Type inference with algebraic universes in the Calculus of Inductive Constructions

Hugo Herbelin

INRIA-Futurs, LIX, F-91128 Palaiseau

Abstract. We describe an algebraic system of universes and a type-checking algorithm for universe constraints in a version of the extended calculus of constructions with inductive types. The use of algebraic universes ensures that the graph of constraints only contains universes already present in the term to type. This algorithm, used in the type-checker of the Coq proof assistant, refines Huet and Harper-Pollack algorithms for typical ambiguity.

The Extended Calculus of Constructions (ECC) [11] is a type theory extending the Calculus of Constructions (CC) [3, 5] with a stratified cumulative hierarchy of type universes, and Σ-types. We call CC_ω the extension of CC with the hierarchy of ECC but without Σ-types.

CC_ω can be seen as a Pure Type System (PTS) [1, 13] extended with subtyping. The signature of CC_ω as a PTS contains the impredicative sort of propositions `Prop` and a stratified hierarchy of predicative types $\texttt{Type}_i$, for $i \geq 1$. The sort of propositions itself is in $\texttt{Type}_1$ and the stratification is expressed by the axioms $\texttt{Type}_i : \texttt{Type}_{i+1}$. The subtyping is generated from the inclusion $\texttt{Prop} \subset \texttt{Type}_1$ and $\texttt{Type}_i \subset \texttt{Type}_{i+1}$, by extension to products, covariantly on the codomain.

CC_ω differs from the "Generalised" Calculus of Constructions (GCC) [4] which extends the Calculus of Constructions with a slightly weaker cumulative hierarchy of type universes (it does not have $\texttt{Prop} \subset \texttt{Type}_1$ and it is not compatible with products). Thanks to its compatibility with products, the hierarchy of CC_ω (comparatively called fully cumulative in Luo [11]) is easier to work with.

Along the formulae-as-type paradigm, types in `Prop` denotes propositions and terms typed of these types are proofs of these propositions.

Thanks to the normalisation of ECC [11], hence of CC_ω, type inference and type checking in CC_ω are decidable (terms of CC_ω are written à la Church with explicitly typed abstractions).

Type checkers for CC_ω (or GCC) have been implemented on computer from 1985 [8, 12] (with further extensions leading to the Coq [2] and LEGO proof assistants). To abstract over the use of type universe levels, Huet [10] proposed to introduce some form of *typical ambiguity* in the type checker of CC_ω: each time a reference to a type universe is given, its level is left anonymous (i.e. the user writes `Type` without mentioning the level) and the type checker handles a graph of constraints between anonymous universes and relies on a decidable graph algorithm of non-circularity check to ensure the mapping of anonymous universes to actual numerical levels. Similarly, Pollack introduced typical ambiguity in the type-checker of *GCC*, but still retaining the possibility of using explicit level of universes. The theory of anonymous universes in GCC has been carefully studied (among other related concepts) by Harper and Pollack [9].

Both algorithms use universe variables that are associated to every occurrence of `Type` occurring in a derivation of some judgement $\Gamma \vdash t : T$. The derivation enforces constraints on the universe variables and the derivation is sound iff the graph of constraints is acyclic (i.e. if the universe variables are mappable to positive integers so that the constraints are true).

The current paper refines Huet's algorithm for Coq by reducing the size of the graph of constraints. This is achieved by using algebraic universe levels in a free algebra generated by the numerical levels, a successor operator `suc` and a maximum-of-two-integers operator `max`. As a result, the graph of constraints for a type inference problem $\Gamma \vdash t : ?$ only contains nodes for the occurrences of `Type` occurring in Γ and t. All other occurrences, the ones occurring inside the derivation and the ones occurring in the type of t (if typable) are algebraic expressions built on the formers. We call $CC^a_{\omega d}$ the extension of CC_ω with algebraic universes to which definitions also are added.

In the first section, we give a type-directed presentation of $CC^a_{\omega d}$. In the second section, we explore the form of the algebraic constraints that appear in derivations of $CC^a_{\omega d}$. Especially, we show that under special reasonable conditions, constraints between algebraic universes are reducible without time complexity penalty to simple constraints between universe variables only. The third section addresses the issue of the Calculus of Inductive Constructions (CCI) as implemented in the Coq proof assistant [2]. Especially, the previous analysis is extended to the subset of CCI that is relevant for the question of type universes. We conclude with a few remarks.

1 $CC^a_{\omega d}$: type-checking CC_ω with algebraic universes

We present an extension of *CC* with Luo's full cumulative hierarchy of universes, and definitions. Our presentation is syntax-directed, in the same vein as Harper-Pollack's presentation of GCC (e.g. Tables 4 or 9 in [9]) or, implicitly, as Luo's type inference algorithm (Definition 6.2.2 in [11], see also Table 5-3 in [12]). Since it is syntax-directed, a derivation $\Gamma \vdash t : T$ can directly be interpreted as a type inference algorithm taking an inference problem $\Gamma \vdash t : ?$ as input and returning, if successful, a type T of t.

More generally, our system does not only include type universes at given numerical levels. It also has universes at floating levels represented by universe variables so that the a type inference problem $\Gamma \vdash t : ?$ actually returns a judgement $\Gamma \overset{\mathcal{C}}{\vdash} t : T$ where T is a type for t and $\mathcal{C}$ is a graph of constraints on the universe variables to be satisfied to ensure the typability of t. In case all universe variables are distinct in Γ and t, this gives a system similar to the system with anonymous universes (where each anonymous universe is associated to a fresh universe variable) in Harper-Pollack [9] (Table 8, or Table 12 for a system with definitions). However, in our system, the inferred type T, if any, is not a type scheme built on fresh universe variables, but an algebraic expression built on the universe variables occurring in Γ and t. And similarly for the set of constraints $\mathcal{C}$.

1.1 Syntax

Let i ranges over *numerical universes* ($i \in \mathbb{N}^*$). Let α ranges over an infinite set of *universe variables*. An *atomic universe* is either a numerical universe or a universe variable. An *algebraic universe* is an expression built from atomic universes by means of a formal successor function suc and a formal maximum function max.

$$l ::= i \mid \alpha$$
$$e ::= l \mid \mathsf{suc}(e) \mid \mathsf{max}(e, e)$$

A *sort* in $CC^a_{\omega d}$ is either a type universe or the propositional sort $\texttt{Prop}$.

$$s ::= \texttt{Type}(e) \mid \texttt{Prop}$$

To define terms, we assume the existence of an infinite set of *variable names* whose inhabitants are written x, y, z, ... The *terms* are defined by the following grammar.

$$t, u, v, T, U, V ::= x \mid \forall x : t.t \mid \lambda x : t.t \mid (t\ t) \mid s$$

We consider terms up to α-conversion. By $t\{u/x\}$ we denote the capture-avoiding substitution of x by u in t.

Typing contexts, or simply *contexts*, are lists of *declarations*, of the form $x : T$, or *definitions*, of the form $x := t : T$. Formally, a context is defined by the grammar

$$\Gamma ::= \epsilon \mid \Gamma, x : t \mid \Gamma, x := t : T$$

where ϵ denotes an empty context (left implicit, unless it causes an ambiguity). We write $(x : T) \in \Gamma$ (respectively $(x := t : T) \in \Gamma$ to say that the corresponding declaration (respectively definition) is part of Γ.

1.2 Conversion, subtyping and constraints

Typing in $CC^a_{\omega d}$ is up to conversion. Since the system has definitions the convertibility is not only based on β-reduction but also on δ-reduction whose effect is to replace variables defined in the context by their value. Thanks to the strong normalisation of *ECC* and the confluence of β-reduction (which obviously still holds in presence of δ-reduction), convertibility of typed terms can be decided by reduction to weak head normal form. For this, it is useful to characterise the set of weak head normal forms. We first define evaluation contexts E as follows.

$$E ::= [] \mid E\, t$$

with notation $E[t]$ to denote the replacement of the hole of E by t. Due to the presence of definitions, the characterisation of weak head normal forms is relative to a context Γ that can contains definitions. We say that t is a *weak head normal form* (shortly *whnf*) relatively to Γ in the following cases.

– w is a "constructed value", i.e. an expression of the form $\forall x : t.t$, or $\lambda x : t.t$ or s
– w is an expression $E[x]$ with x not defined in Γ

The relation t weak head reduces to w relatively to Γ, written $t \downarrow_\Gamma w$ is inductively defined by the following clauses.

$$\frac{t \text{ in weak head normal form}}{t \downarrow_\Gamma t}$$

$$\frac{E[u\{t/x\}] \downarrow_\Gamma w}{E[(\lambda x : T.u)t] \downarrow_\Gamma w} \qquad \frac{(x := t : T) \in \Gamma \quad E[t] \downarrow_\Gamma w}{E[x] \downarrow_\Gamma w}$$

Convertibility involves *algebraic constraints* which consist of equalities and inequalities relating algebraic universes. We say that a constraint is *purely arithmetical* if it does not involve universe variables. Purely arithmetical constraints are directly evaluable in $\mathbb{N}^*$ by respectively interpreting suc and max as the successor and maximum functions. We write $\mathbb{N} \models e = e'$ (or $\mathbb{N} \models e \leq e'$) to express that e and e' are purely arithmetical and that the constraint is true in $\mathbb{N}$. A constraint between atomic universes only is called *atomic*.

An *assignment* σ of universe variables is a mapping from the set of universe variables to the set of strictly positive integers. The interpretation $\sigma(e)$ of an algebraic universes e along an assignment σ is defined by interpreting suc and max respectively as the successor and maximum functions on positive numbers. If t is a term, the assignment $\sigma(t)$ is a term obtained by applying σ on the type universes. Similarly for the assignment $\sigma(\Gamma)$ of a context Γ.

The interpretation of a set of constraint along an assignment σ is defined by interpreting the inequalities and equalities on the set of positive numbers. We say that σ *validates* the constraint set $\mathcal{C}$, and we write $\sigma \models G$, if the equalities and inequalities of $\mathcal{C}$, interpreted in $\mathbb{N}$, are true. If there exists an assignment that validates $\mathcal{C}$ we say that $\mathcal{C}$ is *stratifiable*.

Convertibility on whnf and on terms is mutually inductively defined by the following clauses.

$$\frac{}{\mathtt{Type}(e) \overset{\{e=e'\}}{=_w} \mathtt{Type}(e')} \qquad \frac{\mathbb{N} \models e = e'}{\mathtt{Type}(e) \overset{\emptyset}{=_w} \mathtt{Type}(e')} \qquad \frac{}{\mathtt{Prop} \overset{\emptyset}{=_w} \mathtt{Prop}}$$

$$\frac{t_1 \overset{\mathcal{C}_1}{=} t_1' \qquad t_2 \overset{\mathcal{C}_2}{=} t_2'}{\forall x : t_1.t_2 \overset{\mathcal{C}_1 \cup \mathcal{C}_2}{=_w} \forall x : t_1'.t_2'} \qquad \frac{t_1 \overset{\mathcal{C}_1}{=} t_1' \qquad t_2 \overset{\mathcal{C}_2}{=} t_2'}{\lambda x : t_1.t_2 \overset{\mathcal{C}_1 \cup \mathcal{C}_2}{=_w} \lambda x : t_1'.t_2'}$$

$$\frac{t_1 \downarrow_\Gamma w_1 \quad t_2 \downarrow_\Gamma w_2 \quad w_1 \overset{\mathcal{C}}{=_w} w_2}{t_1 \overset{\mathcal{C}}{=} t_2} \qquad \frac{\text{for all k}, t_k \overset{\mathcal{C}_k}{=} t_k'}{xt_1 \,...\, t_n \overset{\bigcup_k G_k}{=_w} xt_1' \,...\, t_n'}$$

Subtyping is covariant on the codomain of products and includes universe cumulativity. On applications and abstractions it is equivalent to convertibility. Subtyping is mutually and inductively defined on whnf-terms and terms by the following clauses:

$$\frac{}{\mathtt{Type}(e) \overset{e\le e'}{\le}_w \mathtt{Type}(e')} \qquad \frac{\mathbb{N} \models e \le e'}{\mathtt{Type}(e) \overset{\emptyset}{\le}_w \mathtt{Type}(e')}$$

$$\frac{}{\mathtt{Prop} \overset{\emptyset}{\le}_w \mathtt{Type}(e)} \qquad \frac{}{\mathtt{Prop} \overset{\emptyset}{\le}_w \mathtt{Prop}}$$

$$\frac{T_1' \overset{\mathcal{C}_1}{=} T_1 \qquad T_2 \overset{\mathcal{C}_2}{\le} T_2'}{\forall x : T_1.T_2 \overset{\mathcal{C}_1\cup\mathcal{C}_2}{\le}_w \forall x : T_1'.T_2'} \qquad \frac{\lambda x : T_1.t_2 \overset{\mathcal{C}}{=}_w \lambda x : T_1'.t_2'}{\lambda x : T_1.t_2 \overset{\mathcal{C}}{\le}_w \lambda x : T_1'.t_2'}$$

$$\frac{t_1 \downarrow_\Gamma w_1 \qquad t_2 \downarrow_\Gamma w_2 \qquad w_1 \overset{\mathcal{C}}{\le}_w w_2}{t_1 \overset{\mathcal{C}}{\le} t_2} \qquad \frac{E[x] \overset{\mathcal{C}}{=}_w E'[x]}{E[x]...t_n \overset{\mathcal{C}}{\le}_w E'[x]}$$

1.3 Typing rules

Since the typing rules involve convertibility and subtyping, they also involve constraints. The syntax-directed presentation is so that the sequent $\Gamma \overset{\mathcal{C}}{\vdash} t : T$ can be read as a function depending of Γ and t that returns, if possible, both a type T for t and a constraint set $\mathcal{C}$ to be satisfied.

The notation $\Gamma \overset{\mathcal{C}}{\vdash} t : s$ in premises means that there exists a sort s such that $\Gamma \overset{\mathcal{C}}{\vdash} t : T$ and $T \downarrow_\Gamma s$. Similarly, the notation $\Gamma \overset{\mathcal{C}}{\vdash} t : \forall x : T_1.T_2$ in premises means that there exists T_1 and T_2 such that $\Gamma \overset{\mathcal{C}}{\vdash} t : T$ and $T \downarrow_\Gamma \forall x : T_1.T_2$).

Since all products are valid in $CC^a_{\omega d}$ (i.e. $CC^a_{\omega d}$ is full as a PTS), we only need two-level judgements. Especially, in the typing rule of abstraction the product $\forall x : T.T'$ is necessarily valid.

$$\frac{}{\Gamma_1, x : t, \Gamma_2 \overset{\emptyset}{\vdash} x : t} \qquad \frac{}{\Gamma_1, x := t : T, \Gamma_2 \overset{\emptyset}{\vdash} x : T} \qquad \frac{\Gamma \overset{\mathcal{C}}{\vdash} t : s \quad \Gamma, x : t \overset{\mathcal{C}'}{\vdash} t' : s'}{\Gamma \overset{\mathcal{C}\cup\mathcal{C}'}{\vdash} (\forall x : t.t') : \mathcal{P}rod(s, s')}$$

$$\frac{\Gamma \overset{\mathcal{C}}{\vdash} T : s \quad \Gamma, x : T \overset{\mathcal{C}'}{\vdash} t' : T'}{\Gamma \overset{\mathcal{C}\cup\mathcal{C}'}{\vdash} (\lambda x : T.t') : (\forall x : T.T')} \qquad \frac{\Gamma \overset{\mathcal{C}}{\vdash} t : (\forall x : T_1.T_2) \quad \Gamma \overset{\mathcal{C}'}{\vdash} t' : T' \quad T' \overset{\mathcal{C}''}{\le} T_1}{\Gamma \overset{\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}''}{\vdash} (t\ t') : T_2\{t'/x\}}$$

$$\frac{}{\Gamma \overset{\emptyset}{\vdash} \mathtt{Type}(e) : \mathtt{Type}(\mathsf{suc}(e))} \qquad \frac{}{\Gamma \overset{\emptyset}{\vdash} \mathtt{Prop} : \mathtt{Type}(1)}$$

with $\mathcal{P}rod(s, s')$ defined by

- $\mathcal{P}rod(s, \mathtt{Prop}) = \mathtt{Prop}$
- $\mathcal{P}rod(\mathtt{Prop}, \mathtt{Type}(e)) = \mathtt{Type}(e)$
- $\mathcal{P}rod(\mathtt{Type}(e), \mathtt{Type}(e')) = \mathtt{Type}(\mathsf{max}(e, e'))$

We also define a judgement expressing that a context is *valid* relatively to a set of constraints.

$$\frac{\Gamma \overset{\mathcal{C}_1}{\vdash} \quad \Gamma \overset{\mathcal{C}_2}{\vdash} T : s}{\Gamma, x : T \overset{\mathcal{C}_1 \cup \mathcal{C}_2}{\vdash}} \qquad \frac{\Gamma \overset{\mathcal{C}_1}{\vdash} \quad \Gamma \overset{\mathcal{C}_2}{\vdash} t : T \quad \Gamma \overset{\mathcal{C}_3}{\vdash} T' : s \quad T \overset{\mathcal{C}_4}{\leq_{w\Gamma}} T'}{\Gamma, x := t : T' \overset{\mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3 \cup \mathcal{C}_4}{\vdash}}$$

In contrast with Harper-Pollack, we do not have universe polymorphism for definitions (see [9], Table 12): definitions share the same universe levels and add more constraints than if expanded.

If the universes in Γ, t and T are purely arithmetical, the typing system collapses to the operational presentation of CC_ω with definitions in the style of Harper-Pollack's operational presentation of GCC. Indeed, we have a result similar to Harper-Pollack's Theorem 3.3.

Proposition 1.

1. *$\Gamma \vdash$ in $CC_{\omega d}$ iff $\Gamma \overset{\emptyset}{\vdash}$ in $CC^a_{\omega d}$*
2. *$\Gamma \vdash t : T$ in $CC_{\omega d}$ iff there exists T' such that, in $CC^a_{\omega d}$, $\Gamma \overset{\emptyset}{\vdash} t : T'$ and $\Gamma \overset{\emptyset}{\vdash}$ and $T' \overset{\emptyset}{\leq_{w\Gamma}} T$*

where $CC_{\omega d}$ is CC_ω (rules as in Luo [11] but without Σ-types) extended with the rules

$$\frac{\Gamma_1, x := t : T, \Gamma_2 \vdash}{\Gamma_1, x := t : T, \Gamma_2 \vdash x : T} \qquad \frac{\Gamma \vdash t : T}{\Gamma, x := t : T \vdash}$$

Proof. The proof is standard by induction. It relies on the confluence of the $\beta\delta$ reduction and on its strong normalisation for the typed terms.

1.4 Type-checking with anonymous universes

The hierarchy of countable universes is far more powerful than what is probably needed in practise to formalise mathematics. One would probably live well with two or three universes. In spite of that, it is also quite convenient to live with *typical ambiguity*, as introduced by Russell and Whitehead, for which it is implicitly admitted that there exists a disambiguation of the universes that respects the stratification.

The implementation of typical ambiguity is the purpose of Huet [10] and one of the objectives of Harper-Pollack [9].

In practise, users of the type-checkers for CC with universes just write an anonymous `Type` while the type-checker internally binds the anonymous `Type` to some `Type`(α) where α is a fresh universe variable. Evaluable constraints between integers are then replaced by formal constraints between universe variables, and an algorithm ultimately checks if there exists an assignment of the universe variables by positive integers that validates the constraints.

The previous type system can be used to check the validity of context (or of a judgement) in presence of anonymous universes. One just has to replace each occurrence of an anonymous universe by a fresh universe variable and to use the type system as a validity-checker (or a type inference algorithm). Given Γ where the anonymous universes have been replaced by variables, if the type system produces a derivation of $\Gamma \overset{\mathcal{C}}{\vdash}$, then the validity of Γ is reduced to the stratifiability of $\mathcal{C}$. Indeed, we have a result similar to Harper-Pollack's Theorems 5.2 and 6.5.

Proposition 2. *Let σ an assignment. We have.*

1. *$\sigma(\Gamma) \overset{\emptyset}{\vdash}$ iff $\Gamma \overset{\mathcal{C}}{\vdash}$ for some $\mathcal{C}$ such that and $\sigma \models \mathcal{C}$*
2. *$\sigma(\Gamma) \overset{\emptyset}{\vdash} \sigma(t) : T$ iff $\Gamma \overset{\mathcal{C}}{\vdash} t : T'$ for some $\mathcal{C}$ such that $\sigma \models \mathcal{C}$ and some T' such that $\sigma(T') = T$.*

More generally, the above proposition holds not only if Γ and t contain universe variables but also if they contain arbitrary algebraic universes.

Proof. The proof is by induction. The interesting cases are the convertibility and subtyping cases involving type universes. Also, the cases involving `max` and `suc` relates the formal and effective roles of these operators.

We now turn on to the decision of the stratifiability of constraints.

2 Solving stratifiability

2.1 Structure of the sets of constraints

We now study the form of constraints in judgements. Especially, we are interested in the form of the algebraic constraints in $\mathcal{C}$ for the judgements $\Gamma \overset{\mathcal{C}}{\vdash}$ or $\Gamma \overset{\mathcal{C}}{\vdash} t : T$ when Γ and t only contains atomic universes. It turns out that the central notions here are the notions of conclusion-algebraic term and of atomically-upper-bounded constraints.

A term t is called *non-algebraic* if all occurrences of `Type`(e) in t have the form `Type`(l). A term t is called *conclusion-algebraic* if its weak-head normal form is. A whnf-term w is called *conclusion-algebraic* in the following cases:

– w is $\mathtt{Type}(e)$ or $\mathtt{Prop}$
– w is $E[x]$ or $w = \lambda x : T.t$ which are non-algebraic
– w is $\forall x : T.T'$ and T is non-algebraic while T' is conclusion-algebraic

We say that an algebraic universe e is of *depth* 0 if it is atomic or of the form $\mathsf{max}(e', e'')$ with e' and e'' of depth 0. We say that an algebraic universe e is of *depth* 1 if it is of depth 0, or of the form $\mathsf{suc}(e')$ with e' of depth 0, or of the form $\mathsf{max}(e', e'')$ with e' and e'' of depth 1.

Thanks to the associativity of the maximum function, to the distributivity of the successor over the maximum and to the evaluation rules of these functions on integers, any algebraic universes of depth 1 can be represented in a way that respects its numerical interpretation using a n-ary maximum operator as follows

$$\mathsf{max}(i, \alpha_1, \ldots, \alpha_n, \mathsf{suc}(\alpha'_1), \ldots, \mathsf{suc}(\alpha'_n)) \ .$$

If, in the first clause defining *conclusion-algebraic* terms, e is of depth 1, we say that t is *depth-1 conclusion-algebraic.*

An algebraic constraint is said *atomically-upper-bounded* if it is an equality of the form $l = l'$ or an inequality of the form $e \leq l$. It is *depth-1 atomically-upper-bounded* if e is of depth 1 in $e \leq l$. We have the following properties

Lemma 1.

1. *If $t \overset{\mathcal{C}}{=}_w t'$ with t and t' non-algebraic, then $\mathcal{C}$ has only constraints of the form $l' = l$.*
2. *If $t \overset{\mathcal{C}}{\leq}_w t'$ with t' non-algebraic and t conclusion-algebraic then $\mathcal{C}$ has only atomically-upper-bounded constraints.*
3. *If $t \overset{\mathcal{C}}{\leq}_w t'$ with t' non-algebraic and t depth-1 conclusion-algebraic then $\mathcal{C}$ has only depth-1 atomically-upper-bounded constraints.*

Proof. The first part is trivial by induction. The second part, by induction, relies on the characterisation of algebraic terms.

Based on the properties of the subtyping algorithm, we have:

Proposition 1 *Let Γ such that for all $(x : T) \in \Gamma$ or $(x := t : T) \in \Gamma$, T is depth-1 conclusion-algebraic. Let t be non algebraic. If $\Gamma \overset{\mathcal{C}}{\vdash} t : T$ then*

1. *T is depth-1 conclusion-algebraic*
2. *$\mathcal{C}$ only contains depth-1 atomically-upper-bounded constraints*

Proof. The proof proceeds by induction on the derivation of $\Gamma \overset{\mathcal{C}}{\vdash} t : T$, using Lemma 1 for the application case.

As a consequence, for problems involving typical ambiguity (i.e. for problems where the universes are atomic and all universe variables are distinct) we have

Corollary 1 *If Γ and t are non algebraic and $\Gamma \overset{\mathcal{C}}{\vdash} t : T$ then T is depth-1 conclusion-algebraic and $\mathcal{C}$ is depth-1 atomically-upper-bounded.*

This result is important from a time complexity point of view. Indeed constraints that are atomically-upper-bounded can be decomposed into conjunction of atomic constraints without having to consider the stratifiability of disjoint sets (hence without requiring backtracking). This results in a stratifiability problem not more complex than Coq Huet's algorithm based on variables, but on a lightened graph that does not any longer mention those universes that occur only in the inferred types, universes on which no upper constraints apply and that are useless.

Proposition 3. *If $\mathcal{C}$ is a set of atomically-upper-bounded algebraic constraints, there is a set $\mathcal{C}'$ of atomic constraints such that $\mathcal{C}$ is stratifiable iff $\mathcal{C}'$ is stratifiable.*

Proof. Because $\mathsf{max}(i, \alpha_1, \ldots, \alpha_n, \mathsf{suc}(\alpha'_1), \ldots, \mathsf{suc}(\alpha'_n)) \leq l$ is equivalent to

$$i \leq l \wedge \alpha_1 \leq l \wedge \ldots \wedge \alpha_n \leq l \wedge \alpha'_1 < l \wedge \ldots \wedge \alpha'_n < l\,.$$

Set of atomic constraints can be seen as oriented graphs whose nodes are the numerical universes and universe variables occurring in the constraints, and whose oriented edges represent either greater-or-equal or strictly-greater constraints. We write $|\mathcal{C}|$ for the set of atomic constraints associated to the depth-1 atomically-upper-bounded set $\mathcal{C}$.

2.2 Stratifiability of graphs of atomic constraints

A *cycle* is any oriented path in the graph that traverses at least one "strictly greater" edge.

Lemma 2. *A set of constraints $\mathcal{C}$ is stratifiable iff $|\mathcal{C}|$ has no cycle*

We first describe how to ensure that a set of atomic constraints of the form $\alpha = \alpha'$, $\alpha \leq \alpha'$ or $\alpha < \alpha'$ has no cycle. This is an abstract presentation of the algorithm implemented in Coq.

A graph of atomic constraints is represented as a set of classes of equal universes $\hat{\alpha}$ each of them equipped with

- A set $U^{\geq}_{\hat{\alpha}}$ denoting the set of universes explicitly required to be greater or equal to the universes in $\hat{\alpha}$.
- A set $U^{>}_{\hat{\alpha}}$ denoting the set of universes explicitly required to be strictly greater to the universes in $\hat{\alpha}$.

We write $\uparrow^{\geq}_{\hat{\alpha}}$ for the set of universe variables greater or equal to the universe variable α. It corresponds to the transitive closure of $U^{\geq}_{\hat{\alpha}}$. It can be computed from the family of sets $U^{\geq}$ and $U^{>}$ in time $O(mn)$ at worst, where n is the number of equivalence classes and m the number of edges.

We write $\uparrow_\alpha^>$ for the set of universe variables strictly greater to the universe variable α. It corresponds to the subset of the transitive closure of $U_{\hat\alpha}^{\geq}$ that follows at least one $>$ edge. It can also be computed in time $O(mn)$ at worst.

We write $\uparrow_\alpha^{\alpha'}$ for the set of universes that are greater or equal to α and less or equal to α'. It can also be computed in time $O(mn)$ at worst.

Let G a graph that has no cycle. We enumerate sufficient conditions so that the extension of G with a constraint $l = l'$, $l \leq l'$ or $l < l'$ has no cycle. Let G' be the extended graph. We write $U_{\hat\alpha}^{\geq'}$ (respectively $U_{\hat\alpha}^{>'}$) for the new set of universes explicitly required to be greater or equal (respectively strictly greater) to the universe α.

- $G \cup \{\alpha = \alpha'\}$ has no cycle iff $\alpha \notin \uparrow_{\alpha'}^>$ and $\alpha' \notin \uparrow_\alpha^>$. Moreover,
  - if $\alpha' \in \uparrow_\alpha^{\geq}$ then
    - the new common equivalence class of α and α' is $\hat\alpha \cup \hat{\alpha'} \cup \uparrow_\alpha^{\alpha'}$,
    - $U_{\hat\alpha}^{\geq'} = \bigcup_{\beta \in \uparrow_\alpha^{\alpha'}} U_{\hat\beta}^{\geq}$
    - $U_{\hat\alpha}^{>'} = \bigcup_{\beta \in \uparrow_\alpha^{\alpha'}} U_{\hat\beta}^{>}$
  - if $\alpha \in \uparrow_{\alpha'}^{\geq}$ then similarly with $\uparrow_{\alpha'}^{\alpha}$.
  - if $\alpha \notin \uparrow_{\alpha'}^{\geq}$ and $\alpha' \notin \uparrow_\alpha^{\geq}$ then
    - the new common equivalence class of α and α' is $\hat\alpha \cup \hat{\alpha'}$,
    - $U_{\hat\alpha}^{\geq'} = U_{\hat\alpha}^{\geq} \cup U_{\hat{\alpha'}}^{\geq}$
    - $U_{\hat\alpha}^{>'} = U_{\hat\alpha}^{>} \cup U_{\hat{\alpha'}}^{>}$
- $G \cup \{\alpha > \alpha'\}$ has no cycle iff $\alpha' \notin \uparrow_\alpha^{\geq}$. Moreover, if $\alpha \notin \uparrow_{\alpha'}^>$, we have $U_{\hat{\alpha'}}^{>'} = U_{\hat{\alpha'}}^{>} \cup \{\alpha\}$
- $G \cup \{\alpha \geq \alpha'\}$ has no cycle iff $\alpha' \notin \uparrow_\alpha^>$. Moreover,
  - if $\alpha' \in \uparrow_\alpha^{\leq}$, then
    - the new common equivalence class of α and α' is $\hat\alpha \cup \hat{\alpha'} \cup \uparrow_\alpha^{\alpha'}$,
    - $U_{\hat\alpha}^{\geq'} = \bigcup_{\beta \in \uparrow_\alpha^{\alpha'}} U_{\hat\beta}^{\geq}$
    - $U_{\hat\alpha}^{>'} = \bigcup_{\beta \in \uparrow_\alpha^{\alpha'}} U_{\hat\beta}^{>}$
  - if $\alpha' \notin \uparrow_\alpha^{\leq}$, then $U_{\hat{\alpha'}}^{\geq'} = U_{\hat{\alpha'}}^{\geq} \cup \{\alpha\}$

To extend the previous algorithm to the case of constraints the form $\alpha = i$, $\alpha \leq i$, $\alpha < i$, $i \leq \alpha$ and $i < \alpha$, we associate two extra informations to each class of equal universes.

- An integer $l_{\hat\alpha}$ intended to be a lower bound to the universes in $\hat\alpha$.
- An optional integer $u_{\hat\alpha}$ explicitly required to be an upper bound to the universes in $\hat\alpha$.

These informations, that have to be updated each time a constraint is added (even for constraints not mentioning numerical levels), allow to determine easily whether a constraint involving an integer is consistent.

3 Type-checking *CIC* with algebraic universes

We now focus on the Calculus of Inductive Constructions (CCI) such as implemented in the Coq system [2]. This calculus has constructions for fixpoints, cofixpoints, case analysis, inductive and coinductive types, constructors in inductive and coinductive types, local definitions and a "cast" operator to force the exact type of a term. It also has an extra sort `Set` of type $\texttt{Type}_1$ and included in $\texttt{Type}(1)$. Products in the sort `Set` are predicative or not, depending on the version of Coq considered. We restrict our study to inductive and coinductive types that are the only constructions equipped with special conditions regarding the universes. We also only consider the predicative version of the sort `Set`

The extra sort `Set`, in its predicative version, has type $\texttt{Type}_1$ and is a subtype of $\texttt{Type}_1$. This can be expressed by adding the level 0 to the grammar of atomic levels to which correspond the following extra typing and subtyping rules.

$$\frac{}{\Gamma \overset{\emptyset}{\vdash} \texttt{Set} : \texttt{Type}(1)} \qquad \frac{}{\texttt{Set} \overset{\emptyset}{\leq_w} \texttt{Type}(e)} \qquad \frac{}{\texttt{Set} \overset{\emptyset}{\leq_w} \texttt{Prop}}$$

Specifications of (co-)inductive types are declared in the context. We consider specifications of mutually defined (co-)inductive types so that the grammar of contexts is extended as follows:

$$\begin{array}{l} \Gamma ::= \ldots \mid \Gamma, I :=_b S \\ S ::= \{I_i(\overrightarrow{x:T})(\overrightarrow{x_i:T_i}) : s_i := \overrightarrow{C_{ij}:T'_{ij}}\} \end{array}$$

Specifications of (co-)inductive types defines block of mutually (co-)inductive types. In the specification $\{I_i(\overrightarrow{x:T})(\overrightarrow{x_i:T_i}) : s_i := \overrightarrow{C_{ij}:T'_{ij}}\}$, the context $\overrightarrow{x:T}$ represents the type of the global parameters of the family. It is common to all types of the block. For each i, the context $\overrightarrow{x_i:T_i}$ represents the type of the specific parameters of each I_i in the family and s_i is the sort of $I_i\vec{x}\vec{x_i}$. Otherwise said, I_i has type $\forall\overrightarrow{x:T}.\forall\overrightarrow{x_i:T_i}.s_i$. The constructors C^I_{ij} of I_i have the respective types $\forall\overrightarrow{x:T}.T'_{ij}$. The conclusion of each T'_{ij} has the form $I_i\vec{x}\overrightarrow{u_{ij}}$ with $\overrightarrow{u_{ij}:T_i}$. Other occurrences of the I_i can occur in T'_{ij} in "strictly positive" positions. In $I :=_b S$, b is a boolean telling if it is a specification of inductive or coinductive types. The validity of (co-)inductive types specification is expressed by the following rule:

$$\frac{\begin{cases} \Gamma \overset{\mathcal{C}}{\vdash} \overrightarrow{x:T}, \\ \text{for all } i: \ \Gamma, \overrightarrow{x:T} \overset{\mathcal{C}_i}{\vdash} \overrightarrow{x_i:T_i}, \\ \text{for all } i,j: \ \overrightarrow{(\Gamma, \overrightarrow{x:T}, I_i : \forall\overrightarrow{x:T}.\forall\overrightarrow{x_i:T_i}.s_i \overset{\mathcal{C}_{ij}}{\vdash} T'_{ij} : s_{ij})}, \\ \text{for all } i,\ j,\ T_{ij} \text{ satisfies the strict positivity criterion for } \overrightarrow{I_i}, \\ \mathcal{C}' = \mathcal{C} \cup \bigcup_i \mathcal{C}_i \cup \bigcup_{ij} \mathcal{C}_{ij} \cup \bigcup_{ij} cst_{ind}(s_{ij} \leq s_i), \\ \text{purely arithmetical constraints in } \bigcup_{ij} cst_{ind}(s_{ij} \leq s_i) \text{ are true} \end{cases}}{\Gamma, I :=_b \{I_i(\overrightarrow{x:T})(\overrightarrow{x_i:T_i}) : s_i := \overrightarrow{C_{ij}:T'_{ij}}\} \overset{\mathcal{C}'}{\vdash}}$$

where $\Gamma \overset{\mathcal{C}}{\vdash} \overrightarrow{x:T}$ is inductively defined by (ϵ denotes an empty sequence)

$$\frac{}{\Gamma \overset{\emptyset}{\vdash} \epsilon} \qquad \frac{\Gamma \overset{\mathcal{C}}{\vdash} \overrightarrow{x:T} \quad \Gamma, \overrightarrow{x:T} \overset{\mathcal{C}'}{\vdash} T' : s}{\Gamma \overset{\mathcal{C}\cup\mathcal{C}'}{\vdash} \overrightarrow{x:T}, x' : T'}$$

and $cst_{ind}(s \leq s')$ is defined by

$$\begin{array}{lll} cst_{ind}(\texttt{Prop} \leq s') & = \emptyset & \\ cst_{ind}(s \leq \texttt{Prop}) & = \emptyset & \text{(impredicativity of } \texttt{Prop}\text{)} \\ cst_{ind}(\texttt{Type}(e) \leq \texttt{Type}(e')) & = \{e \leq e'\} & (e \text{ and } e' \text{ not both purely arithmetical}) \end{array}$$

We refer to Coquand-Paulin [6] or the Coq reference manual [2] for an exact description of the strict positivity criterion (it has no other effect on the universes constraints than the ones given in the rule above).

Among others constructions, the terms of the CIC include expressions for constructors and (co-)inductive types.

$$t ::= \ldots \mid I_i \mid C_{ij}^I \mid \ldots$$

The corresponding typing rules are

$$\frac{}{\Gamma_1, I :=_b \{I_i(\overrightarrow{x:T})(\overrightarrow{x_i:T_i}) : s_i := \overrightarrow{C_{ij} : T'_{ij}}\}, \Gamma_2 \overset{\emptyset}{\vdash} C_{ij}^I : \forall \overrightarrow{x:T}.T'_{ij}}$$

$$\frac{}{\Gamma_1, I :=_b \{I_i(\overrightarrow{x:T})(\overrightarrow{x_i:T_i}) : s_i := \overrightarrow{C_{ij} : T'_{ij}}\}, \Gamma_2 \overset{\emptyset}{\vdash} I_i : \forall \overrightarrow{x:T}.\forall \overrightarrow{x_i:T_i}.s_i}$$

The properties of $CC_{\omega d}^a$ extend to the Set-predicative Calculus of Inductive Constructions.

Proposition 2 *Let Γ such that all $I :=_b \{I_i(\overrightarrow{x:T})(\overrightarrow{x_i:T_i}) : s_i := \overrightarrow{C_{ij} : T'_{ij}}\}$ or $x : T$ or $x := t : T$ in Γ are such that T and all s_i are depth-1 conclusion-algebraic, and $\overrightarrow{T}$ and all $\overrightarrow{T_i}$ and $\overrightarrow{T_{ij}}$ are non algebraic. Let t be non algebraic. If $\Gamma \overset{\mathcal{C}}{\vdash} t : T$ then*

- *T is depth-1 conclusion-algebraic*
- *$\mathcal{C}$ contains only depth-1 atomically-upper-bounded constraints*

Corollary 2 *If Γ and t are non algebraic and $\Gamma \overset{\mathcal{C}}{\vdash} t : T$ then T is depth-1 conclusion-algebraic and $\mathcal{C}$ is depth-1 atomically-upper-bounded.*

4 Remarks

Contravariant subtyping. Luo [11] mentioned an alternative definition of subtyping that is contravariant with respect to the domain of products. The modified rule is

$$\frac{T_1' \overset{\mathcal{C}_1}{\leq} T_1 \quad T_2 \overset{\mathcal{C}_2}{\leq} T_2'}{\forall x : T_1.T_2 \overset{\mathcal{C}_1 \cup \mathcal{C}_2}{\leq_w} \forall x : T_1'.T_2'}$$

The properties of the sets of constraints in $CC^a_{\omega d}$ still hold with this modified rule.

Modules. The Coq proof assistant is equipped with a module system. Courant [7] points out that to ensure a true modularity when refining the signature of a module, it would be necessary to declare constraints in the signatures (and not only at the global level as it is the case in Coq version 8.0 [2]). Would it then be worth to let the users refer to algebraic universes?

Explicit numerical universes. Algebraic universes (as described in this paper) are used internally in the Coq proof assistant since version 7.1 but version 8.0 [2] still does not allow the user to use numerical universes. Our analysis is a step towards proving that it can be offered without increase of the complexity of the stratifiability algorithm.

References

1. Stefano Berardi. *Type dependence and constructive mathematics.* PhD thesis, Universitá di Torino, 1990.
2. The Coq Development Team. Coq 8.0 reference manual, 2004.
3. Thierry Coquand. *Une Théorie des Constructions.* PhD thesis, Université Paris 7, January 1985.
4. Thierry Coquand. An analysis of Girard's paradox. In *Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1986.
5. Thierry Coquand and Gérard Huet. The Calculus of Constructions. *Information and Computation*, 76(2/3), 1988.
6. Thierry Coquand and Christine Paulin-Mohring. Inductively defined types. In P. Martin-Löf and G. Mints, editors, *Proceedings of Colog'88*, volume 417 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.
7. Judicaël Courant. Explicit universes for the Calculus of Constructions. In Victor A. Carreño, César A. Muñoz, and Sofiène Tahar, editors, *Theorem Proving in Higher Order Logics: 15th International Conference, TPHOLs 2002*, volume 2410 of *Lecture Notes in Computer Science*, pages 115–130, Hampton, VA, USA, August 2002. Springer-Verlag.
8. Project Formel. The Calculus of Constructions. documentation and users's guide. version 4.10. Technical Report 110, INRIA, Rocquencourt, France, August 1989.
9. Robert Harper and Robert Pollack. Type checking with universes. *Theoretical Computer Science*, 89(1):107–136, 1991.

10. Gérard Huet. Extending the Calculus of Constructions with Type:Type, 1987.
11. Zhaohui Luo. *An Extended Calculus of Constructions.* PhD thesis, University of Edinburgh, 1990.
12. Robert Pollack. *The Theory of LEGO, a Proof Checker for the Extended Calculus of Constructions.* PhD thesis, University of Edinburgh, 1994.
13. Jan Terlouw. Een nadere bewijstheoretische analyse van gstt's. Technical report, Faculty of Mathematics and Computer Science, University of Nijmegen, Netherlands, April 1989.