HAL

archives-ouvertes.fr

CoqMTU: a higher-order type theory with a predicative hierarchy of universes parametrized by a decidable first-order theory

Bruno Barras, Jean-Pierre Jouannaud, Pierre-Yves Strub, Qian Wang

CoQMTU: a higher-order type theory with a predicative hierarchy of universes parametrized by a decidable first-order theory

Bruno Barras
INRIA Saclay - Île de France
Orsay, France
bruno.barras@inria.fr

Jean-Pierre Jouannaud
INRIA-LIAMA
Beijing, China
jeanpierre.jouannaud@gmail.com

Pierre-Yves Strub
INRIA-MSR Joint Center
Orsay, France
pierre-yves@strub.nu

Qian Wang
TNLIST, Tsinghua University
Beijing, China
q-w04@mails.tsinghua.edu.cn

Abstract—We study a complex type theory, a Calculus of Inductive Constructions with a predicative hierarchy of universes and a first-order theory $\mathcal{T}$ built in its conversion relation. The theory $\mathcal{T}$ is specified abstractly, by a set of constructors, a set of defined symbols, axioms expressing that constructors are free and defined symbols completely defined, and a generic elimination principle relying on crucial properties of first-order structures satisfying the axioms. We first show that CoQMTU enjoys all basic meta-theoretical properties of such calculi, confluence, subject reduction and strong normalization when restricted to weak-elimination, implying the decidability of type-checking in this case as well as consistency. The case of strong elimination is left open.

I. INTRODUCTION

Modern proof assistants based on the Curry-Howard isomorphism are now used for solving practical problems in various applicative areas, see for example the Trusted Labs' website.[1] These provers are considered secure because they allow the user to prove properties by building (possibly huge) proof terms which are then routinely checked by a trusted kernel. Since the design of such proof assistants allows them to be easily extended, they may incorporate sophisticated logical constructs before they can be proved sound with respect to the existing calculus.

For example, COQ is based on the calculus of constructions, but incorporates as well inductive types, co-inductive types, a predicative hierarchy of universes, implicit arguments, and a module system. The extension COQMT of COQ also includes the possibility of dynamically loading a decision procedure for a first-order theory which is then used in the conversion rule of the obtained calculus. So far, there is a paper proof of decidability of type checking (DTC) for CIC [1], as well as

[1]Trusted Labs has developed in Coq a formal model of a Java Card platform (including a virtual machine, a byte-code verifier and a linker for this language), and later on a proof that applet isolation is enforced by the platform, in the context of an open smart card. Such models passed the highest levels of certification of the Common Criteria standard. See for instance http://www.trusted-labs.com/spip.php?rubrique33.

a partial formal proof in COQ [2]. There is a paper proof of DTC for Luo's Extended Calculus of Constructions with weak elimination [3], an extension for the case of strong elimination restricted to a single predicative universe [4], and a paper proof of consistency which includes strong elimination together with arbitrarily many predicative universes [5]. There are paper proofs of DTC for various versions of CIC modulo a theory $\mathcal{T}$ [6]–[8], and a formal proof of the last version. Calculi with module systems on the one hand, and with implicit arguments on the other hand have been studied separately in the context of CIC without universes [9], [10]. One could argue that proving DTC for a calculus modeling the type theory underlying COQ is just a matter of man power, but this is wrong. Luo's proof of DTC for CIC with universes and weak elimination does not scale to richer calculi with strong elimination. Inferring types for implicit arguments does not scale to built-in theories. And so on.

Objective. Our program is to bridge this gap between theory and practice by eventually showing DTC for a type theory including all features currently available in COQ, possibly revisiting some, like inductive types as well as including a few others like type classes. The program will be considered completed when a formal proof in COQ of DTC for such a type theory will be carried out, possibly under some strong set-theoretic assumptions to make the encoding in COQ possible, therefore allowing us to state that an implementation of the type theory is a secure proof assistant.

Contribution. A first main contribution is a definition of a parametrized elimination principle for an abstract first-order theory $\mathcal{T}$ constrained by three natural axioms: non-triviality, constructor-freeness, and completeness of definitions. This elegant notion captures various previous attempts of the authors which led to the implementation of COQMT. Our second contribution is the definition of COQMTU, and the proof that type-checking in COQMTU is decidable when only weak-elimination is allowed in the first-order structure $\mathcal{T}$. The case of strong elimination is left open, and discussed in conclusion.

Significance. Besides being a step towards our program, COQMTU allows to encode Westbrook's Calculus of Nominal

Inductive Constructions, a recent advance towards the use of higher-order encodings in intensional type theory via a *nominal equational theory* expressing freshness conditions for variable names [11]. We therefore provide a proof of DTC (and consistency) for CNIC when restricted to weak elimination.

Correctness. Only instructive proofs are given here. A complete version of the paper can be found on the fourth author website.[2] With the exception of strong normalization, our results are assessed by a development in COQ available on the third author website.[3]

II. ABSTRACT CALCULUS

A. First-order theory $\mathcal{T}$

We consider a first-order mono-sorted algebra defined by a *sort* o, a non-empty set of *constructor symbols* $\mathcal{C}$, a set of *defined symbols* $\mathcal{D}$, and a set of *variables* $\mathcal{X}$. We denote by $\mathcal{T}(\mathcal{C} \uplus \mathcal{D}, \mathcal{X})$ the set of (first-order) terms, $\mathcal{T}(\mathcal{C}, \mathcal{X})$ the set of constructor terms, $\mathcal{T}(\mathcal{D}, \mathcal{X})$ the set of defined terms, and drop the letter $\mathcal{X}$ for the respective sets of *ground* terms. Constructors and defined symbols are equipped with a fixed *arity*. We use t_Λ for the top function symbol in the term t, $\bar{t}$ for its n-tuple of *immediate subterms*, and $|t|$ for its size. *Substitutions* are finite sets of pairs made of a variable and a term, written $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. We write $t\theta$ for the *instance* of t by the substitution θ.

The semantics of the defined symbols is specified in an abstract form by a *decidable congruence* $\leftrightarrow^*_\mathcal{T}$ over $\mathcal{T}(\mathcal{C} \uplus \mathcal{D}, \mathcal{X})$ (also called first-order theory) such that:

Freeness. For all constructor terms s, t, $s \leftrightarrow^*_\mathcal{T} t$ iff $s = t$.

Non-triviality. $\mathcal{T}(\mathcal{C})$ contains at least two different constructor terms.

Completeness. For all terms t in $\mathcal{T}(\mathcal{C} \uplus \mathcal{D})$, there exists a term u in $\mathcal{T}(\mathcal{C})$ (unique by *Freeness*) such that $t \leftrightarrow^*_\mathcal{T} u$.

Terms s, t such that $s \leftrightarrow^*_\mathcal{T} t$ are called $\mathcal{T}$-*equivalent*. The following technical lemma follows from *Non-triviality*:

Lemma II-A.1. *Let u, v be first-order terms such that $u \leftrightarrow^*_\mathcal{T} v$. Then, there exists a term w that* simplifies *u and v, that is $u \leftrightarrow^*_\mathcal{T} w$, $Var(w) \subseteq Var(u) \cap Var(v)$, and $w_\Lambda \in \mathcal{C}$ if either u or v has a constructor root.*

Proof: Let u, v be first-order terms such that $u \leftrightarrow^*_\mathcal{T} v$, assuming wlog that $u_\Lambda \in \mathcal{C}$ if either u or v has a constructor root. The proof is by induction on $Y = Var(u) \setminus Var(v)$. If $Y = \emptyset$, let $w = u$. Otherwise, let $u' = u\{x \mapsto g\}$ for some $x \in Y$ and $g \in \mathcal{T}(\mathcal{C} \uplus \mathcal{D})$ (using *Non-triviality*). By definition of Y, $v = v\{x \mapsto g\}$. Since $\leftrightarrow^*_\mathcal{T}$ is a congruence, $u' \leftrightarrow^*_\mathcal{T} v$. Since $u'_\Lambda = u_\Lambda$, we conclude by application of the induction hypothesis to u', v. ∎

Quite important, popping up constructors from arbitrary terms by using $\mathcal{T}$ cannot be done *ad libitum*:

Lemma II-A.2. *Let $t \in \mathcal{T}(\mathcal{C} \uplus \mathcal{D}, \mathcal{X})$. Then, there exists a natural number n such that for all u, v satisfying (i) $t \leftrightarrow^*_\mathcal{T} u$ and (ii) $u = v\theta$ with $v \in \mathcal{T}(\mathcal{C}, \mathcal{X})$, we have $|v| < n$.*

Proof: By contradiction. We assume some term $t \in \mathcal{T}(\mathcal{C} \uplus \mathcal{D}, \mathcal{X})$ such that, for any natural number n, there exists some pair (u, v) satisfying (i, ii) and $|v| \geq n$. Then, there must exists an infinite sequence of terms $\{v_i\theta_i\}_i$ with $|v_i| \geq n_i$, where $\{n_i\}_i$ is a strictly increasing sequence of natural numbers. Let now σ be a substitution replacing all variables in $\mathcal{X}$ by some (arbitrary) ground term. Then, $t\sigma$ has infinitely many equivalent ground terms $v_i\theta_i\sigma$. By *Completeness* there exists $w_i \in \mathcal{T}(\mathcal{C})$ such that $w_i \leftrightarrow^*_\mathcal{T} v_i\theta_i\sigma$. By *Freeness*, w_i is unique, and since $\leftrightarrow^*_\mathcal{T}$ is a congruence, $w_i = v_i\gamma_i$, where γ_i is a constructor substitution, hence $|w_i| \geq |v_i|$. Consider w_1 and v_k such that $|v_k| > |w_1|$, hence $|w_k| \geq |v_k| > |w_1|$. Then w_1 and w_k must be different, contradicting *Freeness*. ∎

Note that our first-order framework is slightly restrictive, since we have assumed that the first-order algebra is mono-sorted. Moving to a multi-sorted algebra is easy, to the price of more complex notations.

Our paradigmatic first-order theory is Presburger Arithmetic, whose well-known alphabet is $\mathcal{C} = \{\mathbf{0}, \mathbf{S}\}$ and $\mathcal{D} = \{+\}$. Presburger arithmetic satisfies our axioms, as well as a useful stronger form of completeness that gives more computational strength:

Extensional completeness For all terms $t \in \mathcal{T}(\mathcal{C} \uplus \mathcal{D}, \mathcal{X}) \setminus \mathcal{T}(\mathcal{D}, \mathcal{X})$, there exists a term $u \in \mathcal{T}(\mathcal{C} \uplus \mathcal{D}, \mathcal{X})$ such that (i) $t \leftrightarrow^*_\mathcal{T} u$ and (ii) $u_\Lambda \in \mathcal{C}$.

Note that we only use the unquantified fragment of Presburger arithmetic. For example, we could have taken the unquantified fragment of Peano arithmetic, which is decidable as well.

B. Pseudo-terms

Since our abstract calculus contains the calculus of constructions, universes, and a first-order theory $\leftrightarrow^*_\mathcal{T}$, its term language contains the usual term constructions of CC, universes and terms from the first-order language. Incorporating the latter into the type-theoretic language is easily done by declaring the first-order function symbol as higher-order constants in the calculus. In the example of Presburger arithmetic, this gives $\mathbf{0} : \mathbf{nat}, \mathbf{S} : \mathbf{nat} \to \mathbf{nat}$, and $+: \mathbf{nat} \to \mathbf{nat} \to \mathbf{nat}$. Then, *fully applied* terms like $(+\ \mathbf{0}\ \mathbf{0})$ correspond to first-order terms, while non-fully applied ones like $(+\ \mathbf{0})$ do not.

We describe our language via BNF-style definitions:

Universes: our universes are classically $\mathbf{Prop}$ and $\mathbf{Type}_j$, where j is a strictly positive integer. In the sequel, we shall identify $\mathbf{Prop}$ with $\mathbf{Type}_0$ whenever convenient. This goes against the COQ tradition of identifying $\mathbf{Prop}$ with $\mathbf{Type}_{-1}$ and Set with $\mathbf{Type}_0$, but is a natural fit with the predicativity of Set. Then,

$$s := \mathbf{Type}_{j \geq 0}$$

Variables: variables are elements of the denumerable set $\mathcal{V}ar$ containing $\mathcal{X}$ as a subset.

First-order constants: we denote by o the type of our first-order algebraic expressions, and (abusing notations) by $\mathcal{C}$ and $\mathcal{D}$ the sets of higher-order constants corresponding to the constructors and defined symbols respectively.

Pseudo terms: The pseudo terms are defined as usual:

$$t, u, T, U ::= s \mid o \mid \mathcal{C} \mid \mathcal{D} \mid \mathcal{V}ar$$
$$\mid \; t\,u \mid \lambda[x:U].\,t \mid \forall(x:U).\,T$$
$$\mid \; \mathrm{ELIM}_o(T, \vec{u}, t).$$

In the case of Presburger arithmetic, $\mathrm{ELIM}_n(T, u, v, t)$ stands for the more usual Gödel's (primitive) recursor at higher type $rec(T, t, u, v)$. We denote by $\mathrm{FV}(t)$ the set of free variables of a term t, and by $|t|$ its size, counting the non-variable nodes of its tree representation.

Pseudo substitutions: A (pseudo) *substitution* of *domain* $\mathcal{D}om(\theta) = \{x_1, \ldots, x_n\}$ is a sequence $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ where the x_i's are distinct variables and the t_i's are terms. A substitution θ acts on a term u by replacing all the free occurrences of the variables x_i's in u by the corresponding t_i's, possibly renaming bound variables. We shall sometimes abbreviate $\{x_1 \mapsto t_1, \ldots x_n \mapsto t_n\}$ by $\{\overrightarrow{x_n \mapsto t_n}\}$, and denote by $\theta_{|\neq x}$ the restriction of θ to the domain $\mathcal{D}om(\theta) \setminus \{x\}$.

C. Embedding the algebraic world

A pseudo-term is *algebraic* if it contains symbols in $\mathcal{C}, \mathcal{D}, \mathcal{X}$ only besides application, and is fully applied. Algebraic terms are identified with terms in $\mathcal{T}(\mathcal{C} \uplus \mathcal{D}, \mathcal{X})$. An arbitrary pseudo-term t can be written as $t = \hat{t}\theta_t$ where $\hat{t}$ is an algebraic term, called *cap* of t, and θ_t is a substitution. We require that all variables in the cap are fresh. The cap $\hat{t}$ is *trivial* if it is a variable in $\mathcal{X}$. A pseudo-term is an *alien* if it has only trivial caps. A cap is *maximal* if $\theta_t(x)$ is an alien for every x in its domain.

Definition II-C.1. The relation $\leftrightarrow^*_{\mathcal{T}}$ is extended to pseudo-terms, by induction, as follows:

(1) $\hat{u}\theta_u \leftrightarrow^*_{\mathcal{T}} v = \hat{v}\theta_v$ iff
 either $\hat{u}$ or $\hat{v}$ is not a variable,
 $\hat{u} \leftrightarrow^*_{\mathcal{T}} \hat{v}$ and $\theta_u \leftrightarrow^*_{\mathcal{T}} \theta_v$,

(2) $\lambda[x:T].\,t \leftrightarrow^*_{\mathcal{T}} v = \lambda[x:W].\,w$ iff
 $T \leftrightarrow^*_{\mathcal{T}} W$ and $t \leftrightarrow^*_{\mathcal{T}} w$,

(3) $\forall(x:T_1).\,T_2 \leftrightarrow^*_{\mathcal{T}} \forall(x:W_1).\,W_2$ iff
 $T_1 \leftrightarrow^*_{\mathcal{T}} W_1$ and $T_2 \leftrightarrow^*_{\mathcal{T}} W_2$,

(4) $\mathrm{ELIM}_o(Q_1, \vec{f_1}, t_1) \leftrightarrow^*_{\mathcal{T}} \mathrm{ELIM}_o(Q_2, \vec{f_2}, t_2)$ iff
 $Q_1 \leftrightarrow^*_{\mathcal{T}} Q_2$, $\vec{f_1} \leftrightarrow^*_{\mathcal{T}} \vec{f_2}$ and $t_1 \leftrightarrow^*_{\mathcal{T}} t_2$,

(5) $M\,N \leftrightarrow^*_{\mathcal{T}} M'\,N'$ iff
 $M \leftrightarrow^*_{\mathcal{T}} M'$ and $N \leftrightarrow^*_{\mathcal{T}} N'$,

(6) otherwise, $u \leftrightarrow^*_{\mathcal{T}} v$ iff $u = v$

where $\vec{u} \leftrightarrow^*_{\mathcal{T}} \vec{v}$ is the component-wise extension of $\leftrightarrow^*_{\mathcal{T}}$, and $\theta_u \leftrightarrow^*_{\mathcal{T}} \theta_v$ iff $\forall x \in \mathcal{D}om(\theta_u) \cap \mathcal{D}om(\theta_v), \theta_u(x) \leftrightarrow^*_{\mathcal{T}} \theta_v(x)$.

By induction on the size of terms, we easily get:

Lemma II-C.2. $\leftrightarrow^*_{\mathcal{T}}$ *is a congruence on pseudo terms.*

We now lift the first-order properties to pseudo-terms:

Lemma II-C.3. *Let* $t = \hat{t}\theta_t$ *such that* $\hat{t} \in \mathcal{T}(\mathcal{C} \uplus \mathcal{D}, \mathcal{X}) \setminus \mathcal{T}(\mathcal{D}, \mathcal{X})$. *Then, there exists* $s = \hat{s}\theta_s$ *such that,* $\hat{s}_\Lambda \in \mathcal{C}$, $s \leftrightarrow^*_{\mathcal{T}} t$ *and* $\mathrm{FV}(s) \subseteq \mathrm{FV}(t)$.

Lemma II-C.4. *Let* u, v *s.t.* $u \leftrightarrow^*_{\mathcal{T}} v$ *and* u_Λ, $v_\Lambda \in \mathcal{C}$. *Then* $u_\Lambda = v_\Lambda$ *and* $\overline{u} \leftrightarrow^*_{\mathcal{T}} \overline{v}$.

Definition II-C.5. We inductively define the notion of *pseudo-terms simplification* as follows:

(1) $\hat{v}\theta_v$ simplifies $\hat{u}\theta_u$ iff
 $\hat{u}$ is not a variable, $\hat{v}$ simplifies $\hat{u}$, and θ_v simplifies θ_u
 (that is, $\forall x \in \mathcal{D}om(\theta_v)$, $\theta_v(x)$ simplifies $\theta_u(x)$)

(2) $\lambda[x:U'].\,t'$ simplifies $\lambda[x:U].\,t$ iff
 U' simplifies U and t' simplifies t,

(3) $\forall(x:U').\,V'$ simplifies $\forall(x:U).\,V$ iff
 U' simplifies U and V' simplifies V,

(4) $M'\,N'$ simplifies $M\,N$ iff
 M' simplifies M and N' simplifies N,

(5) $\mathrm{ELIM}_o(Q', \vec{f'}, t')$ simplifies $\mathrm{ELIM}_o(Q, \vec{f}, t)$ iff
 Q', t' and $\vec{f'}$ resp. simplify Q, t and $\vec{f}$,

(6) otherwise, v simplifies u iff $u = v$

Note that $v \leftrightarrow^*_{\mathcal{T}} u$ as soon as v simplifies u. We now show that a common simplified term always exists for any pair of $\leftrightarrow^*_{\mathcal{T}}$-convertible pseudo-terms:

Lemma II-C.6. *Let* u, v *be pseudo-terms such that* $u \leftrightarrow^*_{\mathcal{T}} v$. *Then, there exists a term* w *that* simplifies u *and* v.

Proof: By induction on $|u| + |v|$, cases according to Definition II-C.1 and using Lemma II-A.1. ∎

Well-foundedness of subterm is compatible with $\leftrightarrow^*_{\mathcal{T}}$:

Definition II-C.7. We define the *constructor size* $n_{\mathcal{C}}(t)$ of a term t to be the maximum of the following set:

$$\left\{ |v| \;\middle|\; \begin{array}{l} v \in \mathcal{T}(\mathcal{C}, \mathcal{X}) \\ t \leftrightarrow^*_{\mathcal{T}} v\theta \text{ for some substitution } \theta \end{array} \right\}$$

Lemma II-C.8. *Let* t *be a term. Then, there exists a natural number* n *such that for any pseudo-term* u, *if (i)* $t \leftrightarrow^*_{\mathcal{T}} u$ *and (ii)* $u = v\theta$ *with* $v \in \mathcal{T}(\mathcal{C}, \mathcal{X})$, *then* $|u| < n$.

Proof: By induction on the size of t, use of Definition II-C.1 and Lemma II-A.2. ∎

Corollary II-C.9. *Let* $t \in \mathcal{T}(\mathcal{C} \uplus \mathcal{D}, \mathcal{X})$ *such that (i)* $t \leftrightarrow^*_{\mathcal{T}} u$ *and (ii)* $u = c(u_1, \ldots, u_n)$. *Then* $n_{\mathcal{C}}(t) > n_{\mathcal{C}}(u_i)$ *for any* $i \in [1..n]$.

D. Reduction

There are three kinds of reduction in our calculus: β-reduction $(\rightarrow_\beta)$, ι-reduction $(\rightarrow_\iota)$, and $\iota_{\mathcal{T}}$-reduction $(\rightarrow_{\iota_{\mathcal{T}}})$, whose union is denoted by $\rightarrow_{\beta\iota_{\mathcal{T}}}$.

[β reduction] $\quad (\lambda[x:T].\,u)t \rightarrow_\beta u\{x \mapsto t\}$

[ι reduction] $\quad \rightarrow_\iota$ is the same as in CIC. It is reserved for *big* inductive types - that is, inductive types having a

constructor taking a functional argument - which cannot be declared as a first-order algebra equipped with a decidable theory $\mathcal{T}$.

[$\iota_{\mathcal{T}}$ reduction] $\rightarrow_{\iota_{\mathcal{T}}}$ generalizes pure ι-reduction in the sense that the latter is a particular case of the former for (small) inductive types whose constructors are first-order. For our example of Presburger arithmetic, we have:

$$\text{ELIM}_n(Q, f_{\mathbf{0}},\ f_{\mathbf{S}}, t) \rightarrow_{\iota_{\mathcal{T}}} \begin{cases} f_{\mathbf{0}} & (1) \\ f_{\mathbf{S}}\ u\ \text{ELIM}_n(Q, f_{\mathbf{0}},\ f_{\mathbf{S}}, u) & (2) \end{cases}$$

with

- $t \leftrightarrow^*_{\mathcal{T}} \mathbf{0}$ for case (1), and
- $\exists u,\ t \leftrightarrow^*_{\mathcal{T}} \mathbf{S}\ u$ and $\mathbf{S}\ u$ simplifies t for case (2).

With the traditional elimination rule, t is identical to $\mathbf{S}\ u$, and therefore, u is typable when t is typable, and has a smaller size. Our requirement that $\mathbf{S}\ u$ simplifies t is essential: it ensures that u has a strictly smaller constructor size, and that it is typable when u is, as we shall see later. Furthermore, Lemma II-C.6 ensures that such a term exists as soon as t is $\mathcal{T}$-equivalent to a term headed by $\mathbf{S}$.

Using elimination, we can now define multiplication and more by induction. Of course, these definitions *cannot* be used in the elimination schema, which relies on the sole theory $\mathcal{T}$. Using them would require to have them in the first order theory $\mathcal{T}$ itself.

In the sequel, we assume for simplicity of notations that all inductive types are given as algebraic types equipped with a decidable equational theory, the trivial one for traditional (small) inductive types. Accommodating big inductive types in the traditional way is no challenge: the whole meta-theory including strong-normalization is the same, provided all inductive types are at the propositional level. This would *not* be the case if inductive types were defined at the predicative level, because we could not use the proof-irrelevant interpretation we use here for proving strong normalization.

Notations we use the notations: $\rightarrow$ for $\rightarrow_{\beta\iota_{\mathcal{T}}}$, $\leftarrow$ for the inverse of $\rightarrow$, $\rightarrow^*$ for its reflexive, transitive closure, $\leftrightarrow^*$ for its symmetric, reflexive, transitive closure, and $\simeq$ for the *conversion* relation defined as $(\leftrightarrow^*_{\mathcal{T}} \cup \leftrightarrow^*)^*$.

Reductions are extended to substitutions as expected. For instance, $\theta \rightarrow_{\beta\iota_{\mathcal{T}}} \theta'$ iff $\theta(x) \rightarrow_{\beta\iota_{\mathcal{T}}} \theta'(x)$ for some variable $x \in \mathcal{D}om(\theta) = \mathcal{D}om(\theta')$ and $\theta_{|\neq x} = \theta'_{|\neq x}$. Given t, t', θ, θ' such that $t \rightarrow^*_{\beta\iota_{\mathcal{T}}} t'$ and $\theta \rightarrow^*_{\beta\iota_{\mathcal{T}}} \theta'$, it is clear that $t\theta \rightarrow^*_{\beta\iota_{\mathcal{T}}} t'\theta'$.

E. Typing

An environment Γ is a sequence of pairs made of a (fresh) variable and a pseudo-term. We denote by $\mathcal{D}om(\Gamma) = \{x_i \mid x_i : T_i \in \Gamma\}$ the *domain* of the environment Γ. We often consider environments as substitutions, writing $x\Gamma = T$ if $x : T \in \Gamma$. An environment Δ *contains* an environment Γ, written $\Gamma \subseteq \Delta$, if all pairs in Γ appear in Δ in the same order. An environment Γ' *simplifies* an environment Γ if T'

$$[\text{VAR}]\ \frac{\Gamma \vdash T : \mathbf{Type}_j}{\Gamma, x : T \vdash x : T}\ x \notin \mathcal{D}om(\Gamma)$$

$$[\text{WEAK}]\ \frac{\Gamma \vdash t : T, \quad \Gamma \vdash V : \mathbf{Type}_j}{\Gamma, x : V \vdash t : T}\ x \notin \mathcal{D}om(\Gamma)$$

$$[\text{LAM}]\ \frac{\Gamma, (x : U) \vdash t : V \qquad \Gamma \vdash \forall(x : U).\ V : \mathbf{Type}_j}{\Gamma \vdash \lambda[x : U].\ t : \forall(x : U).\ V}$$

$$[\text{APP}]\ \frac{\Gamma \vdash u : \forall(x : U).\ V \qquad \Gamma \vdash v : U}{\Gamma \vdash u\ v : V[x \mapsto v]}$$

$$[\text{CONV}]\ \frac{\Gamma \vdash t : U \qquad \Gamma \vdash U' : \mathbf{Type}_j}{\Gamma \vdash t : U'}\ U \simeq U'$$

$$[\text{HIERARCHY}_{j \geq 0}]\ \frac{}{\vdash \mathbf{Type}_j : \mathbf{Type}_{j+1}}$$

$$[\text{CUM}_{j \geq 0}]\ \frac{\Gamma \vdash T : \mathbf{Type}_j}{\Gamma \vdash T : \mathbf{Type}_{j+1}}$$

$$[\text{IMPRED}]\ \frac{\Gamma \vdash U : \mathbf{Type}_j \qquad \Gamma, x : U \vdash V : \mathbf{Type}_0}{\Gamma \vdash \forall(x : U).\ V : \mathbf{Type}_0}$$

$$[\text{PRED}]\ \frac{\Gamma \vdash U : \mathbf{Type}_i \qquad \Gamma, x : U \vdash V : \mathbf{Type}_{j \neq 0}}{\Gamma \vdash \forall(x : U).\ V : Type_{max(i,j)}}$$

$$[\text{nat}]\ \frac{}{\vdash \mathbf{nat} : \mathbf{Type}_0}$$

$$[\mathbf{0}]\ \frac{}{\vdash \mathbf{0} : \mathbf{nat}} \qquad [\mathbf{S}]\ \frac{}{\vdash \mathbf{S} : \mathbf{nat} \rightarrow \mathbf{nat}}$$

$$[+]\ \frac{}{\vdash +: \mathbf{nat} \rightarrow \mathbf{nat} \rightarrow \mathbf{nat}}$$

$$[\text{ELIM}]\ \frac{\begin{array}{c} \Gamma \vdash t : \mathbf{nat} \\ \Gamma \vdash P : \forall(x : \mathbf{nat}).\ \mathbf{Type}_0 \qquad \Gamma \vdash f_{\mathbf{0}} : P\ \mathbf{0} \\ \Gamma \vdash f_{\mathbf{S}} : \forall(x : \mathbf{nat}).\ (P\ x \rightarrow P\ (\mathbf{S}\ x)) \end{array}}{\Gamma \vdash \text{ELIM}_n(P, f_{\mathbf{0}}, f_{\mathbf{S}}, t) : P\ t}$$

Fig. 1: CoQMTU typing rules

simplifies T for some pairs $x : T \in \Gamma$ and $x : T' \in \Gamma'$. An environment Γ *reduces* to an environment Γ if T reduces to T' for some pairs $x : T \in \Gamma$ and $x : T' \in \Gamma'$. Two environments Γ, Γ' are *compatible* modulo a relation $\mathcal{R}$ on pseudo-terms if for any $x \in \mathcal{D}om(\Gamma) \cap \mathcal{D}om(\Gamma')$, $(x\Gamma)\ \mathcal{R}\ (x\Gamma')$.

Our typing rules given at Figure 1 come in three parts: for CC, for the universes, and for the first-order symbols.

Definition II-E.1. Given $\Gamma \vdash M : N$ for some environment Γ and terms M, N, we say that:

- Γ is a *valid* environment,
- M is a Γ-*term* (*well-typed* term under Γ),
- M is a Γ-*type* (*well-typed* type under Γ), if N is an

universe,
- a Γ-type M is a Γ-*proposition* if $\Gamma \vdash N : \mathbf{Prop}$ for some $N \simeq M$ and *non-propositional* otherwise,
- M is a Γ-*proof* if N is a Γ-proposition.

III. PROPERTIES

In this section, we consider the case of the theory $\mathcal{T}$ defined as the fragment of Presburger arithmetic with basic type $\mathbf{nat}$, constructors $\{\mathbf{0}, \mathbf{S}\}$, and defined symbols $\{+\}$. This assumption does not imply any loss of generality, but eases the reading. Most proofs are omitted.

A. Church-Rosser property

To prove that $\rightarrow$ is Church-Rosser, we proceed in 3 steps:
Following Hindley's technique, we first define a parallel reduction (Figure 2) and show the relation between this parallel reduction and our reduction:

Lemma III-A.1. *For any u, v s.t. $u \rightarrow v$, we have $u \Rightarrow v$.*

Proof: Induction on the structure of term u. ∎

Lemma III-A.2. *For any u, v s.t. $u \Rightarrow v$, we have $u \rightarrow^* v$.*

Proof: Straightforward induction on the derivation $u \Rightarrow v$. ∎

$$[\textsc{Refl}] \; \frac{}{t \Rightarrow t} \qquad [\textsc{App}] \; \frac{u \Rightarrow u', \; v \Rightarrow v'}{u \; v \Rightarrow u' \; v'}$$

$$[\textsc{Lam}] \; \frac{U \Rightarrow U', \; t \Rightarrow t'}{\lambda[x:U].t \Rightarrow \lambda[x:U'].t'}$$

$$[\textsc{Prod}] \; \frac{U \Rightarrow U', \; T \Rightarrow T'}{\forall(x:U).T \Rightarrow \forall(x:U').T'}$$

$$[\textsc{Elim}] \; \frac{P \Rightarrow P', f_{\mathbf{0}} \Rightarrow f_{\mathbf{0}}', f_{\mathbf{S}} \Rightarrow f_{\mathbf{S}}', t \Rightarrow t'}{\text{ELIM}_n(P, f_{\mathbf{0}}, f_{\mathbf{S}}, t) \Rightarrow \text{ELIM}_n(P', f_{\mathbf{0}}', f_{\mathbf{S}}', t')}$$

$$[\beta] \; \frac{t \Rightarrow t', \; v \Rightarrow v'}{(\lambda[x:U].t)v \Rightarrow t'\{x \mapsto v'\}}$$

$$[\iota_{\mathbf{0}}] \; \frac{t \Rightarrow t', t' \leftrightarrow_{\mathcal{T}}^* \mathbf{0}, f_{\mathbf{0}} \Rightarrow f_{\mathbf{0}}'}{\text{ELIM}_n(P, f_{\mathbf{0}}, f_{\mathbf{S}}, t) \Rightarrow f_{\mathbf{0}}'}$$

$$[\iota_{\mathbf{S}}] \; \frac{\begin{array}{c} P \Rightarrow P', t \Rightarrow t', f_{\mathbf{0}} \Rightarrow f_{\mathbf{0}}', f_{\mathbf{S}} \Rightarrow f_{\mathbf{S}}' \\ \mathbf{S} \; u \text{ simplifies } t' \end{array}}{\text{ELIM}_n(P, f_{\mathbf{0}}, f_{\mathbf{S}}, t) \Rightarrow f_{\mathbf{S}}' \; u \; \text{ELIM}_n(P', f_{\mathbf{0}}', f_{\mathbf{S}}', u)}$$

Fig. 2: Parallel Reduction

Then, following Jouannaud and Kirchner [12], we show the coherence and the confluence of $\Rightarrow$.

Lemma III-A.3. *For any pseudo-terms u, v, v' s.t. $u \leftrightarrow_{\mathcal{T}}^* v$ and $u \Rightarrow u'$, there exists v' such that $v \Rightarrow v'$ and $u' \leftrightarrow_{\mathcal{T}}^* v'$.*

Lemma III-A.4. *For any pseudo-terms t, t_1, t_2 s.t. $t \Rightarrow t_1$, $t \Rightarrow t_2$, there exist t_1', t_2' such that $t_1 \Rightarrow t_1'$, $t_2 \Rightarrow t_2'$ and $t_1' \leftrightarrow_{\mathcal{T}}^* t_2'$.*

Finally, using usual techniques, we obtain:

Theorem III-A.5 (Church-Rosser). *For all t, u such that $t \simeq u$, there exist t', u' such that $t \rightarrow^* t'$, $u \rightarrow^* u'$ and $t' \leftrightarrow_{\mathcal{T}}^* u'$.*

As a consequence of the Church-Rosser property, we obtain:

Corollary III-A.6. *(Universe compatibility)*
If $\mathbf{Type}_i \simeq \mathbf{Type}_j$, then $i = j$.

Corollary III-A.7. *(Product compatibility)*
If $\forall(x:U).T \simeq \forall(x:U').T'$, then $U \simeq U'$ and $T \simeq T'$.

B. Subject Reduction

Subject reduction, as usual, requires proving properties of environments, such as weakening and strengthening, and of conversion, such as stability (under substitution), before proving the usual inversion lemma:

Lemma III-B.1. *(Inversion)*
(1) Let $\Gamma \vdash \lambda[x:W].t : \forall(x:U).V$. There exists T s.t. $\Gamma, (x:W) \vdash t : V$ and $\forall(x:W).T \simeq \forall(x:U).V$,
(2) Let $\Gamma \vdash \mathbf{0} : T$. Then $T \simeq \mathbf{nat}$,
(3) Let $\Gamma \vdash \mathbf{S} \; u : T$. Then $\Gamma \vdash u : \mathbf{nat}$ and $T \simeq \mathbf{nat}$,
(4) Let $\Gamma \vdash + \; u \; v : T$. Then $\Gamma \vdash u, v : \mathbf{nat}$, $T \simeq \mathbf{nat}$,
(5) Let $\Gamma \vdash \text{ELIM}_n(P, f_{\mathbf{0}}, f_{\mathbf{S}}, t) : V$. Then,
$\Gamma \vdash t : \mathbf{nat}$, $\Gamma \vdash P : \forall(x:\mathbf{nat}).\mathbf{Prop}$, $\Gamma \vdash f_{\mathbf{0}} : P \; \mathbf{0}$,
$\Gamma \vdash f_{\mathbf{S}} : \forall(x:\mathbf{nat}).(P \; x \rightarrow P \; (\mathbf{S} \; x))$ and $P \; t \simeq V$.

The last step before subject-reduction is the proof of a very specific form of $\mathcal{T}$-irrelevance:

Lemma III-B.2 ($\mathcal{T}$-irrelevance). *Assume that $\Gamma \vdash t : T$ and t' simplifies t. Then $\Gamma \vdash t' : T$.*

Proof: The proof is by induction on $\Gamma \vdash t : T$, by case on the last rule used. Most cases are straightforward, except the [APP] one, when $t = \hat{t}\theta_t$ is a non-alien term, which needs some properties of the simplification relation. ∎

Now, we finally can show:

Theorem III-B.3. *(Subject reduction) Let $\Gamma \vdash t : T$ and $t \rightarrow t'$. Then $\Gamma \vdash t' : T$.*

As usual, the proof is by induction on $\Gamma \vdash t : T$, proving simultaneously the two following properties:
(1) if Γ reduces to Γ', then $\Gamma' \vdash t : T$, and
(2) if t reduces to t', then $\Gamma \vdash t' : T$.

Most cases are routine, except the [APP] and [ELIM] ones (which require the use of the $\mathcal{T}$-irrelevance) for (1), and the [VAR] one for (2).

C. Universal environment $\mathcal{E}$

In the calculus of constructions, types belong to the impredicative universe $Type_0$. With cumulativity, a type in COQMTU may belong to several universes. Furthermore, convertible types may belong to different sets of universes.

Since convertible types may not have the same set of free variables, carrying typing judgments across conversions raises a technical difficulty solved first thanks to Luo's universal typing environment:

Definition III-C.1. An infinite sequence of the form $\mathcal{E} = \{e_i : E_i\}_{i \geq 0}$ such that $e_k = e_l$ implies $k = l$ is a *universal environment* if for any pseudo-term A and $\mathbf{Type}_p$ such that $\mathcal{E}^j \vdash A : \mathbf{Type}_p$ and for any natural number j, there exists $k > j$ such that $E_k = A$.

Being defined by a closure property, universal environments exist and can be constructed by using a diagonal enumeration argument. Given a universal environment $\mathcal{E}$, we denote by $\mathcal{E}^i$ the environment $\{e_j : E_j\}_{j \leq i}$. A is an $\mathcal{E}$-type if $\mathcal{E}^i \vdash A : T$ for some i, written $\mathcal{E} \vdash A : T$.

D. Level of $\mathcal{E}$-types

Definition III-D.1. Let A, B be $\mathcal{E}$-types. A is smaller than B in the *cumulativity* relation, written $A \preceq B$, iff one of the following properties holds:

(1) $A \simeq B$, and A, B are not equivalent to a universe, nor to a product,
(2) $A \simeq \mathbf{Type}_i$, $B \simeq \mathbf{Type}_j$, and $i \leq j$,
(3) $A \simeq \forall(x : A_1). \, A_2$, $B \simeq \forall(x : B_1). \, B_2$, $A_1 \simeq B_1$ and $A_2 \preceq B_2$.

This definition by induction on the structure of types makes sense because of Corollaries III-A.7 and III-A.6. We get:

Lemma III-D.2. *Cumulativity is a quasi-order on $\mathcal{E}$-types with conversion as its associated equivalence relation, and whose strict part $\succ$ is well-founded.*

Being well-founded, cumulativity allows us choosing a *minimum type* - up to conversion - among all possible types of a typable term t in an environment Γ, written $T_\Gamma(t)$.
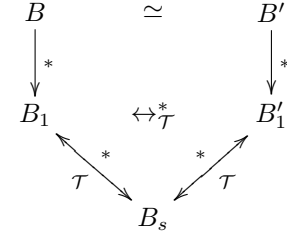
We now move to the minimum universe of a type.

Definition III-D.3. The level $\mathcal{L}(A)$ of an $\mathcal{E}$-type A is the minimum natural number j such that $\mathcal{E} \vdash B : \mathbf{Type}_j$ for some $\mathcal{E}$-type B such that $A \simeq B$.

Lemma III-D.4. *Let A, B be $\mathcal{E}$-types.*

(1) If $A \preceq B$, then $\mathcal{L}(A) \leq \mathcal{L}(B)$.
(2) In particular, $\mathcal{L}(A) = \mathcal{L}(B)$ if $A \simeq B$.

Lemma III-D.5. *Let N be a term such that $\mathcal{E}^k \vdash N : E_{k+1}$ and B be an $\mathcal{E}^{k+1}$-type. Then, $\mathcal{L}(B\{e_{k+1} \mapsto N\}) \leq \mathcal{L}(B)$.*

Proof: Suppose $p = \mathcal{L}(B) < \mathcal{L}(B\{e_{k+1} \mapsto N\}) = j$. By definition, there exists $B' \simeq B$ such that $\mathcal{E} \vdash B' : \mathbf{Type}_p$. By Theorem III-A.5 and Lemma II-C.6, we have the following diagram:

$$
\begin{array}{ccc}
B & \simeq & B' \\
\downarrow{\scriptstyle *} & & \downarrow{\scriptstyle *} \\
B_1 & \overset{*}{\leftrightarrow}_{\mathcal{T}} & B_1' \\
\,\,\,\,{}_{\mathcal{T}}\!\searrow^{*} & & {}^{*}\!\swarrow_{\mathcal{T}} \\
& B_s &
\end{array}
$$

By subject reduction twice: $\mathcal{E} \vdash B_1' : \mathbf{Type}_p$ and B_1 is an $\mathcal{E}^{k+1}$ type. By Lemma II-C.6, B_s simplifies B_1' and B_1, hence $\mathcal{E} \vdash B_s : \mathbf{Type}_p$ and B_s is an $\mathcal{E}^{k+1}$ type by $\mathcal{T}$-irrelevance. Thus $\mathcal{E}^{k+1} \vdash B_s : \mathbf{Type}_p$ by weakening. By stability, $\mathcal{E}^k \vdash B_s\{e_{k+1} \mapsto N\} : \mathbf{Type}_p$. By weakening again, $\mathcal{E} \vdash B_s\{e_{k+1} \mapsto N\} : \mathbf{Type}_p$. Since $B_s\{e_{k+1} \mapsto N\} \simeq B\{e_{k+1} \mapsto N\}$ by Lemma II-C.2, by definition $\mathcal{L}(B\{e_{k+1} \mapsto N\}) = j \leq p$, a contradiction. ∎

As an easy corollary, we get:

Lemma III-D.6. *Let $(M \, N)$ be an $\mathcal{E}$-term. Then:*

$$\mathcal{L}(T_\mathcal{E}(M \, N)) \leq \mathcal{L}(T_\mathcal{E}(M)).$$

Lemma III-D.7. *Let $M \simeq N$ be $\mathcal{E}$-types, then*

(1) $T_\mathcal{E}(M)$ and $T_\mathcal{E}(N)$ are convertible or have the same form.
(2) Especially, if $T_\mathcal{E}(M)$ is an $\mathcal{E}$-proposition, then $T_\mathcal{E}(N)$ is also an $\mathcal{E}$-proposition.

IV. STRONG NORMALIZATION

Calculi with universes were introduced by Luo [3], who showed strong normalization of ECC by clearly separating predicative reductions, defined below, from those which are not.

Definition IV-.8. The *main term* of a redex R is

(1) itself if R is a $\iota_\mathcal{T}$-redex, or
(2) the term M if R is the β-redex $M \, N$.

If a redex R is an $\mathcal{E}$-term, we say that R is *predicative* iff $\mathcal{L}(T_\mathcal{E}(M)) > 0$, where M is the main term of R.

All $\iota_\mathcal{T}$-redexes are *impredicative*, while β-redexes can be predicative or impredicative. Reducing predicative β-redexes yields *predicative reduction*.

Luo showed that ECC predicative β-reduction enjoys normal-forms, a crucial property ensuring that impredicativity is restricted to the propositional level. Furthermore, normal forms have a simple structure which can be used to define interpretations "a la Girard" thanks to a (technically extremely complex) well-founded order on types.

We follow the same proof, with modifications detailed below.

a) Step 1: We show that predicative reduction enjoys normal forms.

Luo's proof uses the fact that forbidding strong elimination principles for types in the impredicative layer makes proof-irrelevance admissible: the whole construction (being the order on types or the interpretation of types as saturated sets) never discriminates proofs.

Since **nat** only allows weak eliminations, we can extend this irrelevance to first-order terms. This is made by putting **nat** in $\textbf{Type}_0$: proof-irrelevance subsumes the identification of first-order terms by $\leftrightarrow^*_{\mathcal{T}}$.

We have to adapt the notion of base terms (corresponding to types that are neither products nor universes) to cope with the new term constructors:

Definition IV-.9. *Base terms* are: the variables, the type constant **nat**, applications of the form $(M\ N)$ with M a base term.

Sometimes, we also need a notion of *object-level base terms* which includes terms of the form $\text{ELIM}_n(Q, f_{\mathbf{0}}, f_{\mathbf{S}}, t)$ where t does not simplify to terms of the form $\mathbf{0}$ or $\mathbf{S}\ u$. Other definitions such as the degree and quasi-normal forms (which we also call predicative quasi-normal forms) are unchanged, bearing in mind that our notion of level is shifted by 1 (the impredicative level is 0 here while it was -1 in Luo's proof). The proof of properties of the degree are adapted easily. The case of $\mathcal{T}$ constants is similar to the variable case, and we take care of the ELIM case simply by applying the induction hypotheses (just like in the abstraction case). Also, redexes of maximum level (greater than 0) cannot be $\iota_{\mathcal{T}}$-redexes, but only redexes of the predicative reduction.

The quasi-normalization lemma can now be proved. Its main corollary is a characterization of types in predicative quasi-normal form:

Theorem IV-.10. *Every $\mathcal{E}$-type can be reduced to some* predicative normal form, *which is either: a universe, a product, or a base term.*

Proof: Again, proving that an $\mathcal{E}$-type t has a predicative quasi-normal form t' follows Luo's proof. The extra cases are either immediate or follow easily from the induction hypothesis (ELIM case).

Then, we show by structural induction that t' has the appropriate form. By subject reduction, t' is an $\mathcal{E}$-type. We treat the case where $t' = (M\ N)$. If $M \in \mathcal{X}$, then t' is a base term. If $M = \lambda[x : U].\, v$, then $T_{\mathcal{E}}(M) \simeq \forall(x : U).\, V$, by the assumption that t' is in predicative quasi-normal form, hence $\mathcal{L}(T_{\mathcal{E}}(M)) = \mathcal{L}(\forall(x : U).\, V) = 0$. By Rule [IMPRED], $\mathcal{L}(V) = 0$. By rule [APP], $\mathcal{E} \vdash t' : V\{x \mapsto N\}$. By Lemma III-D.5, $\mathcal{L}(V\{x \mapsto N\}) = 0$, hence t' is an $\mathcal{E}$-proof, contradicting our assumption. If M is an application, we conclude by application of the induction hypothesis and from the definition of base terms. ∎

b) **Step 2:** Define a complexity measure for types.

Definition IV-.11. Given an $\mathcal{E}$-type A with quasi-normal form B, its *degree* $\mathcal{D}(A)$ is defined by case on B:

(1) if B is $\textbf{Type}_j$ or a base term, then $\mathcal{D}(A) = 1$, and
(2) if $B = \forall(x : C).\, D$, then $\mathcal{D}(A) = max\{\mathcal{D}(C), \mathcal{D}(D)\}+1$.

The *complexity* $\beta(A)$ of an $\mathcal{E}$-type A is defined as the pair $\beta(A)$ made of its *level* $\mathcal{L}(A)$ and its *degree* $\mathcal{D}(A)$. Complexities are compared lexicographically in the order

$\gg = (\geq, >)_{lex}$. While these notions are inspired from [3], the presentation here is much simpler.

Degree and level of an $\mathcal{E}$-type enjoy very similar properties:

Lemma IV-.12. *if A, B are $\mathcal{E}$-types, then*
(1) $\mathcal{D}(A) = \mathcal{D}(B)$ *if $A \simeq B$,*
(2) $\mathcal{D}(A) \leq \mathcal{D}(B)$ *if $A \preceq B$.*

Lemma IV-.13. *Assume that $\mathcal{E}^i \vdash N : E_{k+1}$ and that B is a $\mathcal{E}^{k+1}$-type. If $\mathcal{L}(E_{k+1}) \leq \mathcal{L}(B)$ and $\mathcal{L}(B) = \mathcal{L}(B\{e_{k+1} \mapsto N\})$, then $\mathcal{D}(B\{e_{k+1} \mapsto N\}) \leq \mathcal{D}(B)$.*

We can lift them to complexities:

Lemma IV-.14. *Let A and B be $\mathcal{E}$-types. Then*
(1) $\beta(A) \leq_{lex} \beta(B)$ *if $A \preceq B$, and*
(2) $\beta(A) = \beta(B)$ *if $A \simeq B$.*

Lemma IV-.15. *Let A be a $\mathcal{E}$-type with $\forall(x : A_1).\, A_2$ as predicative normal form. Then*
(1) $\beta(A_i) <_{lex} \beta(A)$ *for $i = 1, 2$, and*
(2) $\beta(A_2\{x \mapsto N\}) <_{lex} \beta(A)$, $\forall N$ *s.t. $\mathcal{E} \vdash N : A_1$.*

c) **Step 3:** Strong normalization of COQMTU.

Strong normalization of COQMTU is proved by Tait and Girard's method. Adapting [3], the interpretation of types is defined by induction on their complexity, which we proceed to do now. Using Rule [CONV], we can assume without loss of generality that predicative types are minimum.

A. Saturated sets

Definition IV-A.1. We define the *key redex* of an application to be the leftmost outermost redex of a term:

(1) If M is a redex, then M is the key redex of itself.
(2) Otherwise, the key redex of $(M\ N)$ is the key redex, if any, of M.

Contracting the key redex of M yields the term $red_k(M)$.

Definition IV-A.2. Given an $\mathcal{E}$-type A, let $\mathcal{SN}(A)$ be the set of all strongly normalizable terms M such that $\mathcal{E} \vdash M : A$. A set S is said to be *A-saturated* if and only if

(S1) $S \subseteq \mathcal{SN}(A)$.
(S2) If $M \in \mathcal{SN}(A)$ is a base term, then $M \in S$,
(S3) If $M \in \mathcal{SN}(A)$ and $red_k(M) \in S$, then $M \in S$.

The set of all A-saturated sets is denoted by $Sat(A)$.

B. Valuations

Let π be an arbitrary new value that will serve to interpret terms of a specific type.

Definition IV-B.1. The set of possible values $V(M)$ of an $\mathcal{E}$-term M is defined by induction on its minimum type $T_{\mathcal{E}}(M)$, assumed to be in quasi-normal form:

(1) If $T_{\mathcal{E}}(M)$ is an universe, then $V(M) = Sat(M)$,
(2) If $T_{\mathcal{E}}(M)$ is an $\mathcal{E}$-proposition or a base term, then $V(M) = \{\pi\}$
(3) If $T_{\mathcal{E}}(M) = \forall(x : A_1).\, A_2$, then $V(M)$ is the set of functions f satisfying the following three properties:

(a) $\mathcal{D}om(f) = \{(N, v) \mid \mathcal{E} \vdash N : A_1, v \in V(N)\}$
(b) $f(N, v) \in V(MN)$ for $(N, v) \in \mathcal{D}om(f)$
(c) $f(N, v) = f(N', v)$ for $(N, v), (N', v) \in \mathcal{D}om(f)$ such that $N \simeq N'$

This definition makes sense: predicative normal forms exist and are of the four possible forms above by Theorem IV-.10, recursive calls operate on terms whose types have a strictly smaller complexity by Lemmas IV-.14 and IV-.15, and condition (c) is easy to ensure.

The following two properties are proved by induction on the complexity of predicative types in normal form:

Lemma IV-B.2. *Let M, N be $\mathcal{E}$-types such that $M \simeq N$. Then $V(M) = V(N)$.*

Lemma IV-B.3. *Let M be a $\mathcal{E}$-term. Then $V(M) \neq \emptyset$.*

Definition IV-B.4. Given k, an $\mathcal{E}$-assignment is a substitution ϕ of domain $\mathcal{D}om(\mathcal{E}^k)$ such that $\mathcal{E} \vdash \phi(e_i) : \phi(E_i)$.

An $\mathcal{E}$-*valuation* is a pair $\rho = (\phi, val)$ such that ϕ is an $\mathcal{E}$-assignment and val is a function from $\mathcal{D}om(\phi)$ to the set of terms such that $val(e_i) \in V(\phi(e_i))$.

Let A be an $\mathcal{E}^k$-type. By definition of $\mathcal{E}$, there exists some $m \geq k$ such that $e_m : A \in \mathcal{E}$.

C. Interpretation of terms

We can now build a model of terms "a la Girard" and go through the usual sequence of properties to show its soundness.

Definition IV-C.1. Let $\rho = (\phi, val)$ be an $\mathcal{E}$-valuation. The evaluation $Eval_\rho(M)$ of an $\mathcal{E}$-term M is defined by induction on its structure as follows:
(1) M is an $\mathcal{E}$-proof. Then $Eval_\rho(M) = \pi$.
(2) M is an universe or **nat**. Then $Eval_\rho(M) = \mathcal{SN}(M)$.
(3) M is a variable. Then $Eval_\rho(M) = val(M)$.
(4) $M = \forall(x : M_1). M_2$. Suppose that $x \notin \mathcal{D}om(\rho)$. Then $Eval_\rho(M)$ is the set of terms F, s.t.
   (a) $\mathcal{E} \vdash F : \phi(M)$,
   (b) $F N \in Eval_{\rho'}(M_2)$ for every $\mathcal{E}$-valuation $\rho' = (\phi', val')$ extending ρ such that $\phi'(x) = N \in Eval_\rho(M_1)$.
(5) $M = \lambda[x : M_1]. M_2$. Suppose that $x \notin \mathcal{D}om(\rho)$. Then $Eval_\rho(M)$ is a function f such that
   (a) $\mathcal{D}om(f) = \{(N, v) \mid \mathcal{E} \vdash N : \phi(M_1), v \in V(N)\}$,
   (b) $f(N, v) = Eval_{\rho'}(M_2)$ for $(N, v) \in \mathcal{D}om(f)$, and ρ' extends ρ such that $\rho'(x) = (N, v)$.
(6) $M = (M_1 \ M_2)$. Then,
   $Eval_\rho(M) = Eval_\rho(M_1)(\phi(M_2), Eval_\rho(M_2))$.

Lemma IV-C.2. *(Well-definedness of interpretations)*
Let $\rho = (\phi, val)$ be an $\mathcal{E}$-valuation and M an $\mathcal{E}$-term.
(1) Assume that $\rho' = (\phi', val')$ is an $\mathcal{E}$-valuation such that $\phi(M) \simeq \phi'(M)$ and $val(x) = val'(x)$ for every $x \in \text{FV}(M)$. Then $Eval_\rho(M) = Eval_{\rho'}(M)$,
(2) $Eval_\rho(M) \in V(\phi(M))$, hence $Eval_\rho(M)$ is a $\phi(M)$-saturated set.

Proof: By mutual induction on the structure of M. ∎

Before the soundness proof, we prove two lemmas to show that evaluation is stable and monotonic.

Lemma IV-C.3. *Let M, N be terms, $\rho = (\phi, val)$ an interpretation, x a variable such that $x \notin \mathcal{D}om(\rho)$, and $\rho' = (\phi', val')$ extending ρ with $\rho'(x) = (\phi(N), Eval_\rho(N))$. Then $Eval_\rho(M\{x \mapsto N\}) = Eval_{\rho'}(M)$.*

Proof: By induction on the structure of M. ∎

Lemma IV-C.4. *Let M, N be $\mathcal{E}$-terms and ρ be an $\mathcal{E}$-valuation.*
(1) Assume that $M \simeq N$. Then $Eval_\rho(M) = Eval_\rho(N)$.
(2) Assume that M, N are $\mathcal{E}$-types such that $M \preceq N$. Then $Eval_\rho(M) \subseteq Eval_\rho(N)$.

Proof: By induction on the structure of M. ∎

The proof of soundness follows:

Lemma IV-C.5. *(Soundness)*
Let $\mathcal{E}^k \vdash M : A$ and $\rho = (\phi, val)$ an $\mathcal{E}$-valuation such that $\phi(e_i) \in Eval_\rho(E_i)$. Then, $\phi(M) \in Eval_\rho(A)$.

D. Strong Normalization and Consistency

Theorem IV-D.1. *(Strong Normalization)*
Let $\Gamma \vdash t : T$. Then t is strongly normalizable.

Proof: By induction on the length of the environment Γ. Assume first that Γ is empty. By Lemmas IV-C.5, IV-C.2 and property of variables, $t = \phi(t) \in Eval_\phi(T) \in V(\phi(A))$, hence t is strongly normalizable. Let now $\Gamma = \Gamma', x : M$. Applying the rule [LAM] and the induction hypothesis yields the conclusion. ∎

Theorem IV-D.2. *(Consistency)* $\nvdash M : \forall(P : \mathbf{Prop}).P$

Proof: By Theorem IV-D.1, there is a normal proof that can be chosen minimum in size. We reason as usual by contradiction. The only delicate case is when $M = \text{ELIM}_n(Q, f_{\mathbf{0}}, f_{\mathbf{S}}, t)$. By inversion, $\vdash t : \mathbf{nat}$. Since the environment is empty, t is ground. Since it is in normal-form, it must be algebraic, hence equivalent to a constructor term and M cannot be in normal form, a contradiction. ∎

We end up with the decidability of type-checking in COQMTU. As usual, if we check $\Gamma \vdash t : T$, we first define an algorithm of type inference and prove its correctness and completeness. This algorithm first checks whether Γ is valid and then gives t the minimum type $T_\Gamma(t)$ under Γ. Then, we prove the decidability of $\simeq$ and $\preceq$ by using Theorem III-A.5 and Theorem IV-D.1. Finally, we check whether $T_\Gamma(t) \preceq T$. Therefore,

Theorem IV-D.3. *Type checking in COQMTU is decidable.*

V. IMPLEMENTATION

A new version of COQ based on this work is available at http://strub.nu/research/coqmt/.

COQMTU allows the user to dynamically load any decision procedure for a first-order theory in the conversion of the system. We exemplify here the use of our system with a development containing dependent words:

```
Inductive dword : nat -> Type :=
| dword0 : dword 0
| dword1 : T -> dword 1
| dwordA :
    forall n p,
      dword n -> dword p -> dword (n + p) .
```

Here, `dword0` is the empty word, `dword1` is the singleton word, and `dwordA` is the concatenation of words. Being dependent, our `dword` type takes the length of word as argument. When writing the `rev` function in the usual way in the COQ system:

```
Fixpoint rev n (xs : dword n) :=
match xs in dword n return dword n with
| dword0 => dword0
| dword1 x => dword1 x
| dwordA n1 n2 xs1 xs2 => dwordA (rev xs2) (rev xs1)
end .
```

the user is confronted to a type error in the third branch: the input word being of length $n_1 + n_2$, the result must be of a length convertible to $n_1 + n_2$. Here, the result of the third branch is of length $n_2 + n_1$ which is not convertible to $n_1 + n_2$ using $\beta\iota$-conversion. One should remark that writing the `rev` function is feasible in COQ. One possibility is the use of a *cast* function:

```
Definition cast :
  forall n1 n2, n1 = n2 -> dword n1 -> dword n2 .
Proof .
  intros n1 n2 E xs; subst n2; exact xs .
Defined .
```

Such a function acts as the identity function, but while taking a word of length $n1$, it returns the same word seen as a word of length $n2$, as long as a proof of $n1 = n2$ is provided. One can then write the `rev` function, in the COQ system, as follows:

```
Fixpoint rev n (xs : dword n) :=
match xs in dword n return dword n with
| dword0 => dword0
| dword1 x => dword1 x
| dwordA n1 n2 xs1 xs2 =>
    cast (addC n2 n1) (dwordA (rev xs2) (rev xs1))
end .
```

From the previous version, only the third branch has been changed: instead of returning the reversed word directly, the `cast` operator - using a proof of $n_1 + n_2 = n_2 + n_1$ - is applied. Although this technique may be used each time one has to cast the length of a word, its has several drawbacks, the worst being the use of Streicher's K axiom (which is provable in the **nat** case) and all its corollaries in order to remove casts in subsequent proofs.

One can check that the first definition of `rev` is accepted without modification in COQMTU. Moreover, proofs of basic properties about dependent lists follow from the ones on non-dependent lists. The full commented example along others can be found in the source release of COQMTU (*/test-suite/dp/**).

VI. CONCLUSION

We have proved the strong normalization, consistency and decidability of typing of COQMTU, a complex type theory mixing together the Calculus of Constructions, small inductive types given via a specification of constructors and defined symbols satisfying non-triviality, freeness of constructors and completeness of defined symbols, and a predicative hierarchy of universes above **Prop**.

This result is a step towards an ambitious program of studying the type theory of the proof assistant COQ. It also enhances the expressivity of COQ as a logical framework since it allows a smooth encoding of Westbrook's CNIC [11] when restricted to weak elimination.

Since we have assumed a weak-elimination principle, our next step is to incorporate strong elimination. The proof of strong normalization must of course be completely different, since strong elimination, unlike weak-elimination, has predicative instances, therefore making the present proof technique obsolete. We have progressed along a completely different proof format written in COQ (with an axiom assessing existence of infinitely many inaccessible cardinals as in [5]).

REFERENCES

[1] B. Werner, "Une théorie des constructions inductives," Ph.D. dissertation, Université de Paris VII, 1994.

[2] B. Barras, "Verification of the interface of a small proof system in coq," in *TYPES*, ser. Lecture Notes in Computer Science, E. Giménez and C. Paulin-Mohring, Eds., vol. 1512. Springer, 1996, pp. 28–45.

[3] Z. Luo, "ECC, an extended calculus of constructions," in *LICS*. IEEE Computer Society, 1989, pp. 386–395.

[4] H. Goguen, "The metatheory of utt," in *TYPES*, ser. Lecture Notes in Computer Science, P. Dybjer, B. Nordström, and J. M. Smith, Eds., vol. 996. Springer, 1994, pp. 60–82.

[5] B. Werner, "Sets in types, types in sets," in *TACS*, ser. Lecture Notes in Computer Science, M. Abadi and T. Ito, Eds., vol. 1281. Springer, 1997, pp. 530–346.

[6] F. Blanqui, J.-P. Jouannaud, and P.-Y. Strub, "Building decision procedures in the calculus of inductive constructions," in *CSL*, ser. Lecture Notes in Computer Science, J. Duparc and T. A. Henzinger, Eds., vol. 4646. Springer, 2007, pp. 328–342.

[7] ——, "From formal proofs to mathematical proofs: A safe, incremental way for building in first-order decision procedures," in *IFIP TCS*, ser. IFIP, G. Ausiello, J. Karhumäki, G. Mauri, and C.-H. L. Ong, Eds., vol. 273. Springer, 2008, pp. 349–365.

[8] P.-Y. Strub, "Coq modulo theory," in *CSL*, ser. Lecture Notes in Computer Science, A. Dawar and H. Veith, Eds., vol. 6247. Springer, 2010, pp. 529–543.

[9] A. Miquel and B. Werner, "The not so simple proof-irrelevant model of cc," in *TYPES*, ser. Lecture Notes in Computer Science, H. Geuvers and F. Wiedijk, Eds., vol. 2646. Springer, 2002, pp. 240–258.

[10] B. Barras and B. Bernardo, "The implicit calculus of constructions as a programming language with dependent types," in *FoSSaCS*, ser. Lecture Notes in Computer Science, R. M. Amadio, Ed., vol. 4962. Springer, 2008, pp. 365–379.

[11] E. Wsetbrook, "A translation to justify higher-order encodings in intensional type theory," Rice University, Tech. Rep., 2010.

[12] J.-P. Jouannaud and H. Kirchner, "Completion of a set of rules modulo a set of equations," *SIAM J. Comput.*, vol. 15, no. 4, pp. 1155–1194, 1986.