A Canonical Form for Universe Levels in Impredicative Type Theory

Yoan Géran
Mines Paris PSL, Centre de Recherche en Informatique, France
Université Paris-Saclay, Laboratoire Méthodes Formelles, ENS Paris-Saclay, France
Lab. ICube UMR 7357 CNRS Université de Strasbourg, France

Abstract

The 0-imax-successor algebra, where imax: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is the function defined by $\text{imax}(n, 0) = 0$ and $\text{imax}(n, S(m)) = \max(n, S(m))$, is used to represent universe levels in impredicative type theory, in particular with universe polymorphism which introduces level variables, so it is present in proof systems such as Rocq and Lean. In particular, we need to know when two elements of this algebra are equivalent, and we may also want to decide the inequality. In this article, we introduce a canonical form for the terms of this algebra, and we provide a canonization algorithm. It permits deciding level equivalence by checking the canonical form equality, and also permits easily checking if a level is smaller than another one.

2012 ACM Subject Classification Theory of computation → Type theory; Theory of computation → Equational logic and rewriting

Keywords and phrases universe levels, canonical form, impredicativity, imax algebra

Digital Object Identifier 10.4230/LIPIcs.CSL.2026.39

Related Version
Full Version: `https://perso.crans.org/geran/research/canonical_levels.pdf`

Supplementary Material *Software (Rock formalisation)*: `https://gitlab.crans.org/geran/level_formalisation`, archived at `swh:1:dir:79df183835f26e788d568780b46e800bf0d0cfc6`

Acknowledgements I want to thank my PhD advisors Olivier Hermant and Gilles Dowek for the helpful discussions and comments.

1 Introduction

The formalization of mathematical theorems and the verification of software lead to the development of many logical systems. Predicate Logic is a quite general theory but does not allow for instance to quantify over predicates, preventing the expression of some propositions. Then, more expressive logic have been introduced through the years. This paper being motivated by the universe polymorphism in impredicative type theory, the introduction will briefly remind the history of these theories, to understand what they bring.

Pure Type Systems

A lot of type theories are based on extensions of Church's simply-typed λ-calculus which does not permit expressing terms over arbitrary types (preventing for instance to talk about all the groups). To address this, Martin-Löf introduced a dependent type theory with a type of all types [25], and later, to avoid paradoxes such as Girard's one [12, 20], introduced a distinction between small types and large types (which are types containing types, and are also called universes) [27].

In the same years, Girard and Reynolds independently invented System F, an extension of Church's simply-typed λ-calculus with type polymorphism, and even later, Girard presented System F_ω which adds type operators *i.e.* the ability to quantify over terms to create types.

The Calculus of Constructions [13] introduced by Coquand in his PhD thesis combined features from both Martin-Löf Type Theory and System F_ω. This system allows quantifying on types or terms to build new types and new terms, and it is the pinnacle of the λ-cube of Barendregt [4], which classifies type systems depending on the quantification possibilities.

The Calculus of Constructions is an elegant system with strong properties such as normalization and logical consistency. However, quantification over Type is not possible since it makes the system incoherent. This led Coquand to generalize the system with a predicative hierarchy of universes [12], in the same way as predicative Martin-Löf type theory [26]. This new system, which we will call CC^∞, contains a countable sequence of universes $\mathrm{Type}_0 : \mathrm{Type}_1 : \cdots$, where Type_0 is the universe of the propositions, the indices being referred to as universes levels.

These logical systems are generalized under the name of *Pure Type Systems* [5, 6].

▶ **Definition 1.** *A Pure Type System (PTS) is defined by a set of sorts $\mathcal{S}$ (that corresponds to universes), a set of axioms $\mathcal{A} \subseteq \mathcal{S}^2$ and a set of rules $\mathcal{R} \subseteq \mathcal{S}^3$.*

$\mathcal{A}$ describes the sorts typing (s_1 has the type s_2 when $(s_1, s_2) \in \mathcal{A}$), and $\mathcal{R}$ describes the possible quantifications and their typing rules. The terms are the following, where $s \in \mathcal{S}$ and x ranges an infinite set of variables.

$$t ::= s \mid x \mid \Pi\, x : t \cdot t \mid \lambda\, x : t \cdot t \mid t\, t.$$

Both CC^∞ and predicative Martin-Löf type theory have a set of sorts indexed over the natural numbers, with for all $i \in \mathbb{N}$, $\mathrm{Type}_i : \mathrm{Type}_{i+1}$. Their difference resides in their set of rules. These rules permit to type products, so we recall the typing rule for products.

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

Impredicativity

With the aim of building a consistent system, paradoxes such as Girard's one should be avoided. When Coquand analysed it, he found that a product from Type to Type could not live in Type: it should live in a greater type (hence the distinction between small and large types).

With an infinite hierarchy of universes, this principle remains: a product from Type_i to Type_j should live in a greater universe. Therefore, in the predicative Martin-Löf type theory, the set of rules is $\left\{\mathrm{Type}_i, \mathrm{Type}_j, \mathrm{Type}_{\max(i,j)}\right\}$. The choice of CC^∞ is different (and does not break the consistency either): a product from Type_i to Type_0 lives in Type_0, so it follows the rules $\left\{\mathrm{Type}_i, \mathrm{Type}_j, \mathrm{Type}_{\mathrm{imax}(i,j)}\right\}$ where $\mathrm{imax} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ is defined for all $i, j \in \mathbb{N}$ by $\mathrm{imax}(i, 0) = 0$ and $\mathrm{imax}(i, j+1) = \max(i, j+1)$.

This corresponds to the so-called impredicativity of Prop (hence the name imax for impredicative max) which notably permits to say that we can quantify over all the propositions and still get a new proposition. Accepting or rejecting impredicativity is a philosophical questioning: some consider that $\Pi\, P : \mathrm{Prop} \cdot P \to P$ should be a proposition (which is of course true), while others think that a proposition cannot be created by quantifying over all the propositions.

Universe Polymorphism

A PTS can be enriched with universe polymorphism which allows the user to quantify over universes [28, 23, 14]. For instance, it permits declaring simultaneously the identity for all the types of any universes with $\lambda\, s\colon \mathcal{S} \cdot \lambda\, A\colon s \cdot \lambda\, x\colon A \cdot x$. This feature adds universe variables to the language of a PTS. In the case of CC^∞, it is equivalent to extend the syntax of the levels with level variables.

▶ **Definition 2** (Levels). *A level is a term of the grammar*

$$\ell \coloneqq 0 \mid S(\ell) \mid \max(\ell, \ell) \mid \mathrm{imax}(\ell, \ell) \mid x$$

where x is an element of a countable set of variables $\mathbf{X}$. We denote by $\mathbf{L}$ the set of levels, and we say that a level is ground if it does not contain any variable.

▶ **Definition 3.** *We call $CC^\infty_\forall$ the extension of CC^∞ with prenex universe polymorphism.*

The universe polymorphic identity of $\mathrm{CC}^\infty_\forall$ is the term

$$\mathrm{id} \coloneqq\coloneqq \lambda\, i\colon \mathbf{L} \cdot \lambda\, A\colon \mathrm{Type}_i \cdot \lambda\, x\colon A \cdot x.$$

We can use it by instantiating the level variable. For instance, $\mathrm{id}\ 1\ \mathrm{Prop}$ is the identity of Prop while $\mathrm{id}\ 2\ \mathrm{Type}_1$ is Type_1's one. This instantiation is done throughout substitution functions, which replace a level variable by a level, and valuation functions which replace level variables by integers.

▶ **Definition 4** (Valuation). *A function $\sigma\colon \mathbf{X} \to \mathbb{N}$ is called a valuation. For all valuations σ, we define inductively the value of a level ℓ over σ, denoted as $[\![\ell]\!]_\sigma$, with*

$$[\![0]\!]_\sigma = 0 \qquad [\![S(\ell)]\!]_\sigma = S([\![\ell]\!]_\sigma) \qquad [\![x]\!]_\sigma = \sigma(x)$$
$$[\![\max(\ell_1, \ell_2)]\!]_\sigma = \max\big([\![\ell_1]\!]_\sigma, [\![\ell_2]\!]_\sigma\big) \qquad [\![\mathrm{imax}(\ell_1, \ell_2)]\!]_\sigma = \mathrm{imax}\big([\![\ell_1]\!]_\sigma, [\![\ell_2]\!]_\sigma\big)$$

This interpretation through the valuations explains why, even if levels are abstract terms, we defined them with the same symbols 0, s, max and imax that are used for the natural numbers. Indeed, the ground levels can clearly be identified as the natural numbers and the levels' semantic, through the valuations, justifies to use the same symbol and permits to see the valuations as functions that *realise* levels, turning them into ground ones.

Besides, two levels can also be compared using these valuations. They are equivalent if they give the same ground levels through any valuation.

▶ **Definition 5** (Level comparison). *Let $\ell_1, \ell_2 \in \mathbf{L}$. We say that $\ell_1 \leqslant \ell_2$ if for all valuations σ, $[\![\ell_1]\!]_\sigma \leqslant [\![\ell_2]\!]_\sigma$. In the same way, we say that $\ell_1 \equiv \ell_2$ if for all valuations σ, $[\![\ell_1]\!]_\sigma = [\![\ell_2]\!]_\sigma$. Hence, $\ell_1 \equiv \ell_2$ if and only if $\ell_1 \leqslant \ell_2$ and $\ell_2 \leqslant \ell_1$. And we say that ℓ_1 and ℓ_2 are incomparable if neither $\ell_1 \leqslant \ell_2$ nor $\ell_2 \leqslant \ell_1$.*

This equivalence shows that universes such as Type_x and $\mathrm{Type}_{\max(x,x)}$ should be identified. However, it is not obvious to check. It is not syntactic, like it was without universe polymorphism, and the imax function makes it complicated.

The aim of this paper is to address this problem. Since it is a word problem, a direct solution consists in finding a canonical form and an algorithm which computes the canonical form of a given term. The level equivalence is then checked by syntactic comparison of the canonical form. We follow this path. To do this, we study the 0-imax-successor algebra, and we provide a canonical form for its terms. Besides, this equivalence is decidable by a reduction to Presburger arithmetic, and so we provide an additional motivation to our work.

Motivation

Our main motivation lies in the interoperability between proof systems. Indeed, it became a big challenge in the research on proof-checking, which aims to avoid the redevelopment of the same proof. Instead of developing translators from each system to another one, *logical frameworks* propose to define theories in a common language, which makes translation easier.

The $\lambda\Pi$-calculus modulo rewriting ($\lambda\Pi/\equiv$) [10] is a logical framework that extends $\lambda\Pi$ (the simply-typed λ-calculus with dependent types) with higher-order rewrite rules [16, 29] that can be used to define functions, but also types; terms are then identified modulo β and these rewrite rules. The computational part of the type theories can then be represented using the expressiveness of rewrite systems.

The Calculus of Constructions and its subtheories can be expressed in $\lambda\Pi/\equiv$ [8], and, in [15], Cousineau and Dowek showed how to express some PTS. Therefore, several systems have been encoded in $\lambda\Pi/\equiv$: HOL-Light [30, 1], Agda [19], Matita [1], but also parts of Rocq [18, 9]. Besides, since there exist multiple implementations of $\lambda\Pi/\equiv$ such as Dedukti [2], Lambdapi [24], or Kontroli [17], these embeddings have been implemented, leading to effective translations [30, 21, 22].

To define $\mathrm{CC}^{\infty}_{\forall}$ in $\lambda\Pi/\equiv$, we need to define its levels. It can be done with a type `nat` together with functions `max`, and `imax`, and rewrite rules to define them. This permits to express CC^{∞} in $\lambda\Pi/\equiv$, but the equivalence relation that comes with level variables adds some difficulties.

Indeed, for all term u of $\mathrm{CC}^{\infty}_{\forall}$, let us note $|u|$ its translation in $\lambda\Pi/\equiv$, and let us consider a function $f\colon \mathrm{Type}_i \to \mathrm{Type}_j$ and a term $t\colon \mathrm{Type}_k$ where $k \equiv i$. Since $f\,t$ is well-typed, then $|f\,t|$ should be well-typed in $\lambda\Pi/\equiv$. Therefore, $|t|$ should have the type $|\mathrm{Type}_i|$, whereas it has the type $|\mathrm{Type}_k|$. We deduce that $|\mathrm{Type}_i|$ and $|\mathrm{Type}_k|$ should be convertible types, and then that equivalent levels should be convertible terms.

Related Work

The 0-max-successor algebra is well-studied, and so, some solutions exist in the predicative case. In [31], Voevodsky represented each level as $\max(n, n_1 + x_1, \ldots, n_k + x_k)$ where $n \geqslant \max(n_1, \ldots, n_k)$. Then, if there exists $i \neq j$ such that $x_j = x_i$, we simplify the term and keep only $\max(n_i, n_j) + x_i$. Therefore, we obtain a minimal representation for the 0-max-successor algebra.

In [19], Genestier encoded the universe polymorphism of Agda in $\lambda\Pi/\equiv$ using a similar idea and a representation modulo associativity and commutativity (for the max symbol). Blanqui gave another presentation of this algebra in [7], with an encoding without matching modulo associativity and commutativity.

The 0-imax-successor algebra is less studied. An encoding is proposed in [3], but it does not fully reflect the equalities; for instance, the levels $\max(\mathrm{imax}(x, y), x)$ and $\max(x, y)$ are not convertible. Besides, Férey also worked on the encoding of universe polymorphism [18].

Finally, an algorithm to check level inequality, and so level equivalence, is presented in [11], but it does not rely on a canonical form.

Outline and Contributions

We use the same idea presented above in the predicative case: find a subset E of levels such that any level can be represented as $\max U$ with $U \subset E$, and such that $\max U$ is a minimal representation that ensures uniqueness property for any other minimal representation $\max V$:

$$\max U \equiv \max V \iff U = V.$$

In the predicative case, $E = \mathbb{N} \cup \{n + x \mid n \in \mathbb{N}, x \in \mathbf{X}\}$; and the minimal representation consists in having one term n (the maximum of two integers can be simplified) and for all $x \in \mathbf{X}$ at most one term $n + x$ since $\max(n + x, m + x) = \max(n, m) + x$. To obtain the canonical representation, we push the successor symbols inside the max, and we obtain $\max U$ with $U \subset E$. Then, U can be simplified by removing u if there exists $v \in U$ such that $v \neq u$ and $u \leqslant v$, leading to the minimal representation.

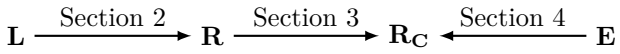
This gives us the intuition that we need. The elements of the subset should be very basic and simple in the sense that it is not equivalent to a maximum of other levels.

Section 2 studies the algebra and introduces $\mathbf{S_C}$, a set of basic terms, called *canonical sublevels*. It turns out that these terms do not have the same expressive power than levels, hence we extend $\mathbf{L}$ into $\mathbf{E}$, the *extended levels*. But, each level can be represented as a maximum of elements of $\mathbf{S_C}$; such a maximum is a *representation*, they will form the set of representations $\mathbf{R}$.

Then, in Section 3 we introduce $\mathbf{R_C}$, the set of *minimal representations* which are representations whose elements are incomparable, and we show that any representation has a unique minimal representation.

Moreover, we will see that $\mathbf{R_C}$ is not stable by level substitution, which can generate extended levels. But, in order to have a correct translation of substitution in $\lambda\Pi/\equiv$, we should be able to identify $|u\{x \coloneqq v\}|$ with $|u|\{x \coloneqq |v|\}$. That is why we generalize the canonical form to the extended levels in Section 4, showing that all extended levels have a minimal representation: its canonical form. Then , we present the canonization algorithm in Appendix A.

Figure 1 summarises this. Besides, the proofs of the results are given in the full version of the paper, as well as a canonization algorithm. Finally, the existence and uniqueness are also proved in Rocq since we formalised the canonization algorithm for the levels [1].



$\mathbf{L}$: levels $\mathbf{E}$: extended levels $\mathbf{S_C}$: canonical sublevels

$\mathbf{R}$: representations (maximum of canonical sublevels)

$\mathbf{R_C}$: minimal representations (elements are incomparable)

Figure 1 Outline.

2 Level Representation

As said in the introduction, in this section we find a set of sublevels that fit our objective of level representation. Before that, let us introduce a basic notion that improves the readability of the text.

▶ **Definition 6.** *Let u be a level and σ be a valuation. We say that u is active (under the valuation σ) if $[\![u]\!]_\sigma \neq 0$. We also say that σ activates u.*

[1] `https://gitlab.crans.org/geran/level_formalisation`

The valuation σ will often be left implicit. For instance, we can say that $\mathrm{imax}(u, v)$ is $\max(u, v)$ if v is active and 0 otherwise in the sense that for all valuation σ, $[\![\mathrm{imax}(u, v)]\!]_\sigma$ is $[\![\max(u, v)]\!]_\sigma$ if v is active under σ and 0 otherwise.

2.1 Levels as Maximum

The very first step to simplify the terms is to express any level as a maximum of levels that do not contain any max, that is the principle of our idea. The successor can be distributed over max since for all $u, v \in \mathbf{L}$, $S(\max(u, v)) \equiv \max(S(u), S(v))$, and the next two observations show how to distribute imax over max.

▶ **Observation 7.** *For all $u, v, w \in \mathbf{L}$,*

$$\mathrm{imax}(u, \max(v, w)) \equiv \max(\mathrm{imax}(u, v), \mathrm{imax}(u, w)).$$

▶ **Observation 8.** *For all $u, v, w \in \mathbf{L}$,*

$$\mathrm{imax}(\max(u, v), w) \equiv \max(\mathrm{imax}(u, w), \mathrm{imax}(v, w)).$$

Then, any level can be expressed as a maximum of levels without max. Note that for this, we consider that max takes a finite set of levels as argument. We obtain the following theorem.

▶ **Theorem 9.** *For all $t \in \mathbf{L}$, there exists $u_1, \ldots, u_n$ in the grammar*

$$\ell \coloneqq 0 \mid S(\ell) \mid \mathrm{imax}(\ell, \ell) \mid x$$

such that $t \equiv \max(u_1, \ldots, u_n)$. Moreover, the variables that appear in $u_1, \ldots, u_n$ are exactly the variables that appear in t.

2.2 Simplification of the Levels

We can now focus on levels without maximum. The uniqueness property sought for the representation requires the levels to be very basic, and then we search to simplify them.

The main issue in the grammar of Theorem 9 is imax because it is asymmetric. We aim to restrict the localisation of the imax symbol to specific parts of the levels in order to understand and control their influence on the levels semantic.

Firstly, we recall some equivalences that are direct consequences of the semantics of imax. They permit to deal with 0 and the successor.

▶ **Observation 10.** *For all $u, v \in \mathbf{L}$,*

$$\mathrm{imax}(u, 0) \equiv 0 \qquad \mathrm{imax}(u, S(v)) \equiv \max(u, S(v))$$

And we show how to remove imax symbol in second argument of imax.

▶ **Observation 11.** *For all $u, v, w \in \mathbf{L}$,*

$$\mathrm{imax}(u, \mathrm{imax}(v, w)) \equiv \max(\mathrm{imax}(u, w), \mathrm{imax}(v, w)).$$

Thus, by applying the simplification suggested by observations 10 and 11, we can restrict the second argument of imax to be a variable. It is more complicated for its first argument. We can simplify the level when it is 0.

▶ **Observation 12.** *For all* $v \in \mathbf{L}$, $\mathrm{imax}(0, v) \equiv v$.

Moreover, we can distribute S over imax, but this cannot be done as directly as we distribute the S over max, as shown in the next example.

▶ **Example 13.** We consider the levels $t_1 = S(\mathrm{imax}(y, x))$ and $t_2 = \mathrm{imax}(S(y), S(x))$. By considering a valuation σ such that $\sigma(x) = 0$ and $\sigma(y) = 1$, $[\![t_1]\!]_\sigma \neq [\![t_2]\!]_\sigma$, therefore $t_1 \not\equiv t_2$.

Besides, we can observe that $S(\mathrm{imax}(u, v))$ is at least $S(v)$, but can also be $S(u)$ when v is active. Therefore, the simplification rule has to take into account the two relevant cases depending on the value of v.

▶ **Observation 14.** *For all* $u, v \in \mathbf{L}$,

$$S(\mathrm{imax}(u, v)) \equiv \max(S(v), \mathrm{imax}(S(u), v)).$$

Finally, observations 10–12 and 14 lead to this grammar restriction.

▶ **Theorem 15.** *For all* $t \in \mathbf{L}$, *there exists* $u_1, \ldots, u_n$ *in the grammar*

$$\ell \coloneqq S^{k+1}(0) \mid S^k(x) \mid \mathrm{imax}(\ell, x)$$

such that $t \equiv \max(u_1, \ldots, u_n)$. *Moreover, the variables that appear in* $u_1, \ldots, u_n$ *are exactly the variables that appear in* t.

▶ Remark 16. For all t in the grammar of Theorem 15, there exists $x_1, \ldots, x_n \in \mathbf{X}$, and $v = S^{k+1}(0)$ or $v = S^k(x)$ such that

$$t = \mathrm{imax}(\mathrm{imax}(\mathrm{imax}(\cdots \mathrm{imax}(v, x_1), x_2) \cdots)), x_{n-1}), x_n).$$

Such a term t will be denoted by $[v, x_1, \ldots, x_n]$. Intuitively, this term is the maximum of

- x_n,
- x_{n-1} if x_n is active,
- x_{n-2} if x_n and x_{n-1} are active,
- etc.
- v if all the x_i are active.

2.3 Introducing New Levels

Here, we continue the simplification process in order to find simple enough terms to reach the uniqueness property. Indeed, the terms of the grammar of Theorem 15 are still not simple enough.

▶ **Example 17.** Let us consider $t = \max(\mathrm{imax}(x, y), x)$. Then, $t \equiv \max(x, y)$.

The problem is the following: if $\mathrm{imax}(x, y)$ permits to take x into account if y is active, it also takes y into account in all cases. Then, it is redundant with y and lead to the equivalence $\mathrm{imax}(x, y) \equiv \max(y, \mathrm{imax}(x, y))$. We would like to obtain $\mathrm{imax}(x, y) = \max(y, t)$ with some level t, but $\mathrm{imax}(x, y)$ cannot be simplified more.

In fact, the second argument of imax has too many responsibilities since it should be taken into account, but it is also a condition to take into account the first argument.

The second concern leads us to devise the introduction of a term "if y then x" such that $[\![\text{if } y \text{ then } x]\!]_\sigma$ is 0 if $[\![y]\!]_\sigma = 0$ and $[\![x]\!]_\sigma$ otherwise. This permits us to simplify $\mathrm{imax}(x, y)$ into $\max(y, \text{if } y \text{ then } x)$, and since if y then $x \leqslant x$, $\max(y, \text{if } y \text{ then } x, x)$ can be turned into $\max(y, x)$.

Since, the imax functions are nested in the grammar of Theorem 15, we may need to have multiple variables as conditions; we generalize this idea of new terms and extend the level's grammar with two symbols $\mathcal{V}$ and $\mathcal{C}$.

▶ **Definition 18** (Extended levels). *An extended level is a term of the grammar*

$$\ell \coloneqq 0 \mid S(\ell) \mid \max(\ell,\ell) \mid \operatorname{imax}(\ell,\ell) \mid x \mid \mathcal{V}(\{\ell,\dots,\ell\},\ell,k) \mid \mathcal{C}(\{\ell,\dots,\ell\},k)$$

where $k \in \mathbb{N}$. We extend $[\![\cdot]\!]_\sigma$ and the level comparison to the extended levels with

$$[\![\mathcal{V}(E,u,k)]\!]_\sigma = \begin{cases} 0 & \text{if } \exists v \in E, [\![v]\!]_\sigma = 0 \\ [\![u]\!]_\sigma + k & \text{otherwise.} \end{cases}$$

$$[\![\mathcal{C}(E,k)]\!]_\sigma = \begin{cases} 0 & \text{if } \exists u \in E, [\![u]\!]_\sigma = 0 \\ k & \text{otherwise.} \end{cases}$$

We denote by $\mathbf{E}$ the set of extended levels. Levels of the form $\mathcal{V}(E,u,k)$ or $\mathcal{C}(E,k)$ are called sublevels*.*

The symbols $\mathcal{V}$ and $\mathcal{C}$ stand for "variable sublevel" and "constant sublevel" in the sense that their semantic consists in taking into account a non-constant or a constant extended level u when a set of extended levels E only contain active ones.

▶ **Definition 19.** *We denote by $\mathbf{S}$ the set of sublevels. Let $u \in \mathbf{S}$, $u = \mathcal{V}(E,v,k)$ or $u = \mathcal{C}(E,k)$. We call E the* verification conditions *of u denoted by $\mathrm{VC}(u)$, and k is its* constant part *denoted by $\omega(u)$. We also define the* variable part *of u denoted by $\nu(u)$ which is 0 in the case of a constant sublevel and v in the case of a variable sublevel.*

Besides, we say that a verification condition $w \in E$ is checked *(by a valuation σ) if $[\![w]\!]_\sigma \neq 0$ and we say that the verification condition E are checked if for all $w \in E$, $[\![w]\!]_\sigma \neq 0$.*

The verification conditions and the parts of a sublevel determine its value and if it is active. Indeed, when u is active, its value is $[\![\omega(u)]\!]_\sigma + [\![\nu(u)]\!]_\sigma$. And for u to be active, the verification condition of u should be checked (otherwise the value of u is automatically 0), and on top of that $[\![\omega(u)]\!]_\sigma + [\![\nu(u)]\!]_\sigma$ (which is then the value of u) should not be 0.

▶ **Proposition 20.** *For all $u \in \mathbf{S}$ and for all valuations σ, $[\![u]\!]_\sigma = \omega(u) + [\![\nu(u)]\!]_\sigma$ if u is active and 0 otherwise. Moreover, u is active if and only if its verifications conditions are checked and $\omega(u) + [\![\nu(u)]\!]_\sigma \neq 0$.*

Keeping with our idea, we want to show that any level is equivalent to a maximum of sublevels, so we do it for the grammar presented in Theorem 15. Here is the intuition behind the replacement of a nested imax by a maximum of sublevels. In $[S^k(y), x_1, \dots, x_n]$,

- x_n is always considered, so we take $\mathcal{V}(\{\}, x_n, 0)$,
- x_{n-1} is considered if x_n is active, so we take $\mathcal{V}(\{x_n\}, x_{n-1}, 0)$,
- x_{n-2} is considered if x_n and x_{n-1} are active, so we take $\mathcal{V}(\{x_n, x_{n-1}\}, x_{n-2}, 0)$,
- $\dots$
- $S^k(y)$ is considered if all the x_i are active, so we take $\mathcal{V}(\{x_n,\dots,x_1\}, y, k)$.

The situation is similar in the case $[S^k(0), x_1, \dots, x_n]$, except that the last taken term is $\mathcal{C}(\{x_n,\dots,x_1\}, k)$.

▶ **Proposition 21.** *Let $n \in \mathbb{N}$, $E = \{x_1,\dots,x_n\} \subset \mathbf{X}$, $k \in \mathbb{N}$, $x_0 \in \mathbf{X}$, and for all $i \in \{0,\dots,n\}$, $u_i = \mathcal{V}(\{x_{i+1},\dots,x_n\}, x_i, 0)$. Then,*

$$[S^k(x_0), x_1, \dots, x_n] \equiv \max(\mathcal{V}(E, x_0, k), u_1, \dots, u_n),$$
$$[S^{k+1}(0), x_1, x_n] \equiv \max(\mathcal{C}(E, k+1), u_1, \dots, u_n).$$

One could note that since the grammar of $\mathbf{E}$ is really permissive, for all $u \in \mathbf{E}$, we have the trivial equivalence $u \equiv \mathcal{V}(\emptyset, u, 0)$ to express u as a sublevel. This shows that the sublevels are at least as expressive as the levels, but this equivalence is a nonsense in terms of level simplification. Proposition 21 is a much stronger and useful result since it states that the verification conditions and the variable part of variable sublevels can be restricted to variables to have a complete representation of levels of $\mathbf{L}$. However, we made the choice of presenting extended levels without this restriction to facilitate the level instantiation (developed in Section 4). Indeed, a variable will then be replaced by any level, and we want to make this substitution transparent in our level representation.

2.4 An Appropriate Set of Sublevels

We have restrained our study to the sublevels whose verification conditions and the variable part (in the case of variable sublevels) are variables. Now, we show that some of them can be obtained as a maximum of other ones. The first restriction is related to the representation of 0. Indeed, for all $E \subset \mathbf{X}$, $\mathcal{C}(E, 0) \equiv 0$. Since we already have $0 \equiv \max(\emptyset)$, we can remove all these sublevels. The second restriction is a little more subtle and is illustrated with this example.

▶ **Example 22.** With $t_1 = \mathcal{V}(\emptyset, x, 0)$ and $t_2 = \mathcal{V}(\{x\}, x, 0)$, we have $t_1 \equiv t_2$ since for all valuation σ, $[\![t_1]\!]_\sigma = \sigma(x) = [\![t_2]\!]_\sigma$.

The issue here is the fact that the variable part of a variable sublevel does not necessarily appear in its first argument. This is the key of the following equivalence.

▶ **Proposition 23.** *Let $x \in \mathbf{X}$, $E \subset \mathbf{X} \setminus \{x\}$ and $k \in \mathbb{N}$. Then*

$$\mathcal{V}(E, x, k) \equiv \max(\mathcal{V}(E \cup \{x\}, x, k), \mathcal{C}(E, k)).$$

If $k = 0$, the sublevel $\mathcal{C}(E, k)$ obtained with Proposition 23 is removed accordingly to the first restriction. We end up with the following set of sublevels which permits to express any level.

▶ **Definition 24** (Canonical sublevels)**.** *A canonical sublevel is an element of the set*

$$\mathbf{S_C} = \{\mathcal{V}(E, x, k) \mid E \subset \mathbf{X}, x \in E\} \cup \{\mathcal{C}(E, k) \mid E \subset \mathbf{X}, k > 0\}.$$

▶ **Theorem 25.** *Let $t \in \mathbf{L}$. Then there exists a finite $U \subset \mathbf{S_C}$ such that $t \equiv \max U$. Moreover, the variables that appear in the elements of U are exactly the variables that appear in t.*

▶ **Definition 26.** *Let U be a finite subset of $\mathbf{S_C}$. We say that $\max U$ is a representation, and we denote by $\mathbf{R}$ the set of representations.*

Besides, for all $t \in \mathbf{E}$, we say that $\max U$ is a representation of t if $t \equiv \max U$, and we say that the elements u of U are the elements of the representation.

The canonical sublevels correspond to the set of sublevels that we were searching for. We could try to merge the two types of sublevels by introducing a special variable $\mathbf{1}$ such that for all valuation σ, $\sigma(\mathbf{1}) = 1$. We can then see $\mathcal{C}(E, k+1)$ as $\mathcal{V}(E \cup \{\mathbf{1}\}, \mathbf{1}, k)$. This simplifies some results but makes the presentation less clear, and the distinction should still be done in a lot of cases.

▶ Remark 27. Let $u \in \mathbf{S_C}$ and let σ be a valuation. Then u is active if and only if all its verification conditions are checked.

3 A Canonical Form for levels

The previous section defined $\mathbf{R}$, the set of representations, and showed that any level is equivalent to one of its elements. The goal of this one is to show that any level has a minimal representation and that it is unique. This will be the canonical form.

▶ **Definition 28** (Minimal representation). *Let* $\max U \in \mathbf{R}$. *We say that* $\max U$ *is minimal if and only if for all* $u, v \in U$ *such that* $u \neq v$, u *and* v *are incomparable. We denote by* $\mathbf{R_C}$ *the set of the minimal representations.*

By Theorem 25, any level has a representation. Since the set of representations is well-founded with the inclusion order, any level has a minimal representation. The challenging part is the uniqueness of minimal representations. To show it, we study the core of the definition of a minimal representation: the sublevel comparison.

3.1 Sublevel Comparison

The sublevels can be easily compared which is quite normal since we chose them to be very basic. To have $u \leqslant v$, v should be active whenever u is active hence $\mathrm{VC}(v) \subset \mathrm{VC}(u)$. And when they are both active, the value of v should be greater than the one of u.

With two variable sublevels it means that their variable part is the same or else we can set a very large value to $\nu(u)$ in order to falsify the inequality. This also explains that we cannot have $\mathcal{V}(E, x, l) \leqslant \mathcal{C}(F, k)$. And if u is a constant sublevel and v a variable sublevel, we should remember that when they are active, $\nu(v)$ is active (because it is included in $\mathrm{VC}(v)$) and then $\omega(u) \leqslant \omega(v) + 1$ suffices.

▶ **Theorem 29** (Sublevels comparison). *The following equivalences permit to compare elements of* $\mathbf{S_C}$.

$$\mathcal{V}(E, x, l) \not\leqslant \mathcal{C}(F, k) \tag{1}$$
$$\mathcal{C}(E, l) \leqslant \mathcal{C}(F, k) \iff F \subset E \wedge l \leqslant k \tag{2}$$
$$\mathcal{C}(E, l) \leqslant \mathcal{V}(F, x, k) \iff (F \subset E \wedge l \leqslant k + 1) \tag{3}$$
$$\mathcal{V}(E, x, l) \leqslant \mathcal{V}(F, y, k) \iff F \subset E \wedge x = y \wedge l \leqslant k \tag{4}$$

As a corollary, we get that the sublevel equivalence is a syntactic equality, which is expected to ease uniqueness.

▶ **Corollary 30.** *Let* $t_1, t_2 \in \mathbf{S_C}$. *Then* $t_1 \equiv t_2 \iff t_1 = t_2$.

Figure 2 illustrates the comparison of the canonical sublevels on the set of variables $\{x, y\}$ and with 0 or 1 as constant part. We see that we get a greater sublevel by increasing the constant part, reducing the set of verification conditions, or moving to a variable sublevel (in that case, the constant part can be reduced by one).

3.2 The Uniqueness Property

Now, we can show the uniqueness property. We have to show that two equivalent minimal representations $\max U$ and $\max V$ have the same sublevels. Here, we show a slightly different proof. It is more elegant, and it emphasises the property that ensures the uniqueness.

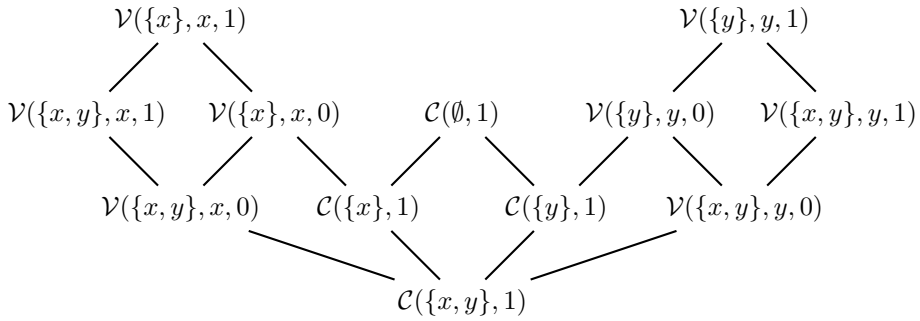


Figure 2 Sublevel comparison on $\{x, y\}$.

We will show that $u \leqslant \max V$ implies the existence of $v \in V$ such that $u \leqslant v$. This statement lead to the uniqueness property. Indeed, if $\max U$ and $\max V$ are two equivalent minimal representation, then for all $u \in U$ there exists a $v \in V$ such that $u \leqslant v$, and similarly there exists $u' \in U$ such that $v \leqslant u'$. Minimality of $\max U$ permits to conclude that $u = u'$, and then $u \equiv v$ and finally $u = v$ by comparison of canonical sublevels.

To prove this statement, the idea is to find, for any sublevel u, a valuation σ such that the only way to have $[\![u]\!]_\sigma \leqslant [\![V]\!]_\sigma$ is to have $v \in V$ with $u \leqslant v$. Such a valuation should only activate the verification condition of u. Moreover, $[\![u]\!]_\sigma$ should be large enough to overtake the other sublevels of the representation. When u is a variable sublevel, it means to set its variable part to a large number.

▶ **Definition 31.** *Let $u \in \mathbf{S_C}$ be a constant sublevel and $V \subset \mathbf{S_C}$ be a set of canonical sublevel. The V-minimal over-valuation of u is the valuation defined by*

$$\sigma(x) = \begin{cases} 0 & \text{if } x \notin \mathrm{VC}(u) \\ 1 & \text{if } x \in \mathrm{VC}(u). \end{cases}$$

▶ **Definition 32.** *Let $u \in \mathbf{S_C}$ be a variable sublevel and $V \subset \mathbf{S_C}$ be a set of canonical sublevel. The V-minimal over-valuation of u is the valuation defined by*

$$\sigma(x) = \begin{cases} 0 & \text{if } x \notin \mathrm{VC}(u) \\ 1 & \text{if } x \in \mathrm{VC}(u) \setminus \nu(u) \\ 2 + \max\{\omega(v) \mid v \in V\} & \text{otherwise.} \end{cases}$$

▶ **Proposition 33.** *Let $u \in \mathbf{S_C}$ be a variable sublevel, $V \subset \mathbf{S_C}$ be a set of canonical sublevel and σ be the V-minimal over-valuation of u. We have*

$$[\![u]\!]_\sigma \leqslant [\![\max V]\!]_\sigma \iff \exists v \in V, u \leqslant v.$$

This proposition is fundamental for the following theorem.

▶ **Theorem 34.** *For all $u \in \mathbf{S_C}$ and $\max V \in \mathbf{R}$, $u \leqslant \max V$ if and only if there exists $v \in V$ such that $u \leqslant v$.*

And we obtain that equivalence of minimal representations is set equality.

▶ **Proposition 35.** *For all $\max U, \max V \in \mathbf{R_C}$, $\max U \equiv \max V \iff U = V$.*

Finally, we obtain the main theorem: the existence and uniqueness of a minimal representation for each level, that is to say a canonical form. First, we show the intuitive property that the minimal representation of a maximum of sublevels is formed with some of them.

▶ **Proposition 36.** *For all* $\max U \in \mathbf{R}$*, there exists a unique* $\max V \in \mathbf{R_C}$ *such that* $\max U \equiv \max V$*. Moreover,* V *is a subset of* U*.*

▶ **Theorem 37** (Minimal Representation)**.** *For all* $t \in \mathbf{L}$*, there exists a unique* $\max U \in \mathbf{R_C}$ *such that* $t \equiv \max U$*. We say that* $\max U$ *is the canonical form of* t*.*

This theorem states the existence of a canonical form function c for $\mathbf{L}$, c being the function that associates any level to its minimal representation.

▶ Remark 38. The key point of this result is Theorem 34. It should be understood as an *independence* property. Indeed, if we consider $\max(u_1, \dots, u_n)$ as a linear combination of $u_1, \dots, u_n$, then this theorem states that the only way to be smaller than a linear combination is to depend on and be smaller than one of the elements of this combination.

This analogy provides a new point of view on our work: $\mathbf{S_C}$ is a "linearly independent" family (uniqueness of the minimal representation) which generates all the levels through "linear combinations".

Moreover, Theorem 34 provides a method to compare two levels, by comparing each sublevel of the minimal representation of the first one to the second one. More generally, two representations are compared in the following way.

▶ **Theorem 39.** *For all* $\max U, \max V \in \mathbf{R}$*,* $\max U \leqslant \max V$ *if and only if for all* $u \in U$*, there exists* $v \in V$ *such that* $u \leqslant v$*.*

4 A Canonical Form for Extended Levels

We are now interested in extending the representation theorem to the whole grammar of extended levels in order to have substitution. Indeed, if $u = \max(\mathcal{V}(\{x\}, x, 0))$ (the canonical form of x), then

$$v = u\{x := \max(y, 1)\} = \max(\mathcal{V}(\{\max(y, 1)\}, \max(y, 1), 0))$$

is not a representation. Since $\mathbf{L}$ is stable by substitution, $u\{x := \max(y, 1)\}$ is a level and then v has a minimal representation (which is $\max(\mathcal{V}(\{y\}, y, 0), \mathcal{C}(\emptyset, 1))$).

Here, we provide a way to find this representation by extending the representation to all extended levels, showing that they have representations.

▶ **Theorem 40.** *For all* $u \in \mathbf{E}$*, there exists* $\max V \in \mathbf{R}$ *such that* $u \equiv \max V$*.*

▶ Remark 41. One could think that we do not have to deal with S and imax because we know that the successor and imax of representations are representations. But we do not yet have this result ; we only know that it is true for representations of levels (because levels can be represented). Moreover, the development of this section will be helpful to develop a recursive canonization algorithm.

4.1 The Successor

In order to define the successor of a canonical sublevel in terms of representation, we define $\mathrm{inc} : \mathbf{S_C} \to \mathbf{S_C}$ such that for all $E \subset \mathbf{X}, k > 0$, $\mathrm{inc}(\mathcal{C}(E, k)) = \mathcal{C}(E, k+1)$ and for all $E \subset \mathbf{X}, x \in E, k > 0$, $\mathrm{inc}(\mathcal{V}(E, x, k)) = \mathcal{V}(E, x, k+1)$. This function nearly increments a canonical sublevel but does not fulfil that objective when some elements of E are not active (it will give 0 instead of 1). That's why we take it in combination with $\mathcal{C}(\emptyset, 1)$ which is 1.

▶ **Proposition 42.** *For all* $u \in \mathbf{S_C}$,

$S(u) \equiv \max(\mathrm{inc}(u), \mathcal{C}(\emptyset, 1))$.

We immediately deduce the following result.

▶ **Proposition 43.** *Let* $\max U \in \mathbf{R}$. *Then,*

$S(\max U) \equiv \max\{\mathrm{inc}(u) \mid u \in U\} \cup \{\mathcal{C}(\emptyset, 1)\}$.

4.2 The Impredicative Maximum

Following the equivalences $\mathrm{imax}(0, u) \equiv u$ and $\mathrm{imax}(u, 0) \equiv 0$, and observations 7 and 8, we have the following equivalence.

▶ **Proposition 44.** *Let* $\max U, \max V \in \mathbf{R_C}$. *We have*

$$\mathrm{imax}(\max U, \max V) \equiv \begin{cases} \max(\emptyset) \equiv 0 & \text{if } V = \emptyset \\ \max V & \text{if } U = \emptyset \\ \max\limits_{\substack{u \in U \\ v \in V}} \mathrm{imax}(u, v) & \text{else.} \end{cases} \tag{5}$$

Then, it is sufficient to show that for all $u, v \in \mathbf{S_C}$, $\mathrm{imax}(u, v)$ has a representation. We will then obtain a representation of $\mathrm{imax}(\max U, \max V)$ by taking the elements of the ones of $\mathrm{imax}(u, v)$ for all $u \in U$ and $v \in V$.

▶ **Proposition 45.** *Let* $v \in \mathbf{S_C}$, $E \subset \mathbf{X}$, $x \in E$ *and* $k \in \mathbb{N}$. *Then,*

$\mathrm{imax}(\mathcal{C}(E, k+1), v) \equiv \max(\mathcal{C}(E \cup \mathrm{VC}(v), k+1), v)$

$\mathrm{imax}(\mathcal{V}(E, x, k), v) \equiv \max(\mathcal{V}(E \cup \mathrm{VC}(v), x, k), v)$

4.3 The Sublevels

We now assume that the verification conditions of a sublevel are representations and that its variable part, in the case of a variable sublevel is a representation as well. First, we show how to transform a sublevel t into a maximum of sublevels whose verifications conditions are variable.

For the sublevel to be active, all its verification conditions should be checked. Since these verification conditions are representations, it means that each of them have an active sublevel. Then, we have to consider all the combinations obtained by taking one sublevel for each verification condition of t. Moreover, a sublevel is active when its verifications conditions are all checked. Then, we have to consider all the combinations of verification conditions of the verification conditions of t.

▶ **Definition 46.** *For all* $t, u_1, \ldots, u_n \in \mathbf{S_C}$, *we define*

$$P(t, u_1, \ldots, u_n) = \begin{cases} \mathcal{C}\left(\bigcup_{0 \leqslant i \leqslant n} \mathrm{VC}(u_i), \omega(t)\right) & \text{if } t \text{ is a constant sublevel} \\ \mathcal{V}\left(\bigcup_{0 \leqslant i \leqslant n} \mathrm{VC}(u_i), \nu(t), \omega(t)\right) & \text{else.} \end{cases}$$

The term $P(t, u_1, \ldots, u_n)$ means that if $u_1, \ldots, u_n$ are checked, then we can take into account the value of the sublevel that we want to simplify (which is then $\nu(t) + \omega(t)$). Then, we consider such terms when $u_i, \ldots, u_n$ are a combination of verification conditions taken from each verification conditions of t. This leads to sublevels whose verification conditions are variables.

▶ **Proposition 47.** *Let* $\max U_1, \ldots, \max U_n \in \mathbf{R}$ *and* t *be a sublevel whose verifications conditions are* $\max U_1, \ldots, \max U_n$*. Then*

$$t \equiv \max\{P(t, u_1, \ldots, u_n) \mid u_1 \in U_1, \ldots, u_n \in U_n\}.$$

▶ Remark 48. It is important to have $\max U_1, \ldots, \max U_n \in \mathbf{R}$ and not only maximum of (possibly not canonical) sublevels. Indeed, we use the fact that $u \in \mathbf{S_C}$ is active if and only if all its verification conditions are checked (Remark 27), which is not always true with non-canonical sublevels. Moreover, note that the result still holds if there exists an i such that $U_i = \emptyset$, since both terms are equivalent to 0.

The Constant Sublevels

With Proposition 47, we have shown how to transform a constant sublevel u whose verification conditions are representations into a representation. Indeed, in the constant sublevel case, for all $u_1, \ldots, u_n \in \mathbf{S_C}$, $P(u_1, \ldots, u_n) \in \mathbf{S_C}$ if $k > 0$ (hence we obtain a representation of t). Besides, if $k = 0$, a representation of t is $\max \emptyset$.

The Variable Sublevels

However, in the variable sublevels, it is not the case; Proposition 47 only permits us to obtain variable sublevels where the verification conditions are variables. Besides, the variable part of $P(u_1, \ldots, u_n)$ is not necessarily a variable. That is why we have to take care of the variable part of variable sublevels.

In $t = \mathcal{V}(V, U, k)$ with $V \in \mathbf{E}$ and $U \in \mathbf{R}$, the value of t when it is activated by σ is the maximum of the value of $[\![u]\!]_\sigma + k$ with $u \in U$. Therefore, t can be easily split into a maximum.

▶ **Proposition 49.** *Let* $\max U \in \mathbf{R}, V \subset \mathbf{E}$*, and* $k \in \mathbb{N}$*. Then,*

$$\mathcal{V}(V, \max U, k) \equiv \max\{\mathcal{V}(V, u, k) \mid u \in U\}.$$

So, it results in sublevels as variable part of variable sublevels, and we handle these in the next proposition.

To transform the $\mathcal{V}(V, u, k)$ obtained with Proposition 49 into a canonical sublevel, we note that its value can be 0, or k, or $\nu(u) + \omega(u) + k$. Then, we need a canonical sublevel with $\omega(u) + k$ as constant part, and $\nu(u)$ as variable part (we will get a constant or a variable sublevel depending on the nature of its variable part $\nu(u)$). Besides, u has to be active to obtain the value $\nu(u) + \omega(u) + k$, then the verification conditions of u have to be verification conditions of the targeted sublevel. However, when u is not active, $\mathcal{V}(V, u, k)$ is not necessarily evaluated to 0 since it can also be evaluated to k when V are checked. Therefore, we add a second sublevel $\mathcal{C}(V, k)$ to keep this behaviour.

▶ **Proposition 50.** *Let* $V \subset \mathbf{E}$*,* $u \in \mathbf{S}$*, and* $k \in \mathbb{N}$*. We note*

$$f(u) = \begin{cases} \mathcal{C}(V \cup \mathrm{VC}(u), k + \omega(u)) & \text{if } u \text{ is a constant sublevel} \\ \mathcal{V}(V \cup \mathrm{VC}(u) \cup \{u\}, \nu(u), k + \omega(u)) & \text{else} \end{cases}$$

Then $\mathcal{V}(V, u, k) \equiv \max(\mathcal{C}(V, k), f(u))$.

Note that when $u \in \mathbf{S_C}$, having $\mathrm{VC}(u)$ as verification conditions is equivalent to having u as verification condition which simplifies $f(u)$ in the case where u is a variable sublevel.

We can apply these two propositions to the $P(u_1, \ldots, u_n)$ from Definition 46. For that, we define the following.

▶ **Definition 51.** *Let $k \in \mathbb{N}$. For all $v, u_1, \dots, u_n \in \mathbf{S_C}$, we define (by noting $u_0 = v$),*

$$f(v, u_1, \dots, u_n) = \begin{cases} \mathcal{C}\Big(\bigcup_{0 \leqslant i \leqslant n} \mathrm{VC}(u_i), k + \omega(v)\Big) & \text{if } v \text{ is a constant sublevel} \\ \mathcal{V}\Big(\bigcup_{0 \leqslant i \leqslant n} \mathrm{VC}(u_i), \nu(v), k + \omega(v)\Big) & \text{else} \end{cases}$$

and

$$Q(v, u_1, \dots, u_n) = \max\Big(\mathcal{C}\Big(\bigcup_{1 \leqslant i \leqslant n} \mathrm{VC}(u_i), k\Big), f(v, u_1, \dots, u_n)\Big).$$

Here, $f(v, u_1, \dots, u_n)$, corresponds to the case where all the verification conditions $u_1, \dots, u_n$ are checked and v is active and $\mathcal{C}\Big(\bigcup_{1 \leqslant i \leqslant n} \mathrm{VC}(u_i), k\Big)$ corresponds to the case where $u_1, \dots, u_n$ are checked but v is not active, hence they form $Q(v, u_1, \dots, u_n)$. Finally, we get the following proposition.

▶ **Proposition 52.** *Let $k \in \mathbb{N}$, $\max U_1, \dots, \max U_n \in \mathbf{R}$, $\max V \in \mathbf{R}$, and let t be the variable sublevel $\mathcal{V}(\{\max U_1, \dots, \max U_n\}, \max V, k)$. Then,*

$$t \equiv \max\{Q(v, u_1, \dots, u_n) \mid u_1 \in U_1, \dots, u_n \in U_n, v \in V\}$$

▶ Remark 53. As for the constant sublevels case, we should take care to consider the sublevel $f(u_1, \dots, u_n)$ only if its constant part is not 0. Otherwise, it is equivalent to 0, and it has to be removed from the max if you want a canonical form.

Besides, one could think that we should have the same consideration with the sublevel $g(v, u_1, \dots, u_n)$ (whose constant part is $k + \omega(v)$) when v is a constant sublevel, but since $v \in \mathbf{S_C}$ (because it is an element of a representation), then $\omega(v) > 0$.

After that, the case of the variable sublevel is solved. We have proved all the induction cases, which terminates the proof of Theorem 40.

General Representation Theorem

By Theorem 40 any level has a representation, and therefore a minimal one by Proposition 36.

▶ **Theorem 54.** *For all $u \in \mathbf{E}$, there exists a unique $v \in \mathbf{R_C}$ such that $u \equiv v$.*

With Theorem 54, we have shown that $\mathbf{R_C}$ is as expressive as $\mathbf{E}$ whereas $\mathbf{R_C} \subsetneq \mathbf{E}$. To finish this section, we compare the expressiveness of the different shape of levels.

If $\mathbf{L}$ is semantically, and even syntactically, a subset of $\mathbf{E}$, one could note that some extended levels are not equivalent to any level. In fact, even some canonical sublevels are not equivalent to any level. For instance, for all $x \in \mathbf{X}$, there is no $u \in \mathbf{L}$ such that $u \equiv \mathcal{C}(\{x\}, 1)$.

In the same way we show that some levels are not expressible with just one canonical level. For instance, for all $x, y \in \mathbf{X}$, there is no $u \in \mathbf{S_C}$ such that $u \equiv \max(x, y)$.

We end up with the inclusion Hasse diagram displayed in Figure 3.

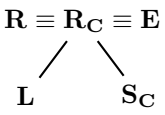


Figure 3 Hierarchy of levels.

5 Conclusion

We studied the 0-imax-successor and introduced a canonical form for its terms, which gives us an easy procedure decision for the equivalence problem. For that, we extended the grammar with new terms called sublevels, and we expressed any term as a maximum of sublevels, what we have called a representation. Since not all representations are actually terms of the algebra, a next step could be to characterize the representations that are. This could lead to an even better understanding of the 0-imax-successor algebra.

We only provide a naive canonization algorithm since level expressions are usually quite small. However, searching for a better algorithm and a good data structures for representations could be useful.

Finally, this representation can be expressed in $\lambda\Pi/\equiv$ with rewrite rules, which is our initial motivation, and it is used in a Work In Progress translator from Lean to Dedukti showing that it can indeed be used to express $\mathrm{CC}^{\infty}_{\forall}$ in $\lambda\Pi/\equiv$. The next step here, is to study how the expression of universe polymorphism, thanks to this level representation, behaves well together with other features such as inductive types or cumulativity.

References

1 Ali Assaf. *A framework for defining computational higher-order logics*. Theses, École polytechnique, September 2015. URL: `https://pastel.archives-ouvertes.fr/tel-01235303`.

2 Ali Assaf, Guillaume Burel, Raphaël Cauderlier, David Delahaye, Gilles Dowek, Catherine Dubois, Frédéric Gilbert, Pierre Halmagrand, Olivier Hermant, and Ronan Saillard. Dedukti : a Logical Framework based on the λΠ-Calculus Modulo Theory, 2016.

3 Ali Assaf, Gilles Dowek, Jean-Pierre Jouannaud, and Jiaxiang Liu. Encoding Proofs in Dedukti: the case of Coq proofs. In *Proceedings Hammers for Type Theories*, Proc. Higher-Order rewriting Workshop, Coimbra, Portugal, July 2016. Easy Chair. URL: `https://inria.hal.science/hal-01330980`.

4 Henk Barendregt. Introduction to generalized type systems. *Journal of Functional Programming*, 1(2):125–154, 1991. `doi:10.1017/S0956796800020025`.

5 Henk Barendregt, S. Abramsky, D. Gabbay, T. Maibaum, and Henk (Hendrik) Barendregt. Lambda Calculi with Types, 2000.

6 Stefano Berardi. *Type dependence and Constructive mathematics*. PhD thesis, PhD thesis, Dipartimento di Informatica, Torino, Italy, 1990.

7 Frédéric Blanqui. Encoding Type Universes Without Using Matching Modulo Associativity and Commutativity. In Amy P. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*, volume 228 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.FSCD.2022.24`.

8 Frédéric Blanqui, Gilles Dowek, Émilie Grienenberger, Gabriel Hondet, and François Thiré. Some Axioms for Mathematics. In Naoki Kobayashi, editor, *6th International Conference on Formal Structures for Computation and Deduction (FSCD 2021)*, volume 195 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.FSCD.2021.20`.

9 Mathieu Boespflug and Guillaume Burel. CoqInE: Translating the Calculus of Inductive Constructions into the λΠ-calculus Modulo. In *"Second International Workshop on Proof Exchange for Theorem Proving*, 2012.

10 Mathieu Boespflug, Quentin Carbonneaux, and Olivier Hermant. The λΠ-calculus Modulo as a Universal Proof Language. *CEUR Workshop Proceedings*, 878, June 2012. URL: `https://ceur-ws.org/Vol-878/paper2.pdf`.

11 Mario Carneiro. The Type Theory of Lean. Master's thesis, Carnegie Mellon University, 2019. URL: `https://github.com/digama0/lean-type-theory/releases`.

12 Thierry Coquand. An Analysis of Girard's Paradox. In *Proceedings of the First Annual IEEE Symposium on Logic in Computer Science (LICS 1986)*, pages 227–236. IEEE Computer Society Press, June 1986.

13 Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 76(2):95–120, 1988. `doi:10.1016/0890-5401(88)90005-3`.

14 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*, pages 115–130, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. `doi:10.1007/3-540-45685-6_9`.

15 Denis Cousineau and Gilles Dowek. Embedding Pure Type Systems in the Lambda-Pi-Calculus Modulo. In *Typed Lambda Calculi and Applications, 8th International Conference, TLCA 2007, Paris, France, June 26-28, 2007, Proceedings*, pages 102–117, June 2007. `doi:10.1007/978-3-540-73228-0_9`.

16 Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite Systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 243–320. Elsevier and MIT Press, 1990. `doi:10.1016/b978-0-444-88074-1.50011-1`.

17 Michael Färber. Safe, Fast, Concurrent Proof Checking for the Lambda-Pi Calculus modulo Rewriting. In *Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2022, pages 225–238, New York, NY, USA, 2022. Association for Computing Machinery. `doi:10.1145/3497775.3503683`.

18 Gaspard Férey. *Higher-Order Confluence and Universe Embedding in the Logical Framework. (Confluence d'ordre supérieur et encodage d'univers dans le Logical Framework)*. PhD thesis, École normale supérieure Paris-Saclay, France, 2021. URL: `https://lmf.cnrs.fr/downloads/Perso/Ferey-thesis.pdf`.

19 Guillaume Genestier. Encoding Agda Programs Using Rewriting. In Zena M. Ariola, editor, *5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020)*, volume 167 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.FSCD.2020.31`.

20 Girard, Jean-Yves. Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur, 1972.

21 Yoan Géran. Mathématiques inversées de Coq. Master's thesis, ENS Paris-Saclay, September 2021. URL: `https://inria.hal.science/hal-04319183`.

22 Yoan Géran. STT∀ GeoCoq, 2021. URL: `https://github.com/Karnaj/sttfa_geocoq_euclid`.

23 Robert Harper and Robert Pollack. Type Checking with Universes. In *2nd International Joint Conference on Theory and Practice of Software Development*, TAPSOFT '89, pages 107–136, NLD, 1991. Elsevier Science Publishers B. V. `doi:10.1016/0304-3975(90)90108-T`.

24 Gabriel Hondet and Frédéric Blanqui. The New Rewriting Engine of Dedukti (System Description). In Zena M. Ariola, editor, *5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020)*, volume 167 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.FSCD.2020.35`.

25 Per Martin-Löf. A theory of types, 1971. Preprint, Stockholm University.

26 Per Martin-Löf. An Intuitionistic Theory of Types: Predicative Part. *Studies in logic and the foundations of mathematics*, 80:73–118, 1975.

27 Per Martin-Löf. An intuitionistic theory of types. In Giovanni Sambin and Jan M. Smith, editors, *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 127–172. Oxford University Press, 1998.

28 Matthieu Sozeau and Nicolas Tabareau. Universe Polymorphism in Coq. In Gerwin Klein and Ruben Gamboa, editors, *Interactive Theorem Proving*, pages 499–514, Cham, 2014. Springer International Publishing. doi:10.1007/978-3-319-08970-6_32.

29 Terese. *Term rewriting systems*, volume 55 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 2003.

30 François Thiré. Sharing a Library between Proof Assistants: Reaching out to the HOL Family. In Frédéric Blanqui and Giselle Reis, editors, *Proceedings of the 13th International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice, LFMTP@FSCD 2018, Oxford, UK, 7th July 2018*, volume 274 of *EPTCS*, pages 57–71, 2018. doi:10.4204/EPTCS.274.5.

31 Vladimir Voevodsky. A universe polymorphic type system, October 2014. An unfinished unreleased manuscript. URL: https://www.math.ias.edu/Voevodsky/files/files-annotated/Dropbox/Unfinished_papers/Type_systems/UPTS_current/Universe_polymorphic_type_sytem.pdf.

A Computation Algorithm

We design a recursive algorithm suited to the inductive structure of $\mathbf{E}$. It is presented in Algorithm 1 which already contains the code for the base cases 0 and x for which the canonical form are respectively $\max(\emptyset)$ and $\max(\mathcal{V}(\{x\}, x, 0))$.

Algorithm 1 Canonization algorithm.

Input: $u \in \mathbf{E}$
Result: $c(u)$, the canonical form of u
Function normalize(u)
  if $u = 0$ **then return** $\max(\emptyset)$
  else if $u = x$ **then return** $\max(\mathcal{V}(\{x\}, x, 0))$
  else if $u = \max(u, v)$ **then** `// Algorithm 3`
  else if $u = S(u)$ **then** `// Algorithm 4`
  else if $u = \mathrm{imax}(u, v)$ **then** `// Algorithm 5`
  else if $u = \mathcal{C}(\{u_1, \ldots, u_n\}, k)$ **then** `// Algorithm 6`
  else if $u = \mathcal{V}(\{u_1, \ldots, u_n\}, v, k)$ **then** `// Algorithm 6`
end

We are now interested in the code to compute the canonical form in the other cases. The algorithm follows the induction performed in Section 4 to prove Theorem 54 since it gives us a representation for the different shapes of levels. Then, it is sufficient to minimize this representation to stick to Proposition 36. For that, we write Algorithm 2 which inserts a sublevel in an independent set of canonical sublevels.

Algorithm 2 Insertion algorithm.

Input: $U \subset \mathbf{S_C}$ independent, $v \in \mathbf{S_C}$
Result: W such that $c(\max(U \cup \{v\})) = \max(W)$
Function insert(v, U)
  $W \leftarrow \emptyset$
  for $u \in U$ **do**
    if $v \leqslant u$ **then return** U
    else if $u \not\leqslant v$ **then** $W \leftarrow W \cup \{u\}$
  end
  return $W \cup \{v\}$
end

The Maximum

To compute the canonical form of $\max(u, v)$, we use Algorithm 2 to insert the sublevels of v into the ones of u.

Algorithm 3 Case of the maximum.

Data: $u, v \in \mathbf{E}$
Result: $c(\max(u, v))$
$s \leftarrow \text{sublevels}(\text{normalize}(u))$
for $v_i \in \text{sublevels}(\text{normalize}(v))$ **do**
  $s \leftarrow \text{insert}(v_i, s)$
end
return $\max(s)$

The Successor

Thanks to Proposition 42, for all $\max U \in \mathbf{R_C}$, we know a representation of $S(\max U)$. To obtain its canonical form, we could use insert to add its sublevels to an initially empty set. But, we have a simpler operation.

▶ **Proposition 55.** *Let* $\max U \in \mathbf{R_C}$ *and* $E = \{\text{inc}(u) \mid u \in U\}$.

$$c(S(\max U)) = \begin{cases} \max E & \text{if } \exists u \in U, \text{VC}(u) = \emptyset \\ \max E \cup \{\mathcal{C}(\emptyset, 1)\} & \text{else} \end{cases}$$

We implement this strategy in Algorithm 4.

Algorithm 4 Case of the successor.

Data: $u \in \mathbf{E}$
Result: The canonical form of $S(u)$
$U \leftarrow \text{sublevels}(\text{normalize}(u))$
$s \leftarrow \{\text{inc}(u) \mid u \in U\}$
for $u \in U$ **do**
  if $\text{VC}(u) = \emptyset$ **then return** $\max(s)$
end
return $\max(s \cup \mathcal{C}(\emptyset, 1))$

The Impredicative Maximum

For all $u, v \in \mathbf{S_C}$, Proposition 45 expresses $\mathrm{imax}(u, v)$ as a maximum of canonical sublevels, and for all $\max U, \max V \in \mathbf{R_C}$, Proposition 44 expresses $\mathrm{imax}(\max U, \max V)$ as a maximum of $\mathrm{imax}(u, v)$ with $u \in U$ and $v \in V$ (hence $u, v \in \mathbf{S_C}$). Using these two results, we design Algorithm 5.

Algorithm 5 Case of the impredicative maximum.

Data: u and v levels
Result: The canonical form of $\mathrm{imax}(u, v)$
$U \leftarrow \mathrm{sublevels}(\mathrm{normalize}(u))$
$V \leftarrow \mathrm{sublevels}(\mathrm{normalize}(v))$
$s \leftarrow \mathrm{sublevels}(\mathrm{normalize}(v))$
for $u \in U, v \in V$ **do**
  if $u = \mathcal{V}(E, x, k)$ **then** $s \leftarrow \mathrm{insert}(\mathcal{V}(E \cup \mathrm{VC}(v), x, k), s)$
  else if $u = \mathcal{C}(E, k)$ **then** $s \leftarrow \mathrm{insert}(\mathcal{C}(E \cup \mathrm{VC}(v), k), s)$
end
return $\max(s)$

The Constant Sublevels

The computation of the canonical form of a constant sublevel relies on Proposition 47. Here, we immediately returns $\max(\emptyset)$ if some VC is 0, and we do not forget the case $k = 0$ which results in 0.

Algorithm 6 Case of the constant sublevels.

Data: $u_1, \ldots, u_n \in \mathbf{E}$, $k \in \mathbb{N}$
Result: $c(\mathcal{C}(\{u_1, \ldots, u_n\}, k))$
if $k = 0$ **then return** $\max(\emptyset)$
for $1 \leqslant i \leqslant n$ **do**
  $U_i \leftarrow \mathrm{sublevels}(\mathrm{normalize}(u_i))$
  if $U_i = \emptyset$ **then return** $\max(\emptyset)$
end
$s \leftarrow \emptyset$
for $u_1 \in U_1, \ldots, u_n \in U_n$ **do**
  $s \leftarrow \mathrm{insert}\left(\mathcal{C}\left(\bigcup_{1 \leqslant i \leqslant n} \mathrm{VC}(u_i), k\right), s\right)$
end
return $\max(s)$

The Variable Sublevels

The case of the variable sublevel is very similar and relies on Proposition 52.

Algorithm 7 Case of the variable sublevels.

Data: $u_1, \dots, u_n, v \in \mathbf{E}$, $k \in \mathbb{N}$
Result: $c(\mathcal{V}(\{u_1, \dots, u_n\}, v, k))$
for $1 \leqslant i \leqslant n$ **do**
  $u_i \leftarrow \text{sublevels}(\text{normalize}(u_i))$
  if $U_i = \emptyset$ **then return** $\max(\emptyset)$
end
$V \leftarrow \text{sublevels}(\text{normalize}(v))$
$s \leftarrow \emptyset$
for $u_1 \in U_1, \dots, u_n \in U_n$ **do**
  if $k \neq 0$ **then** $s \leftarrow \text{insert}(\mathcal{C}(\cup_{1 \leqslant i \leqslant n} \text{VC}(u_i), k), s)$
  for $u_0 \in V$ **do**
    if $u_0 = \mathcal{C}(E, l)$ **then** $s \leftarrow \text{insert}(\mathcal{C}(\cup_{0 \leqslant i \leqslant n} \text{VC}(u_i), k + l), s)$
    else if $u_0 = \mathcal{V}(E, x, l)$ **then** $s \leftarrow \text{insert}(\mathcal{V}(\cup_{0 \leqslant i \leqslant n} \text{VC}(u_i), x, k + l), s)$
  end
end
return $\max(s)$

▶ **Theorem 56** (Correction). *Let $u \in \mathbf{E}$. Then,* normalize(u) *computes $c(u)$, the canonical form of u.*