment γ and interpretation of each of these terms (and types) is in the interpretation of the type (correspondingly sort) that is expected based on the typing derivation.*[3] *Then, $[\![\Gamma \vdash \mathcal{D}]\!]_\gamma\downarrow$, $[\![\Gamma \vdash \mathcal{D}]\!]_\gamma \in \mathcal{V}\kappa_i$ and $[\![\Gamma \vdash \mathcal{D}.d_j]\!]_\gamma \in [\![\Gamma \vdash \Delta_I(d_j)]\!]_\gamma$.*

Proof. The construction of $[\![\Gamma \vdash \mathcal{D}]\!]_\gamma$ depends only on the interpretation of terms $[\![\Gamma \vdash u]\!]_\gamma$ where u appears in Δ_I or Δ_C and by our assumptions these are all defined. Therefore, we can easily see that $[\![\Gamma \vdash \mathcal{D}]\!]_\gamma\downarrow$.

We proceed by showing first that $[\![\Gamma \vdash \mathcal{D}]\!]_\gamma \in \mathcal{V}\kappa_i$. By Lemma 3.4 we need to show that there is a cardinal number α in the universe $\mathcal{V}\kappa_i$ such that for any rule $\frac{\phi}{\psi}$, we have $|\phi| \leq \alpha$. Notice that the cardinality of each rule in the interpretation of the inductive block is fixed depending on which constructor the rule corresponds to. Since there are finitely many constructors, we can take α to be the supremum of these cardinalities. For each constructor, this cardinality is the supremum of the cardinality corresponding to each (mutually) recursive arguments of the constructor. Each recursive argument of the a constructor is a function type with codomain being an inductive type in the block where, *crucially*, no inductive type of the same block appears in the domain. Therefore, the cardinality of the antecedents corresponding to each (mutually) recursive argument of a constructor is simply the cardinality of the domain, i.e., product of cardinalities of different arguments of the aforementioned function. This latter cardinality is by our assumptions is in the universe $\mathcal{V}\kappa_i$.

By the fact that $[\![\Gamma \vdash \mathcal{D}]\!]_\gamma \in \mathcal{V}\kappa_i$, as we just proved it, and by construction of the model it follows easily that $[\![\Gamma \vdash \mathcal{D}.d_j]\!]_\gamma \in [\![\Gamma \vdash \Delta_I(d_j)]\!]_\gamma$. □

Lemma 4.8. *Let $\mathcal{D} \equiv \mathsf{Ind}_n \{\Delta_I := \Delta_C\}$ and $\mathcal{D}' \equiv \mathsf{Ind}_n \{\Delta'_I := \Delta'_C\}$ be two blocks of inductive types with $\mathsf{dom}(\Delta_I) = \mathsf{dom}(\Delta'_I)$ and $\mathsf{dom}(\Delta_C) = \mathsf{dom}(\Delta'_C)$. Furthermore, assume that the for each $d \in \mathsf{dom}(\Delta_I)$ the interpretation of the arguments of the arity of $\Delta_C(d)$ are subsets of the interpretation of corresponding arguments of the arity of $\Delta_C(d')$. Similarly for the arguments of the constructors. In addition, assume that in each case, the interpretation of values given as parameters and arities in the resulting type of each corresponding constructors (the inductive*

[2] Note that this is the case in our work as there are no inductive types in `Prop`.

[3] These conditions will hold by induction hypotheses when this lemma is used in practice.

type being constructed by that constructor) are equal. These are conditions in the rule IndLeq *in Figure 5 where cumulativity (subtyping) relation is replaced with subset relation on the interpretation and the judgemental equality is replaced with equality of interpretations.*[4] *In this case,* $[\![\Gamma \vdash \mathcal{D}]\!]_\gamma \subseteq [\![\Gamma \vdash \mathcal{D}']\!]_\gamma$ *and consequently* $[\![\Gamma \vdash \mathcal{D}.d]\!]_\gamma \subseteq [\![\Gamma \vdash \mathcal{D}'.d]\!]_\gamma$

Proof. By transfinite induction on α up to the closing ordinal of $\Phi^\gamma_{\Gamma,\mathcal{D}}$ we show that $\mathcal{O}^\alpha_{\Phi^\gamma_{\Gamma,\mathcal{D}}} \subseteq \mathcal{O}^\alpha_{\Phi^\gamma_{\Gamma,\mathcal{D}'}}$. The base case and the case of limit ordinals are trivial. We only need to show the result for successor ordinals, i.e.,

$$\mathcal{O}^{\alpha^+}_{\Phi^\gamma_{\Gamma,\mathcal{D}}} = \mathcal{O}_{\Phi^\gamma_{\Gamma,\mathcal{D}}}(\mathcal{O}^\alpha_{\Phi^\gamma_{\Gamma,\mathcal{D}}}) \subseteq \mathcal{O}_{\Phi^\gamma_{\Gamma,\mathcal{D}'}}(\mathcal{O}^\alpha_{\Phi^\gamma_{\Gamma,\mathcal{D}'}}) = \mathcal{O}^{\alpha^+}_{\Phi^\gamma_{\Gamma,\mathcal{D}'}}$$

where by induction hypothesis we know that $\mathcal{O}^\alpha_{\Phi^\gamma_{\Gamma,\mathcal{D}}} \subseteq \mathcal{O}^\alpha_{\Phi^\gamma_{\Gamma,\mathcal{D}'}}$. The result follows by the fact that any rule in the rule set $\Phi^\gamma_{\Gamma,\mathcal{D}}$ is also a rule in the rule set $\Phi^\gamma_{\Gamma,\mathcal{D}'}$ guaranteed by the subset relations in our assumptions which are consequences of their corresponding subtypings in the rules pertaining to cumulativity of inductive types. □

4.2 Modeling eliminators

We define the interpretation for eliminators by constructing a rule set which will interpret the eliminators for the whole inductive block. We will define interpretation of individual eliminators based on the fixpoint of this rule set.

Definition 4.9 (Interpretation of recursors)**.** *Let* $\mathcal{D} \equiv \mathsf{Ind}_n \{\Delta_I := \Delta_C\}$ *an arbitrary but fixed inductive block and assume, without loss of generality, that* $\Delta_I = d_0 : A_0, \dots d_l : A_l$ *and* $\Delta_C = c_0 : T_0, \dots c_{l'} : T_{l'}$*. Where* $A_i \equiv \mathbf{\Pi} \vec{p} : \vec{P}. \mathbf{\Pi} \vec{x} : \vec{B}. s$ *for some types* $\vec{B}$ *and some sort* s*. Here,* $\vec{P}$ *are parameters of the inductive block. The type of constructors are of the following form:* $T_k \equiv \mathbf{\Pi} \vec{p} : \vec{P}. \mathbf{\Pi} \vec{x} : \overrightarrow{V_k}. d_{i_k}\ \vec{p}\ \vec{t_k}$ *for some* $\vec{t_k}$*. For the sake of simplicity of presentation, let us write* $ELB \equiv \mathsf{Elim}^{\mathcal{D}}(Q_{d_1}, \dots, Q_{d_l}) \{f_{c_1}, \dots, f_{c_{l'}}\}$ *for the block of eliminators being interpreted. Furthermore, let* $\xi^{Q_{d_1},\dots,Q_{d_l}}_{\mathcal{D}}(c_k, T_k) \equiv \mathbf{\Pi} \vec{x} : \overrightarrow{U_k}. Q_{d_{i_k}}\ \vec{u}$ *for some terms* $\vec{u}$*. Notice that* $\xi^{Q_{d_1},\dots,Q_{d_l}}_{\mathcal{D}}$ *is simply the type of the constructor where after each (mutually) recursive argument, an argument is added for the result of the elimination of that (mutually) recursive argument. Let us write* $J_{k,i}$ *for the index of the* i^{th} *argument of the* k^{th} *constructor,* $V_{k,i}$*, in* $\overrightarrow{U_k}$ *above. More precisely, whenever* $rec(V_{k,i})$ *holds, we have* $U_{J_{k,i}+1}$ *is the argument of* $\xi^{Q_{d_1},\dots,Q_{d_l}}_{\mathcal{D}}(c_k, T_k)$ *that corresponds to the result of the elimination of* $V_{k,i} = U_{J_{k,i}}$*. We first define a rule set* $\Phi^\gamma_{\Gamma,ELB}$ *for this interpretation:*

$$\Phi^\gamma_{\Gamma,ELB} \triangleq \bigcup_{d_i \in \mathsf{dom}(\Delta_I)} \bigcup_{c_k \in \mathsf{Constrs}(d_i)} \left\{ \frac{\Psi_{d_i,c_k}}{\psi_{d_i,c_k}} \middle| \vec{a} \in \left[\!\!\left[\Gamma \vdash \overrightarrow{U_k}\right]\!\!\right]_\gamma \right\}$$

Let $\vec{b}$ *be the subsequence of* $\vec{a}$ *corresponding to arguments of the constructor, i.e., it is obtained from* $\vec{a}$ *by dropping any term in the sequence that corresponds to some* $U_{J_{k,j}+1}$ *whenever* $rec(V_{k,j})$*.*

$$\begin{aligned}
\psi_{d_i,c_k} &\triangleq \left\langle \left\langle c_k; \vec{b} \right\rangle ; \overrightarrow{\mathsf{App}}([\![\Gamma \vdash f_{c_k}]\!]_\gamma, \vec{a}) \right\rangle \\
\Psi_{d_i,c_k} &\triangleq \bigcup_{rec(V_{k,j})} \left\{ \left\langle \begin{array}{l} \overrightarrow{\mathsf{App}}([\![\Gamma_{k,j} \vdash U_{J_{k,j}}]\!]_{\gamma,\vec{e}}, \vec{u}); \\ \overrightarrow{\mathsf{App}}([\![\Gamma_{k,j} \vdash U_{J_{k,j}+1}]\!]_{\gamma,\vec{e}}, \vec{u}) \end{array} \right\rangle \middle| \vec{u} \in \left[\!\!\left[\Gamma_{k,j} \vdash \overrightarrow{W_{k,j}}\right]\!\!\right]_{\gamma,\vec{e}} \right\} \\
\Gamma_{k,j} &\triangleq \Gamma, \vec{p} : \vec{P}, x_1 : V_{k,1}, \dots, x_{j-1} : V_{k,j-1} \\
\vec{e} &\triangleq b_1, \dots b_{j-1}
\end{aligned}$$

[4] These conditions will hold by the induction hypothesis when we shall use this lemma.

For Ψ_k *we assume that* $V_{k,j} \equiv \mathbf{\Pi}\, \vec{y} : \overrightarrow{W_{k,j}}.\, d_{I_{k,j}}\ \vec{q}\ \vec{t}$.

We define the interpretation individual eliminators as follows:

$$\left[\!\left[\Gamma \vdash \mathbf{Elim}(t; \mathcal{D}.d_i; Q_{d_1}, \ldots, Q_{d_l})\left\{f_{c_1}, \ldots, f_{c_{l'}}\right\}\right]\!\right]_\gamma \triangleq u$$

if there is a unique u *such that* $\langle t, u\rangle \in \mathcal{I}(\Phi^\gamma_{\Gamma,ELB})$ *and undefined otherwise.*

The following lemma shows that in Definition 4.9 above we have indeed captured and interpreted all of the elements of all of the inductive types in the block.

Lemma 4.10. *Let* $\mathcal{D} \equiv \mathbf{Ind}_n\{\Delta_I := \Delta_C\}$ *be a block of inductive types with* $[\![\Gamma \vdash \mathcal{D}]\!]_\gamma$ *defined and let* $Q_{d_1}, \ldots, Q_{d_l}$ *and* $f_{c_1}, \ldots, f_{c_{l'}}$ *be such that*

$$[\![\Gamma \vdash Q_{d_i}]\!]_\gamma \in \left[\!\left[\Gamma \vdash \mathbf{\Pi}\vec{x} : \vec{A}.\, (d_i\ \vec{x}) \to s'\right]\!\right]_\gamma$$

and

$$[\![\Gamma \vdash f_{c_i}]\!]_\gamma \in \left[\!\left[\Gamma \vdash \xi^{\vec{Q}}_{\mathcal{D}}(c_i, \Delta_C(c_i))\right]\!\right]_\gamma$$

Also, assume that for the term t *we have* $[\![\Gamma \vdash t]\!]_\gamma \in [\![\Gamma \vdash \mathcal{D}.d\ \vec{u}]\!]_\gamma$. *Then,*

$$\left[\!\left[\Gamma \vdash \mathbf{Elim}(t; \mathcal{D}.d; Q_{d_1}, \ldots, Q_{d_l})\left\{f_{c_1}, \ldots, f_{c_{l'}}\right\}\right]\!\right]_\gamma\!\downarrow$$

and

$$\left[\!\left[\Gamma \vdash \mathbf{Elim}(t; \mathcal{D}.d; Q_{d_1}, \ldots, Q_{d_l})\left\{f_{c_1}, \ldots, f_{c_{l'}}\right\}\right]\!\right]_\gamma \in [\![\Gamma \vdash Q_{d_i}\ \vec{u}\ t]\!]_\gamma$$

Proof. We show by transfinite induction up to the closing ordinal of $\Phi^\gamma_{\Gamma,\mathcal{D}}$ that for any α and for any $\left\langle\langle c; \vec{a}, \vec{m}\rangle; \vec{t}\right\rangle \in \left\{\left\langle\langle c; \vec{a}, \vec{m}\rangle; \vec{t}\right\rangle \middle| \langle d_i; \vec{a}; \vec{t}; \langle c; \vec{a}, \vec{m}\rangle\rangle \in \mathcal{O}^\alpha_{\Phi^\gamma_{\Gamma,\mathcal{D}}}\right\}$ (note that by Lemma 4.6 $\vec{t}$ is uniquely determined by c $\vec{a}$ and $\vec{m}$) there is a unique $b \in \left[\!\left[\Gamma \vdash \overrightarrow{\mathsf{App}}(Q_{d_i}, \vec{a}, \vec{t}, \langle c; \vec{a}, \vec{m}\rangle)\right]\!\right]_\gamma$ such that $\langle\langle c; \vec{a}, \vec{m}\rangle; b\rangle \in \mathcal{O}^\alpha_{\Phi^\gamma_{\Gamma,ELB}}$ for

$$ELB \equiv \mathbf{Elim}^{\mathcal{D}}(Q_{d_1}, \ldots, Q_{d_l})\left\{f_{c_1}, \ldots, f_{c_{l'}}\right\}$$

For the base case, $\alpha = 0$ this holds trivially. For the other cases, it suffices to notice that all the argument elements taken for the (mutually) recursive arguments of constructors, i.e., corresponding to $U_{J_{k,j}}$ for $rec(V_{J_{k,j}})$ are *uniquely* determined from the arguments of the constructor ($\vec{b}$ in the interpretation of recursors above) as is so restricted by the antecedents of each rule.

Notice that the argument above also shows that the closing ordinal of $\Phi^\gamma_{\Gamma,ELB}$ and $\Phi^\gamma_{\Gamma,ELB}$ are the same which concludes the proof. □

4.3 Proof of Soundness

We show that the model that we have constructed throughout this section is sound. That is, we show that for any typing context Γ, term t and type A such that $\Gamma \vdash t : A$ we have that for any environment $\gamma \in [\![\Gamma]\!]$, $[\![\Gamma \vdash t]\!]_\gamma\!\downarrow$, $[\![\Gamma \vdash A]\!]_\gamma\!\downarrow$ and that $[\![\Gamma \vdash t]\!]_\gamma \in [\![\Gamma \vdash A]\!]_\gamma$. We use this result to prove consistency of pCuIC.

Lemma 4.11. *Let Γ be a typing context, γ be an environment, t such that $[\![\Gamma \vdash t]\!]_\gamma\downarrow$. Then $FV(t) \subseteq \mathsf{dom}(\Gamma)$.*

Proof. Follows easily by induction on t. □

Lemma 4.12 (Weakening)**.** *Let Γ be a typing context, γ be an environment, t and A be terms such that $[\![\Gamma, \Gamma']\!]\downarrow$, $\gamma \in [\![\Gamma]\!]$, $\gamma, \gamma' \in [\![\Gamma, \Gamma']\!]$ and $[\![\Gamma, \Gamma' \vdash t]\!]_{\gamma,\gamma'}\downarrow$. Furthermore, let Ξ be a typing context and δ be an environment such that $(\mathsf{dom}(\Gamma) \cup \mathsf{dom}(\Gamma')) \cap \mathsf{dom}(\Xi) = \emptyset$, and we have that variables in $\mathsf{dom}(\Xi)$ do not appear in Γ' freely such that $[\![\Gamma, \Xi, \Gamma']\!]\downarrow$ and $\gamma, \delta, \gamma' \in [\![\Gamma, \Xi, \Gamma']\!]$. Then*

1. $[\![\Gamma, \Xi, \Gamma' \vdash t]\!]_{\gamma,\delta,\gamma'}\downarrow$
2. $[\![\Gamma, \Xi, \Gamma' \vdash t]\!]_{\gamma,\delta,\gamma'} = [\![\Gamma, \Gamma' \vdash t]\!]_{\gamma,\gamma'}$

Proof. We prove the result above by induction on t. Most cases follow easily by induction hypotheses. Here, for demonstration purposes we show the case for variables and (dependent) function types.

- $t = x$: In this case, both Case 1 and Case 2 above follow by definition of the model.
- $t = \mathbf{\Pi} x : A.\, B$: We know by induction hypothesis that

$$[\![\Gamma, \Xi, \Gamma' \vdash A]\!]_{\gamma,\delta,\gamma'}\downarrow$$

$$[\![\Gamma, \Xi, \Gamma' \vdash A]\!]_{\gamma,\delta,\gamma'} = [\![\Gamma, \Gamma' \vdash A]\!]_{\gamma,\gamma'}$$

  Also by induction hypotheses we have that for all $a \in [\![\Gamma, \Xi, \Gamma' \vdash A]\!]_{\gamma,\delta,\gamma'}$,

$$([\![\Gamma, \Xi, \Gamma', x : A \vdash B]\!]_{\gamma,\delta,\gamma',a})\downarrow$$

$$[\![\Gamma, \Xi, \Gamma', x : A \vdash B]\!]_{\gamma,\delta,\gamma',a} = [\![\Gamma, \Gamma', x : A \vdash B]\!]_{\gamma,\gamma',a}$$

  Given the above induction hypotheses, both Case 1 and Case 2 above follow by the definition of the model. Note that by Lemma 4.11, we know that variables in $\mathsf{dom}(\Xi)$ do not appear freely in A. This is why induction hypothesis applies to B above. □

Lemma 4.13 (Substitutivity)**.** *Let Γ be a typing context, γ be an environment, u and A be terms such that $[\![\Gamma]\!]\downarrow$, $\gamma \in [\![\Gamma]\!]$ and $[\![\Gamma \vdash u]\!]_\gamma\downarrow$. Then*

1. *If $[\![\Gamma, x : A, \Xi]\!]\downarrow$ and $\gamma, [\![\Gamma \vdash u]\!]_\gamma, \delta \in [\![\Gamma, x : A, \Xi]\!]$ then $\gamma, \delta \in [\![\Gamma, \Xi\,[u/x]]\!]$*
2. *If $[\![\Gamma, x : A, \Xi]\!]\downarrow$ and $\gamma, [\![\Gamma \vdash u]\!]_\gamma, \delta \in [\![\Gamma, x : A, \Xi]\!]$ and $[\![\Gamma, x : A, \Xi \vdash t]\!]_{\gamma,[\![\Gamma\vdash u]\!]_\gamma,\delta}\downarrow$ then*
   (a) $[\![\Gamma, \Xi\,[u/x] \vdash t\,[u/x]\,]\!]_{\gamma,\delta}\downarrow$
   (b) $[\![\Gamma, \Xi\,[u/x] \vdash t\,[u/x]\,]\!]_{\gamma,\delta} =$
   $[\![\Gamma, x : A, \Xi \vdash t]\!]_{\gamma,[\![\Gamma\vdash u]\!]_\gamma,\delta} = [\![\Gamma, x := u : A, \Xi \vdash t]\!]_{\gamma,[\![\Gamma\vdash u]\!]_\gamma,\delta}$

Proof. We prove this lemma by well-founded induction on $\sigma(\Xi, t) = size(\Xi) + size(t) - 1/2$. In the following we reason as though we are conducting induction on the structure of terms and contexts. Note that this is allowed because our measure $\sigma(\Xi, t)$ does decrease for the induction hypotheses pertaining to structural induction in each case $\sigma(\Xi, t)$ is decreasing. This is in particular crucial in some sub-cases of the case where $\Xi = \Xi, y : B$. There, we are performing induction on t which enlarges the context Ξ (similar to case of (dependent) functions in Lemma 4.12).

- Case $\Xi = \cdot$: then Case 1 holds trivially. Case 2, follows by induction on t. The only non-trivial case (not immediately following from the induction hypothesis) is the case where t is a variable. In this case, both 2a and 2b follow by definition of the model.
- Case $\Xi = \Xi', y : B$ and correspondingly, $\delta = \delta', a$:
  1. Follows by definition of the model and the induction hypothesis corresponding to Case 2b: $[\![\Gamma, \Xi'\,[u/x] \vdash B\,[u/x]]\!]_{\gamma,\delta'} = [\![\Gamma, x : A, \Xi' \vdash B]\!]_{\gamma,[\![\Gamma\vdash u]\!]_\gamma,\delta'}$
  2. We proceed by induction on t. The only non-trivial case is when t is a variable, $t = z$. Notice, when $z \neq x$ then both 2a and 2b hold trivially by definition of the model. Otherwise, we have to show that $[\![\Gamma, \Xi'\,[u/x], y : B\,[u/x] \vdash u]\!]_{\gamma,\delta,a}\downarrow$ and $[\![\Gamma, \Xi\,[u/x], y : B\,[u/x] \vdash u]\!]_{\gamma,\delta,a} = [\![\Gamma, x : A, \Xi, y : B\,[u/x] \vdash x]\!]_{\gamma,[\![\Gamma\vdash u]\!]_\gamma,\delta.a} = [\![\Gamma, x := u : A, \Xi, y : B\,[u/x] \vdash x]\!]_{\gamma,[\![\Gamma\vdash u]\!]_\gamma,\delta,a}$. These follow by Lemma 4.12 above and our assumption that $[\![\Gamma \vdash u]\!]_\gamma\downarrow$.
- Case $\Xi = \Xi', \mathbf{Ind}_n\,\{\Delta_I := \Delta_C\}$: We prove this case by induction on t. The only non-trivial case here is the case where $t = \mathbf{Ind}_n\,\{\Delta_I := \Delta_C\}\,.z$. Notice that the interpretation of $[\![\Gamma, \Xi'\,[u/x], y : B\,[u/x] \vdash t]\!]_{\gamma,\delta,a}$ is defined based on the interpretation of terms of the form $[\![\Gamma, \Xi'\,[u/x] \vdash m]\!]_{\gamma,\delta}$ where m appears in Δ_I or Δ_C to which the induction hypothesis (that of induction on Ξ) applies. □

Theorem 4.14 (Soundness of the model)**.** *The model defined in this section is sound for our typing system. That is, the following statements hold:*

1. *If* $\mathcal{WF}(\Gamma)$ *then* $[\![\Gamma]\!]\downarrow$
2. *If* $\Gamma \vdash t : A$ *then* $[\![\Gamma]\!]\downarrow$ *and for any* $\gamma \in [\![\Gamma]\!]$ *we have* $[\![\Gamma \vdash t]\!]_\gamma\downarrow$, $[\![\Gamma \vdash A]\!]_\gamma\downarrow$ *and* $[\![\Gamma \vdash t]\!]_\gamma \in [\![\Gamma \vdash A]\!]_\gamma$
3. *If* $\Gamma \vdash t \simeq t' : A$ *then* $[\![\Gamma \vdash t]\!]_\gamma\downarrow$, $[\![\Gamma \vdash t']\!]_\gamma\downarrow$, $[\![\Gamma \vdash A]\!]_\gamma\downarrow$ *and* $[\![\Gamma \vdash t]\!]_\gamma = [\![\Gamma \vdash t']\!]_\gamma \in [\![\Gamma \vdash A]\!]_\gamma$
4. *If* $\Gamma \vdash A \preceq B$ *then* $[\![\Gamma \vdash A]\!]_\gamma\downarrow$, $[\![\Gamma \vdash B]\!]_\gamma\downarrow$ *and* $[\![\Gamma \vdash A]\!]_\gamma \subseteq [\![\Gamma \vdash B]\!]_\gamma$

Proof. Note that judgements of the form $\mathcal{WF}(\Gamma)$, $\Gamma \vdash t : A$, $\Gamma \vdash t \simeq t' : A$ and $\Gamma \vdash A \preceq B$ are defined mutually. We prove the theorem by mutual induction on the derivation of these judgements.

- Case WF-CTX-EMPTY: trivial by definition.
- Case WF-CTX-HYP: trivial by definition and induction hypothesis.
- Case WF-CTX-DEF: trivial by definition and induction hypothesis.
- Case PROP: trivial by definition.
- Case HIERARCHY: trivial by definition.
- Case VAR: trivial by definition and induction hypotheses.
- Case LET: by definition, induction hypothesis and Lemma 4.13.
- Case LET-EQ: by definition, induction hypotheses and Lemma 4.13.

- Case Prod: by induction hypotheses we know that $[\![\Gamma \vdash A]\!]_\gamma \in [\![\Gamma \vdash s_1]\!]_\gamma$ and that $[\![\Gamma, x : A \vdash B]\!]_{\gamma,a} \in [\![\Gamma \vdash s_2]\!]_\gamma$ for any $a \in [\![\Gamma \vdash A]\!]_\gamma$. We also know $\mathcal{R}_s(s_1, s_2, s_3)$. We have to show that

$$[\![\Gamma \vdash \mathbf{\Pi} x : A.\, B]\!]_\gamma \in [\![\Gamma \vdash s_3]\!]_\gamma \tag{1}$$

  Since $\mathcal{R}_s(s_1, s_2, s_3)$ holds, we have that either $s_2 = s_3 = \texttt{Prop}$ or $s_1 = \texttt{Type}_i$, $s_2 = \texttt{Type}_j$ and $s_3 = \texttt{Type}_{\mathsf{max}\{i,j\}}$ or $s_1 = \texttt{Prop}$, $s_2 = \texttt{Type}_i$ and $s_3 = \texttt{Type}_i$ hold. In the first case, The membership relation above, (1), follows from Case 1 of Lemma 3.8. In the other two cases, note that $[\![\Gamma \vdash s_3]\!]_\gamma$ is a von Neumann universe and is hence closed under (dependent) function space and also (by axiom schema of replacement) under taking trace-encoding of elements of any set.

- Case Prod-eq: by definition and induction hypotheses.

- Case Lam: by definition and induction hypotheses.

- Case Lam-eq: by definition and induction hypotheses.

- Case App: by induction hypotheses we know that $[\![\Gamma \vdash N]\!]_\gamma \in [\![\Gamma \vdash A]\!]_\gamma$ and $[\![\Gamma \vdash M]\!]_\gamma \in [\![\Gamma \vdash \mathbf{\Pi} x : A.\, B]\!]_\gamma$. From the latter we know that $[\![\Gamma \vdash M]\!]_\gamma = \mathsf{Lam}(f)$ for some f with domain $\mathsf{dom}(f) = [\![\Gamma \vdash A]\!]_\gamma$ such that $f(a) = [\![\Gamma, x : A \vdash B]\!]_{\gamma,a}$. Therefore,

$$\begin{aligned}[\![\Gamma \vdash M\ N]\!]_\gamma &= \mathsf{App}(\mathsf{Lam}(f), [\![\Gamma \vdash N]\!]_\gamma) \\ &= f([\![\Gamma \vdash N]\!]_\gamma) \in [\![\Gamma, x : A \vdash B]\!]_{\gamma, [\![\Gamma \vdash N]\!]_\gamma} = [\![\Gamma \vdash B\,[N/x]]\!]_\gamma\end{aligned}$$

  where the last equality is by Lemma 4.13.

- Case App-eq: by a reasoning similar to that of Case App above.

- Case Ind-WF: by definition and induction hypotheses and Lemma 4.7.

- Case Ind-type: by induction hypotheses we know that $[\![\Gamma]\!]\downarrow$ which by definition means that for any environment $\gamma \in [\![\Gamma]\!]$ and any inductive block $\mathcal{D} \in \Gamma$ we have (using the weakening lemma above) $[\![\Gamma \vdash \mathcal{D}]\!]_\gamma\downarrow$. By Lemma 4.7 the desired result follows.

- Case Ind-constr: by induction hypotheses we know that $[\![\Gamma]\!]\downarrow$ which by definition means that for any environment $\gamma \in [\![\Gamma]\!]$ and any inductive block $\mathcal{D} \in \Gamma$ we have (using the weakening lemma above) $[\![\Gamma \vdash \mathcal{D}]\!]_\gamma\downarrow$. The desired result follows by the definition of $[\![\Gamma \vdash \mathcal{D}.c]\!]_\gamma$, Lemma 4.13 and the fact that $[\![\Gamma \vdash \mathcal{D}]\!]_\gamma\downarrow$ is the fixpoint of the rule-set used to construct it.

- Case Ind-Elim: by Lemma 4.10 and induction hypotheses.

- Case Ind-Elim-eq: similarly to Case Ind-Elim.

- Case Eq-ref: trivial by definition and induction hypothesis.

- Case Eq-sym: trivial by definition and induction hypothesis.

- Case Eq-trans: trivial by definition and induction hypothesis.

- Case Beta: by induction hypotheses that

$$[\![\Gamma, x : A \vdash M]\!]_\gamma\downarrow$$

$$\forall a \in [\![\Gamma \vdash A]\!]_\gamma \,.\, [\![\Gamma, x : A \vdash M]\!]_{\gamma,a} \in [\![\Gamma, x : A \vdash B]\!]_{\gamma,a}$$

On the other hand, we have that

$$[\![\Gamma \vdash N]\!]_\gamma\!\downarrow$$
$$[\![\Gamma \vdash N]\!]_\gamma \in [\![\Gamma \vdash A]\!]_\gamma$$
$$[\![\Gamma \vdash A]\!]_\gamma\!\downarrow$$

and therefore

$$[\![\Gamma, x : A \vdash B]\!]_{\gamma,[\![\Gamma\vdash N]\!]_\gamma} \in [\![\Gamma \vdash s]\!]_\gamma$$

Also, by Lemma 4.13 we get that

$$[\![\Gamma \vdash B\,[N/x]]\!]_\gamma = [\![\Gamma, x : A \vdash B]\!]_{\gamma,[\![\Gamma\vdash N]\!]_\gamma} \in [\![\Gamma \vdash s]\!]_\gamma$$

and that

$$[\![\Gamma \vdash M\,[N/x]]\!]_\gamma = [\![\Gamma, x : A \vdash M]\!]_{\gamma,[\![\Gamma\vdash u]\!]_\gamma} \in [\![\Gamma, x : A \vdash B]\!]_{\gamma,[\![\Gamma\vdash u]\!]_\gamma}$$

We finish the proof by:

$$\begin{aligned}
[\![\Gamma \vdash (\boldsymbol{\lambda} x : A.\, M)\ N]\!]_\gamma &= \mathsf{App}([\![\Gamma \vdash \boldsymbol{\lambda} x : A.\, M]\!]_\gamma, [\![\Gamma \vdash N]\!]_\gamma) \\
&= \mathsf{App}(\mathsf{Lam}(a \in [\![\Gamma \vdash A]\!]_\gamma \mapsto [\![\Gamma, x : A \vdash M]\!]_{\gamma,a}), [\![\Gamma \vdash N]\!]_\gamma) \\
&= [\![\Gamma, x : A \vdash M]\!]_{\gamma,[\![\Gamma\vdash N]\!]_\gamma} \\
&= [\![\Gamma \vdash M\,[N/x]]\!]_\gamma
\end{aligned}$$

- Case Eta: We need to show that

$$[\![\Gamma \vdash t]\!]_\gamma = [\![\Gamma \vdash \boldsymbol{\lambda} x : A.\, t\ x]\!]_\gamma \in [\![\Gamma \vdash \boldsymbol{\Pi} x : A.\, B]\!]_\gamma$$

We know by induction hypothesis that $[\![\Gamma \vdash t]\!]_\gamma\!\downarrow$ and that $[\![\Gamma \vdash t]\!]_\gamma \in [\![\Gamma \vdash \boldsymbol{\Pi} x : A.\, B]\!]_\gamma$ and consequently we know that there is a set theoretic function f:

$$f \in \Pi a \in [\![\Gamma \vdash A]\!]_\gamma.\ [\![\Gamma, x : A \vdash B]\!]_{\gamma,a}$$

such that $[\![\Gamma \vdash t]\!]_\gamma = \mathsf{Lam}(f)$. This implies that,

$$\begin{aligned}
[\![\Gamma \vdash \boldsymbol{\lambda} x : A.\, t\ x]\!]_\gamma &= \mathsf{Lam}\left(\left\{(a, [\![\Gamma, x : A \vdash t\ x]\!]_{\gamma,a}) \middle| a \in [\![\Gamma \vdash A]\!]_\gamma\right\}\right) \\
&= \mathsf{Lam}\left(\left\{(a, \mathsf{App}([\![\Gamma, x : A \vdash t]\!]_{\gamma,a}, a)) \middle| a \in [\![\Gamma \vdash A]\!]_\gamma\right\}\right) \\
&= \mathsf{Lam}\left(\left\{(a, \mathsf{App}([\![\Gamma \vdash t]\!]_{\gamma,a}, a)) \middle| a \in [\![\Gamma \vdash A]\!]_\gamma\right\}\right) \\
&= \mathsf{Lam}\left(\left\{(a, \mathsf{App}(\mathsf{Lam}(f), a)) \middle| a \in [\![\Gamma \vdash A]\!]_\gamma\right\}\right) \\
&= \mathsf{Lam}\left(\left\{(a, f(a)) \middle| a \in [\![\Gamma \vdash A]\!]_\gamma\right\}\right) \\
&= \mathsf{Lam}(f) \\
&= [\![\Gamma \vdash t]\!]_\gamma
\end{aligned}$$

- Case Delta: by Lemma 4.13 and induction hypotheses.

- Case ZETA: trivial by definition and induction hypothesis.
- Case IOTA: by definition of the interpretation of recursors and induction hypothesis. Notice that by construction of the interpretation of eliminators in Definition 4.9 the interpretation of elimination of $c_i\ \vec{a}$ is basically, the result of applying $f_{c_i}\ \vec{b}$ where $\vec{a}$ is a subsequence of $\vec{b}$. The only difference between $\vec{b}$ and $\vec{a}$ is that in $\vec{b}$ after each value that corresponds to a (mutually) recursive argument of the constructor c_i we have a value that corresponds to the interpretation of elimination of that (mutually) recursive argument. For those terms, $\mu_{\mathcal{D}}^{\vec{Q};\vec{f}}$ applies the elimination using the eliminator, **Elim**, while the rule in rule set corresponding $c_i\ \vec{a}$ ensures in its antecedents that all elements are eliminated correctly according to the interpretation of the eliminator (the fixpoint taken in Definition 4.9). Notice that if the constructor is of an inductive sub-type, then, by construction of interpretation of constructors the interpretation of the two constructors applied to those terms are *equal*.
- Case PROP-IN-TYPE: trivial by definition.
- Case CUM-TYPE: trivial by definition.
- Case CUM-TRANS: trivial by definition and induction hypothesis.
- Case CUM-WEAKEN: trivial by definition, induction hypothesis and Lemma 4.12.
- Case CUM-PROD: trivial by definition and induction hypothesis.
- Case CUM-EQ-L: trivial by definition and induction hypothesis.
- Case CUM-EQ-R: trivial by definition and induction hypothesis.
- Case CUM: trivial by definition and induction hypothesis.
- Case CUM-EQ: trivial by definition and induction hypothesis.
- Case EQ-CUM: trivial by definition and induction hypothesis.
- Case C-IND: easy by definition, induction hypothesis and Lemma 4.8.
- Case IND-EQ: by definition, induction hypotheses and Lemma 4.8.
- Case CONSTR-EQ-L: by definition, induction hypothesis.
- Case CONSTR-EQ-R: by definition, induction hypothesis. □

Corollary 4.15 (Consistency of pCuIC)**.** *Let s be a sort, then, there* does not exist *any term t such that $\cdot \vdash t : \mathbf{\Pi} x : s.\, x$.*

Proof. If there where such a term t by Theorem 4.14 we should have $[\![\cdot \vdash t]\!]_{\mathsf{nil}} \in [\![\cdot \vdash \mathbf{\Pi} x : s.\, x]\!]_{\mathsf{nil}}$. However, $[\![\cdot \vdash \mathbf{\Pi} x : s.\, x]\!]_{\mathsf{nil}} = \emptyset$. □

Strong normalization We believe that our extension to pCIC maintains strong normalization and that the model constructed by Barras [2012] for pCIC could be easily extended to support our added rules.

4.4 The model and axioms of type theory

Although our system does not explicitly feature any of the axioms mentioned below, they are consistent with the model that we have constructed.

Our model is a proof-irrelevant model. That is, all provable propositions (terms of type `Prop`) are interpreted identically. Therefore, it satisfies the axiom of proof irrelevance and also the axiom of propositional extensionality (that any two logically equivalent propositions are equal). This model also satisfies definitional/judgemental proof irrelevance for proposition. This is similar to how Agda treats irrelevant arguments Abel [2011].

We do not support inductive types in the sort `Prop` in our system. However, if the Paulin-style equality is encoded using inductive types in higher sorts, then the interpretation of these types would simply be collections of reflexivity proofs. Hence, our model supports the axiom UIP (unicity of identity proofs) and consequently all other logically equivalent axioms, e.g., axiom K Streicher [1993].

This model, being a set theoretic model, also supports the axiom of functional extensionality as set theoretic functions are extensional. This is indeed why our model supports η-equivalence.

All these axioms are also supported by the model constructed by Lee and Werner [2011].

Acknowledgements This work was partially supported by the CoqHoTT ERC Grant 637339 and partially by the Flemish Research Fund grants G.0058.13 (until June 2017) and G.0962.17N (since July 2017).

References

A. Abel. *Irrelevance in Type Theory with a Heterogeneous Equality Judgement*, pages 57–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-19805-2. doi: 10.1007/978-3-642-19805-2_5. URL https://doi.org/10.1007/978-3-642-19805-2_5.

P. Aczel. An introduction to inductive definitions. *Studies in Logic and the Foundations of Mathematics*, 90:739 – 782, 1977. ISSN 0049-237X. doi: http://dx.doi.org/10.1016/S0049-237X(08)71120-0. URL http://www.sciencedirect.com/science/article/pii/S0049237X08711200. HANDBOOK OF MATHEMATICAL LOGIC.

P. Aczel. On relating type theories and set theories. In T. Altenkirch, B. Reus, and W. Naraschewski, editors, *Types for Proofs and Programs: International Workshop, TYPES' 98 Kloster Irsee, Germany, March 27–31, 1998 Selected Papers*, pages 1–18, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. doi: 10.1007/3-540-48167-2_1. URL https://doi.org/10.1007/3-540-48167-2_1.

B. Barras. Semantical investigation in intuitionistic set theory and type theoris with inductive families, 2012. Habilitation thesis, University Paris Diderot – Paris 7.

F. R. Drake. *Set theory : an introduction to large cardinals*. Studies in logic and the foundations of mathematics 76. North-Holland, Amsterdam, 1974. ISBN 0720422795.

P. Dybjer. *Inductive Sets and Families in Martin-Löf's Type Theory and Their Set-Theoretic Semantics*, pages 280–306. Cambridge University Press, Cambridge, 1991.

M. Gitik. All uncountable cardinals can be singular. *Israel Journal of Mathematics*, 35(1):61–88, Sep 1980. ISSN 1565-8511. doi: 10.1007/BF02760939. URL https://doi.org/10.1007/BF02760939.

G. Lee and B. Werner. Proof-irrelevant model of CC with predicative induction and judgmental equality. *Logical Methods in Computer Science*, 7(4), 2011. doi: 10.2168/LMCS-7(4:5)2011. URL http://dx.doi.org/10.2168/LMCS-7(4:5)2011.

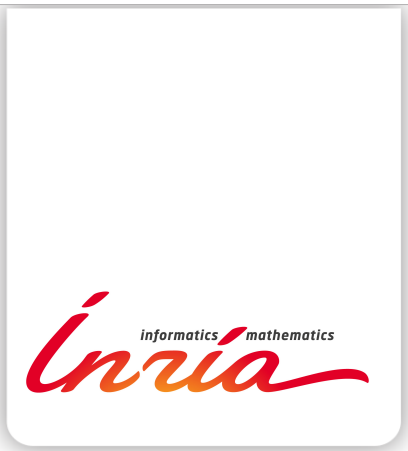
A. Miquel and B. Werner. *The Not So Simple Proof-Irrelevant Model of CC*, pages 240–258. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-540-39185-2. doi: 10.1007/3-540-39185-1_14. URL https://doi.org/10.1007/3-540-39185-1_14.

T. Streicher. Investigations into intensional type theory, 1993. Habilitiation thesis, Ludwig Maximilian Universität.

The Coq Development Team. The coq proof assistant, version 8.7+beta2., Oct. 2017. URL https://doi.org/10.5281/zenodo.1003421.

A. Timany and B. Jacobs. First steps towards cumulative inductive types in CIC. In *Theoretical Aspects of Computing - ICTAC 2015, Proceedings*, pages 608–617, Cali, Colombia, 2015. Springer. doi: 10.1007/978-3-319-25150-9_36. URL http://dx.doi.org/10.1007/978-3-319-25150-9_36.

B. Werner. Sets in types, types in sets. In M. Abadi and T. Ito, editors, *Theoretical Aspects of Computer Software: Third International Symposium, TACS'97 Sendai, Japan, September 23–26, 1997 Proceedings*, pages 530–546, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. ISBN 978-3-540-69530-1. doi: 10.1007/BFb0014566. URL https://doi.org/10.1007/BFb0014566.



RESEARCH CENTRE
PARIS

2 rue Simone Iff - CS 42112
75589 Paris Cedex 12

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399