Encoding impredicative hierarchy of type universes with variables

Yoan Géran

Encoding impredicative hierarchy of type universes with variables

Yoan Géran
Mines Paris - PSL, Centre de Recherche en Informatique
Université Paris-Saclay, Laboratoire Méthodes Formelles, ENS Paris-Saclay

Abstract

Logical frameworks can be used to translate proofs from a proof system to another one. For this purpose, we should be able to encode the theory of the proof system in the logical framework. The Lambda Pi calculus modulo theory is one of these logical frameworks. Powerful theories such as pure type systems with an infinite hierarchy of universes have been encoded, leading to partial encodings of proof systems such as Coq, Matita or Agda. In order to fully represent systems such as Coq and Lean, we introduce a representation of an infinite universe hierarchy with an impredicative universe and universe variables where universe equivalence is equality, and implement it as a terminating and confluent rewrite system.

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

Keywords and phrases type theory, logical framework, rewriting theory, type universes

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Supplementary Material An implementation is available at `https://gitlab.crans.org/geran/dedukti-level-implementation`

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 softwares are done in several tools, and many logical systems and theories are developed as the research on proof-checking makes progress. Interoperability is then a big challenge which aims to avoid the redevelopment of the same proof in each system. Instead of developing translators from each system to another one, *logical frameworks* propose to define theories in a common language, which makes translation easier. Thus, the logical framework should be expressive enough and work should be done to define the wanted theories in the framework.

In this paper, our goal is to show how to define the universe levels of the theory of the Coq proof system in one of these frameworks, the $\lambda\Pi$-calculus modulo rewriting. Since a lot of theories are expressed as extensions of *Pure Type Systems*, the first part of this introduction will define them. Then, we will present the $\lambda\Pi$-calculus modulo rewriting and the type system behind Coq, in particular the universe levels.

Pure Type Systems

A lot of theories are based on extensions of Church's simply-typed λ-calculus (STLC). In [5], Barendregt introduced the λ-cube which classifies type systems depending on the possibility to quantify on types or terms to build new types or new terms. It captures systems such as System F (with type polymorphism), $\lambda\underline{\omega}$ (with type operators), $\lambda\Pi$ (with dependent types), and the Calculus of Constructions (CC) which allows all these quantifications.

More generally, these constructions can be extended, leading to more powerful systems called *Pure Type Systems* [6, 8].

▶ **Definition 1.** *A Pure Type System (PTS) is defined by a set of sorts $\mathcal{S}$ (that we will also call 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 is an element of a countable set of variables $\mathcal{X}$.

$$t \coloneqq s \mid x \mid \Pi x\colon t \cdot t \mid (\lambda x\colon t \cdot t) \mid t\ t$$

and the typing rules are given in Figure 1.

$$(\textsc{Empty})\ \frac{}{[\,]\ \mathrm{WF}} \qquad (\textsc{Decl})\ \frac{\Gamma \vdash A\colon s \qquad x \notin \Gamma}{\Gamma, x\colon A\ \mathrm{WF}} \qquad (\textsc{Var})\ \frac{\Gamma\ \mathrm{WF} \qquad (x\colon A) \in \Gamma}{\Gamma \vdash x\colon A}$$

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

$$(\textsc{App})\ \frac{\Gamma \vdash t\colon \Pi x\colon A \cdot B \qquad \Gamma, \vdash u\colon A}{\Gamma \vdash t\ u\colon B[x \coloneqq u]} \qquad (\textsc{Abs})\ \frac{\Gamma, x\colon A \vdash t\colon B \qquad \Gamma \vdash \Pi x\colon A \cdot B\colon s}{\Gamma \vdash \lambda x \cdot t\colon \Pi x\colon A \cdot B}$$

$$(\textsc{Conv})\ \frac{\Gamma \vdash B\colon s \qquad \Gamma \vdash t\colon A \qquad A \equiv_\beta B}{\Gamma \vdash t\colon B}\ s \in \mathcal{S}$$

Figure 1 Typing rules

▶ **Definition 2** (Functional and full PTS)**.** *A PTS is said to be functional if $\mathcal{A}$ and $\mathcal{R}$ are functional relations from $\mathcal{S}$ and $\mathcal{S} \times \mathcal{S}$ to $\mathcal{S}$, that is to say $(s_1, s_2) \in \mathcal{A} \wedge (s_1, s_3) \in \mathcal{A} \implies s_2 = s_3$ and $(s_1, s_2, s_3) \in \mathcal{R} \wedge (s_1, s_2, s_4) \in \mathcal{R} \implies s_3 = s_4$.*

A PTS is called full if $\mathcal{A}$ and $\mathcal{R}$ are total functions from $\mathcal{S}$ and $\mathcal{S} \times \mathcal{S}$ to $\mathcal{S}$.

The $\lambda\Pi$-calculus modulo rewriting

$\lambda\Pi$, the extension of STLC with dependent types, is the language of the *Edinburgh Logical Framework* (ELF) [22]. However, computation plays an essential role in type theories, then in modern proof assistant, and $\lambda\Pi$ is not well-suited for this. To address this point, the $\lambda\Pi$-calculus modulo rewriting ($\lambda\Pi/\equiv$) [12] extends $\lambda\Pi$ by allowing user-defined higher-order rewrite rules [16, 30] that can be used to define functions but also types. Types are then identified modulo β and these rewrite rules.

In order to have good properties such as the decidability of the type-checking, the rewrite rules introduced should preserve typing and form a confluent and strongly normalizing rewrite system, which adds some restrictions and requires more efforts to show that these properties are respected.

▶ Remark 3. In the rest of this article, we will use the syntax $u \longrightarrow v$ (where u may contains free variables used for matching) to define a rewrite rule and the syntax $u \hookrightarrow v$ to indicate that the term u rewrites to the term v.

The $\lambda\Pi/\equiv$ can express CC and its subtheories [10], and, in [15], Cousineau and Dowek show how to embed functional PTS (with a possibly infinite number of symbols and rules):

1. for each sort s, symbols $\mathtt{U_s}\colon \mathtt{Type}$ and $\mathtt{El_s}\colon \mathtt{U_s} \to \mathrm{Type}$,
2. for each axiom (s_1, s_2), a symbol $\mathtt{u}_{s_1}\colon \mathtt{U}_{s_2}$ and a rewrite rule $\mathtt{El}_{s_2}(s_1) \longrightarrow \mathtt{U}_{s_1}$,

3. for each rule (s_1, s_2, s_3), a symbol $\pi_{(\mathtt{s}_1,\mathtt{s}_2)} : (x : \mathtt{U}_{\mathtt{s}_1}) \to (\mathtt{El}_{\mathtt{s}_1}(x) \to \mathtt{U}_{\mathtt{s}_2}) \to \mathtt{U}_{\mathtt{s}_3}$ and a rewrite rule $\mathtt{El}_{\mathtt{s}_3}(\pi_{(\mathtt{s}_1,\mathtt{s}_2)}(A, B)) \longrightarrow (x : \mathtt{El}_{\mathtt{s}_1}(A)) \to \mathtt{El}_{\mathtt{s}_2}(B)$.

$\mathtt{u}_{\mathtt{s}}$ corresponds to the sort s as a term, $\mathtt{U}_{\mathtt{s}}$ to the sort s as a type, and $\mathtt{El}_{\mathtt{s}}$ associates a sort (as term) of type s to its corresponding type, hence the rewrite rule added for each axiom, and $\pi_{(\mathtt{s}_1,\mathtt{s}_2)}AB$ is the term corresponding to the types of the function from A to $B(A)$, hence the rewrite rule added to obtain the type associated to this term.

Several systems have been encoded in $\lambda\Pi/\equiv$: HOL-Light [31, 2], Agda [20], Matita [2], but also parts of Coq, on which we will come back to later. Besides, since there exist multiple implementations of the $\lambda\Pi/\equiv$ such as Dedukti [3], Lambdapi [24], or Kontroli [18], these embeddings have been effectively implemented leading to translations from the proofs systems to these implementations, but also to translations from these implementations of $\lambda\Pi/\equiv$ to proof assistants [31, 32].

Coq's type system

The theory of Coq is based on CC extended with an infinite hierarchy of universes and an impredicative universe Prop. It corresponds to a slightly different version of the following PTS (where Prop is denoted as Type_0).

▶ **Definition 4** (Impredicative max). *We define* $\mathrm{imax} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ *by* $\mathrm{imax}(i, 0) = 0$ *and* $\mathrm{imax}(i, j + 1) = \max(i, j + 1)$.

▶ **Definition 5** (CC^∞). *CC^∞ is the full PTS defined with an infinite sequence of sorts* Type_i *indexed on* $\mathbb{N}$*, the axioms* $\left(\mathrm{Type}_i, \mathrm{Type}_{i+1}\right)$*, and the rules* $\left(\mathrm{Type}_i, \mathrm{Type}_j, \mathrm{Type}_{\mathrm{imax}(i,j)}\right)$.

▶ Remark 6. One can also define a predicative PTS where the products from Type_i to Type_j are elements of $\mathrm{Type}_{\max(i,j)}$ instead of $\mathrm{Type}_{\mathrm{imax}(i,j)}$. This latter is the building on Agda proof system while CC^∞ is the one of Coq but also of Lean or Matita.

In both cases, the sorts are characterized by *levels* indexed on $\mathbb{N}$; the functions $\mathcal{R}$ and $\mathcal{A}$ can be defined in the $\lambda\Pi/\equiv$, and we can adapt the general embedding of Cousineau and Dowek to a finite embedding. For that, we define the natural numbers $\mathtt{N}$ with the successor function $\mathtt{s}$, $\mathtt{max} : \mathtt{N} \to \mathtt{N} \to \mathtt{N}$, $\mathtt{imax} : \mathtt{N} \to \mathtt{N} \to \mathtt{N}$, and

- symbols $\mathtt{U} : \mathtt{N} \to \mathtt{Type}$ and $\mathtt{El} : (i : \mathtt{N}) \to \mathtt{U}(i) \to \mathrm{Type}$,
- a symbol $\mathtt{u} : (i : \mathtt{N}) \to \mathtt{U}(\mathtt{s}(i))$ and a rewrite rule $\mathtt{El}\ \_\ i \longrightarrow \mathtt{U}\ i$,
- a symbol $\pi : (i : \mathtt{N}) \to (j : \mathtt{N}) \to (A : \mathtt{U}\ i) \to (\mathtt{El}\ i\ A \to \mathtt{U}\ j) : \mathtt{U}\ (\mathtt{imax}\ i\ j)$ and a rewrite rule $\mathtt{El}\ \_\ (\pi\ i\ j\ A\ B) \longrightarrow (x : \mathtt{El}\ i\ A) \to \mathtt{El}\ j\ (B\ x)$.

Coq extend CC^∞ with other features. Some of them have been encoded, leading to a partial translator from Coq to Dedukti [11, 17], and to the sharing of developements of the GeoCoq library [7], a formalization of geometry, to other proof assistants [21]. Extensions such as inductive types [13, 28] and cumulativity [27] have been widely covered: Burel and Boespflug in [11], then Férey in his thesis [17] proposes embeddings of inductive constructions and cumulativity have been studied by Assaf [1, 2], Férey [17] and Thiré [32].

In this paper, we are interested in another feature, the level variables, which permits to extend CC^∞ with floating universes [25] or with universe polymorphism [29, 23, 14].

Level variables

We extend the syntax of the levels with variables.

▶ **Definition 7** (Levels). *A level is a term of the grammar*

$$t \coloneqq 0 \mid s(t) \mid \max(t,t) \mid \operatorname{imax}(t,t) \mid x$$

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

▶ **Definition 8** (Valuation). *A valuation is a function $\sigma\colon \mathcal{X} \to \mathbb{N}$.*

▶ **Definition 9.** *Let $\sigma\colon \mathcal{X} \to \mathbb{N}$ be a a valuation. We define inductively the value of a level t over σ, denoted as $[\![t]\!]_\sigma$ with*

$$[\![0]\!]_\sigma = 0 \qquad [\![s(t)]\!]_\sigma = s([\![t]\!]_\sigma) \qquad [\![x]\!]_\sigma = \sigma(x)$$
$$[\![\max(t_1,t_2)]\!]_\sigma = \max\big([\![t_1]\!]_\sigma, [\![t_2]\!]_\sigma\big) \qquad [\![\operatorname{imax}(t_1,t_2)]\!]_\sigma = \operatorname{imax}\big([\![t_1]\!]_\sigma, [\![t_2]\!]_\sigma\big)$$

We use the same symbol s, max, and imax for the syntax of the levels and the functions of the natural numbers. However, levels are abstract terms and are interpreted through valuations. Besides, the concrete levels can clearly be identified as the natural numbers which justifies the use of the same symbol and permits to see the interpretation as a function that *concretizes* a level, turning it into a concrete level.

▶ **Definition 10** (Level comparison). *Let $t_1, t_2 \in \mathcal{L}$. We say that $t_1 \leqslant_{\mathcal{L}} t_2$ if for all valuations σ, $[\![t_1]\!]_\sigma \leqslant [\![t_2]\!]_\sigma$. In the same way, we say that $t_1 =_{\mathcal{L}} t_2$ if for all valuations σ, $[\![t_1]\!]_\sigma = [\![t_2]\!]_\sigma$. Hence $t_1 =_{\mathcal{L}} t_2$ if and only if $t_1 \leqslant_{\mathcal{L}} t_2$ and $t_2 \leqslant_{\mathcal{L}} t_1$.*

With level variables, the equivalence is no more the syntactic equality and the above embedding does not reflect it anymore: $\texttt{max}(x,y)$ and $\texttt{max}(y,x)$ are not convertible and adding rules for that would lead to a non-terminating system. In the same way, commutativity and equivalences such as $\max(x,x) =_{\mathcal{L}} x$ or $\max(s(x),x) =_{\mathcal{L}} s(x)$ are hard to express, and the impredicativity introduces other:$\operatorname{imax}(x,x) =_{\mathcal{L}} x$, $\max(\operatorname{imax}(x,y),x) =_{\mathcal{L}} \max(x,y)$, etc.

And yet, a correct embedding of the levels should reflect these equivalences. For instance, a term of the universe Type_x is also a term of the universe $\mathrm{Type}_{\operatorname{imax}(x,x)}$, and then we should be able to identify the universes such as Type_x and $\mathrm{Type}_{\operatorname{imax}(x,x)}$. This paper presents a new embedding that faithfully represents levels with variables.

Related work

Some solutions have been studied in the predicative case. The big issue is the associativity and commutativity of the max symbol. In [20], Genestier solved this problem to encode Agda's universe polymorphism. For that, he used rewriting modulo associativity and commutativity (AC). The idea, also mentionned in a draft of Voevodsky [33], is to represent each level as $\max(n, n_1 + x_1, \ldots, n_k + x_k)$ where $n \geqslant \max(n_1, \ldots, n_k)$. Besides, if there exists $i \neq j$ such that $x_j = x_i$, we simplify the term and keep only $\max(n_i, n_j) + x_i$. Then, we obtain a minimal representation of terms of the max-successor algebra.

Blanqui gives another presentation of this algebra in [9], with an encoding without matching modulo AC. However, this solution requires to keep the level in some AC canonical form, and can then require the modification of the $\lambda\Pi/\equiv$ type-checker.

The imax-successor algebra is less studied and we do not know easy ways to reflect its equalities. A confluent encoding is proposed in [4], but it does not fully reflect the equalities; for instance, the levels $\max(\operatorname{imax}(x,y),x)$ and $\max(x,y)$ are not convertible. Besides, Férey designed a non-confluent encoding of universe polymorphism in [17].

Contribution and outline

We introduce a new representation for the levels, using the idea presented above in the predicative case: find a set of subterms such that any level can be expressed as a maximum of subterms. They should be easily comparable to simplify $\max(u, v)$ into u if $u \leqslant_{\mathcal{L}} v$ and obtain minimal representations, and they should ensure the uniqueness property:

$$\max(u_1, \ldots, u_n) =_{\mathcal{L}} \max(v_1, \ldots, v_m) \iff \{u_1, \ldots, u_n\} = \{v_1, \ldots, v_m\}.$$

Intuitively, the subterms should be very basic and simple: a subterm u must not be equivalent to a maximum of other subterms. With this representation, we obtain a deep understanding of the imax-successor algebra, and an easy procedure-decision for the level inequality problem.

In the Section 2, we study the semantic of the imax operator and establish a suitable set of subterms. Then, in the Section 3, we introduce the minimal representation and show that equivalent terms have the same minimal representation. And the Section 4 is dedicated to the implementation of this representation into the $\lambda\Pi/\equiv$ as a first-order confluent and terminating rewrite system. An implementation in Dedukti is available on `https://gitlab.crans.org/geran/dedukti-level-implementation`.

2 Universe representation in impredicative hierarchy

In this section, we study the imax operators and its interaction with max and the successor and establish semantic equalities that permits to simplify the levels in order to find a set of sublevels for the desired representation.

2.1 Levels as maximum

The very first step is to show that any level can be expressed as a maximum of levels that do not contain any max, that is the principle of our idea of representation. The succesor can be distributed over max, the two next propositions show how to distribute imax over max.

▶ **Proposition 11.** *For all* $u, v, w \in \mathcal{L}$, $\mathrm{imax}(u, \max(v, w)) =_{\mathcal{L}} \max(\mathrm{imax}(u, v), \mathrm{imax}(u, w))$.

Proof. Let σ be a valuation, $t = \mathrm{imax}(u, \max(v, w))$, $t_1 = \mathrm{imax}(u, v)$ and $t_2 = \mathrm{imax}(u, w)$.

- If $[\![v]\!]_\sigma = [\![w]\!]_\sigma = 0$, then $[\![\max(t_1, t_2)]\!]_\sigma = 0 = [\![t]\!]_\sigma$.
- If $[\![v]\!]_\sigma \neq 0$ and $[\![w]\!]_\sigma = 0$, then $[\![\max(t_1, t_2)]\!]_\sigma = \max([\![u]\!]_\sigma, [\![v]\!]_\sigma) = [\![t]\!]_\sigma$.
- If $[\![v]\!]_\sigma = 0$ and $[\![w]\!]_\sigma \neq 0$, then $[\![\max(t_1, t_2)]\!]_\sigma = \max([\![u]\!]_\sigma, [\![w]\!]_\sigma) = [\![t]\!]_\sigma$.
- Else, $[\![\max(t_1, t_2)]\!]_\sigma = \max([\![u]\!]_\sigma, [\![v]\!]_\sigma, [\![w]\!]_\sigma) = [\![t]\!]_\sigma$.

◀

▶ **Proposition 12.** *For all* $u, v, w \in \mathcal{L}$, $\mathrm{imax}(\max(u, v), w) =_{\mathcal{L}} \max(\mathrm{imax}(u, w), \mathrm{imax}(v, w))$.

Proof. Let σ be a valuation.

- If $[\![w]\!]_\sigma = 0$, then $[\![\max(\mathrm{imax}(u, w), \mathrm{imax}(u, w))]\!]_\sigma = 0 = [\![\mathrm{imax}(\max(u, v), w)]\!]_\sigma$.
- Else, $[\![\max(\mathrm{imax}(u, w), \mathrm{imax}(v, w))]\!]_\sigma = \max([\![u]\!]_\sigma, [\![v]\!]_\sigma, [\![w]\!]_\sigma) = [\![\mathrm{imax}(\max(u, v), w)]\!]_\sigma$.

◀

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

▶ **Theorem 13.** *For all* $t \in \mathcal{L}$, *there exists* $u_1, \ldots, u_n$ *in the grammar* $t \coloneqq 0 \mid s(t) \mid \mathrm{imax}(t, t) \mid x$ *such that* $t =_{\mathcal{L}} \max(u_1, \ldots, u_n)$.

2.2 Simplification of the levels

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

The main issue is imax: its asymmetry complicates its interaction with other symbols. The previous equalities show how to remove the interaction between imax and max, now, we will study the interactions between imax and the other symbols. The goal is to restrict the localisation of the imax symbols to specific parts of the levels in order to understand and control their influence on the levels semantic.

Firstly, we recall these equalities that are direct consequences of the semantic of imax. They permit to deal with 0 and the successor.

▶ **Proposition 14.** *For all* $u, v \in \mathcal{L}$, $\mathrm{imax}(u, 0) =_{\mathcal{L}} 0$ *and* $\mathrm{imax}(u, s(v)) = \max(u, s(v))$.

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

▶ **Proposition 15.** *For all* $u, v, w \in \mathcal{L}$, $\mathrm{imax}(u, \mathrm{imax}(v, w)) =_{\mathcal{L}} \max(\mathrm{imax}(u, w), \mathrm{imax}(v, w))$.

Proof. Let σ be a valuation.

- If $[\![w]\!]_\sigma = 0$, then $[\![\max(\mathrm{imax}(u, w), \mathrm{imax}(u, w))]\!]_\sigma = 0 = [\![\mathrm{imax}(u, \mathrm{imax}(v, w))]\!]_\sigma$.
- Else, $[\![\max(\mathrm{imax}(u, w), \mathrm{imax}(v, w))]\!]_\sigma = \max([\![u]\!]_\sigma, [\![v]\!]_\sigma, [\![w]\!]_\sigma) = [\![\mathrm{imax}(u, \mathrm{imax}(v, w))]\!]_\sigma$.

◀

Thus, we can consider that the second argument of a imax is always a variable. It is more complicated to directly enforce the form of its first argument, but we can obtain one restriction by distributing the successor over the imax. However, we cannot do it as directly as we distribute the successor over the max, as shown in the next example.

▶ **Example 16.** We show that $s(\mathrm{imax}(y, x)) \neq_{\mathcal{L}} \mathrm{imax}(s(y), s(x))$ by considering a valuation σ such that $\sigma(x) = 0$ and $\sigma(y) = 1$.

▶ **Proposition 17.** *For all* $u, v, w \in \mathcal{L}$, $s(\mathrm{imax}(v, w)) =_{\mathcal{L}} \max(s(w), \mathrm{imax}(s(v), w))$.

Proof. Let σ be a valuation.

- If $[\![w]\!]_\sigma = 0$, then $[\![s(\mathrm{imax}(v, w))]\!]_\sigma = s(0) = [\![\max(s(w), \mathrm{imax}(s(v), w))]\!]_\sigma$.
- Else $[\![s(\mathrm{imax}(v, w))]\!]_\sigma = s(\max([\![v]\!]_\sigma, [\![w]\!]_\sigma)) = [\![\max(s(w), \mathrm{imax}(s(v), w))]\!]_\sigma$.

◀

Finally, all of these propositions lead to this grammar restriction.

▶ **Theorem 18.** *For all* $u \in \mathcal{L}$, *there exists* $u_1, \ldots, u_n$ *in the grammar* $t \coloneqq s^k(x) \mid s^k(0) \mid \mathrm{imax}(t, x)$ *such that* $u =_{\mathcal{L}} \max(u_1, \ldots, u_n)$.

▶ Remark 19. For all t in the grammar of Theorem 18, there exists $x_1, \ldots, x_n \in \mathcal{X}$, and $v = s^k(0)$ or $v = s^k(x)$ such that $t = \mathrm{imax}(\mathrm{imax}(\mathrm{imax}(\cdots \mathrm{imax}(v, x_1), x_2) \cdots), x_{n-1}), x_n)$.

2.3 Introducting new levels

Here, we continue the simplification of the levels under max in order to find simple enough terms to reach the uniqueness property. Indeed, the terms of the grammar of Theorem 18 are still not enough.

▶ **Example 20.** Let us consider $t = \max(\mathrm{imax}(x, y), \mathrm{imax}(y, x))$. Then, $t =_{\mathcal{L}} \max(x, y)$.

To better understand the issue, we should have a deep understanding of the semantic of $\mathrm{imax}(x, y)$. It means that we always consider the value of y, but we only consider the value of x if y is not zero: the position of the variables is essential in the levels. But, taking into account $\mathrm{imax}(y, x)$ leads to offset this, and the value of x and y are always considered.

Here, we come to a solution that consists of introducing new symbols that permit to represent the current levels, without taking into account the order of the variables.

▶ **Definition 21** (Sublevels). *We define new symbols $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{A}$ takes as argument a set of variables, a variable and an integer, and $\mathcal{B}$ takes as argument a set of variables and an integer. They have the following semantic.*

$$[\![\mathcal{A}(E, x, S)]\!]_\sigma = \begin{cases} 0 & \text{if there exists } y \in E \text{ such that } \sigma(y) = 0 \\ [\![x]\!]_\sigma + S & \text{else} \end{cases}$$

$$[\![\mathcal{B}(E, S)]\!]_\sigma = \begin{cases} 0 & \text{if there exists } y \in E \text{ such that } \sigma(y) = 0 \\ S & \text{else} \end{cases}$$

In the next proposition, we show that any level of the grammar of Theorem 18 can be written as a maximum of terms of this new grammar.

▶ **Proposition 22.** *If* $t = \mathrm{imax}\big(\mathrm{imax}\big(\mathrm{imax}\big(\cdots \mathrm{imax}\big(s^k(y), x_1\big)\cdots\big)\big), x_{n-1}\big), x_n\big)$, *then*

$$t =_{\mathcal{L}} \max(\mathcal{A}(\emptyset, x_n, 0), \mathcal{A}(\{x_n\}, x_{n-1}, 0), \ldots, \mathcal{A}(\{x_2, \ldots, x_n\}, x_1, 0), \mathcal{A}(\{x_1, \ldots, x_n\}, y, k)).$$

And if $t = \mathrm{imax}\big(\mathrm{imax}\big(\mathrm{imax}\big(\cdots \mathrm{imax}\big(s^k(0), x_1\big)\cdots\big)\big), x_{n-1}\big), x_n\big)$, *then*

$$t =_{\mathcal{L}} \max(\mathcal{A}(\emptyset, x_n, 0), \mathcal{A}(\{x_n\}, x_{n-1}, 0), \ldots, \mathcal{A}(\{x_2, \ldots, x_n\}, x_1, 0), \mathcal{B}(\{x_1, \ldots, x_n\}, k)).$$

Proof. We show the result for the first case. Let σ be a valuation. If forall $1 \leqslant i \leqslant n$, $\sigma(x_i) \neq 0$, then

$$[\![t]\!]_\sigma = \max(\sigma(x_n), \ldots, \sigma(x_1), k + \sigma(y))$$
$$\forall 1 < i \leqslant n, [\![\mathcal{A}(\{x_n, \ldots, x_{i+1}\}, x_i, 0)]\!]_\sigma = \sigma(x_i)$$
$$[\![\mathcal{A}(\{x_n, \ldots, x_1\}, y, k)]\!]_\sigma = k + \sigma(y)$$

and else, we take the largest $1 \leqslant i \leqslant n$ such that $\sigma(x_i) = 0$, then

$$[\![t]\!]_\sigma = \max(\sigma(x_n), \ldots, \sigma(x_{i+1})) \qquad \forall 1 < j \leqslant i, [\![\mathcal{A}(\{x_n, \ldots, x_{j+1}\}, x_j, 0)]\!]_\sigma = 0$$
$$[\![\mathcal{A}(\{x_n, \ldots, x_1\}, y, k)]\!]_\sigma = 0 \qquad \forall i < j \leqslant n, [\![\mathcal{A}(\{x_n, \ldots, x_{j+1}\}, x_j, 0)]\!]_\sigma = \sigma(x_j)$$

hence the equality.

And the second case is very similar. Let σ be a valuation. If forall $1 \leqslant i \leqslant n$, $\sigma(x_i) \neq 0$, then

$$[\![t]\!]_\sigma = \max(\sigma(x_n), \ldots, \sigma(x_1), k)$$
$$\forall 1 < i \leqslant n, [\![\mathcal{A}(\{x_n, \ldots, x_{i+1}\}, x_i, 0)]\!]_\sigma = \sigma(x_i)$$
$$[\![\mathcal{B}(\{x_n, \ldots, x_1\}, k)]\!]_\sigma = k$$

and else, we take the largest $1 \leqslant i \leqslant n$ such that $\sigma(x_i) = 0$, then

$$[\![t]\!]_\sigma = \max(\sigma(x_n), \ldots, \sigma(x_{i+1})) \qquad \forall 1 < j \leqslant i, [\![\mathcal{A}(\{x_n, \ldots, x_{j+1}\}, x_j, 0)]\!]_\sigma = 0$$
$$[\![\mathcal{B}(\{x_n, \ldots, x_1\}, k)]\!]_\sigma = 0 \qquad \forall i < j \leqslant n, [\![\mathcal{A}(\{x_n, \ldots, x_{j+1}\}, x_j, 0)]\!]_\sigma = \sigma(x_j)$$

hence the equality. ◀

The two equivalences are quite similar, the difference being in the very last subterm of the max which is a $\mathcal{A}$ in the first case (we have to take into account $s^k(y)$ that is to say $k+y$) and a $\mathcal{B}$ in the second one (k is taken into account with a $\mathcal{B}$).

The sublevels that we search for our representation are elements of this grammar. We just have two slightly modifications, two simplifications to make. The first one is illustrated by this example.

▶ **Example 23.** With $t_1 = \mathcal{A}(\emptyset, x, 0)$ and $t_2 = \mathcal{A}(\{x\}, x, 0)$, we have $t_1 =_{\mathcal{L}} t_2$ since for all valuation σ, $[\![t_1]\!]_\sigma = \sigma(x) = [\![t_2]\!]_\sigma$.

The issue here is the fact that the second argument of a $\mathcal{A}$ symbol does not necesarily appear in its first argument. This creates equalities as shown in the next proposition.

▶ **Proposition 24.** *Let $x \in \mathcal{X}$, $E \subset \mathcal{X} \setminus \{x\}$ and $S \in \mathbb{N}$. Then*

$$\mathcal{A}(E, x, S) =_{\mathcal{L}} \max(\mathcal{A}(E \cup \{x\}, x, S), \mathcal{B}(E, S)).$$

Proof. Let σ be a valuation, $t = \mathcal{A}(E, x, S)$, $u = \mathcal{A}(E \cup \{x\}, x, S)$ and $v = \mathcal{B}(E, S)$.

- If there exists $y \in E$ such that $\sigma(y) = 0$, then $[\![t]\!]_\sigma = [\![u]\!]_\sigma = [\![v]\!]_\sigma = 0$.
- Else, if $\sigma(x) = 0$, then $[\![t]\!]_\sigma = S$, $[\![u]\!]_\sigma = 0$ and $[\![v]\!]_\sigma = S$.
- Else, $\sigma(x) \neq 0$, and then $[\![t]\!]_\sigma = \sigma(x) + S$, $[\![u]\!]_\sigma = \sigma(x) + S$ and $[\![v]\!]_\sigma = S$.

Hence the result. ◀

Then, we will only consider elements of the form $\mathcal{A}(E, x, S)$ such that $x \in E$.

The second modification is related to the representation of 0. Indeed, for all $E \subset \mathcal{X}$, $\mathcal{B}(E, 0) =_{\mathcal{L}} 0$, and we should then keep at most one of them. However, since we already have $0 =_{\mathcal{L}} \max(\emptyset)$, we can remove all of them.

And we end up with this set of sublevels which permits to express any level of $\mathcal{L}$.

▶ **Definition 25.** *We denote by $\mathcal{L}_s$ the set of the sublevels that check the following conditions.*

1. $\mathcal{A}(E, x, S) \in \mathcal{L}_s \iff x \in E$,
2. $\mathcal{B}(E, S) \in \mathcal{L}_s \iff S > 0$.

▶ **Theorem 26.** *Let $t \in \mathcal{L}$. Then there exists $u_1, \ldots, u_n \in \mathcal{L}_s$ such that $t =_{\mathcal{L}} \max(u_1, \ldots, u_n)$.*

▶ Remark 27. By convenience, we will note $u_i \in t$ if there exists $u_1, \ldots, u_n \in \mathcal{L}_s$ such that $t =_{\mathcal{L}} \max(u_1, \ldots, u_n)$.

3 A minimal representation

In the previous section, we find a set $\mathcal{L}_s$ and showed that any level can be represented as a maximum of elements of $\mathcal{L}_s$. The goal of this one is to show that any level has a minimal representation as maximum of elements of $\mathcal{L}_s$ and that this representation is unique. Intuitevely, the sublevels of a minimal representation should be incomparable (else one of the sublevels could be removed).

▶ **Definition 28** (Minimal representation)**.** *Let t be a term. We say that t is a minimal representation if and only if there exists $u_1 \ldots, u_n \in \mathcal{L}_s$ such that*

1. $t = \max(u_1, \ldots, u_n)$,
2. *forall $i \neq j$, u_i and u_j are incomparable.*

We denote by $\mathcal{L}_r$ the set of the minimal representations.

Of course, any level t has a minimal representation. We can express t as a maximum of elements of $\mathcal{L}_{\text{s}}$ (by Theorem 26) and we remove elements if they are comparabe. The challenging part is its uniqueness. To show it, we study the core of the definition of a minimal representation: the sublevel comparison.

▶ **Theorem 29** (Sublevels comparison). *Elements of $\mathcal{L}_s$ are compared as follows.*

$$\mathcal{A}(E,x,S) \not\leqslant_{\mathcal{L}} \mathcal{B}(F,K) \tag{1}$$

$$\mathcal{B}(E,S) \leqslant_{\mathcal{L}} \mathcal{B}(F,K) \iff F \subset E \wedge S \leqslant K \tag{2}$$

$$\mathcal{B}(E,S) \leqslant_{\mathcal{L}} \mathcal{A}(F,x,K) \iff (F \subset E \wedge S \leqslant K+1) \tag{3}$$

$$\mathcal{A}(E,x,S) \leqslant_{\mathcal{L}} \mathcal{A}(F,y,K) \iff F \subset E \wedge x = y \wedge S \leqslant K \tag{4}$$

Proof. With σ such that $\sigma(x) = K+1$ and $\sigma(y) = 1$ if $y \neq x$, we show the first case. Indeed, $[\![\mathcal{A}(E,x,S)]\!]_\sigma = K+1+S > K = [\![\mathcal{B}(F,K)]\!]_\sigma$ hence $\mathcal{A}(E,x,S) \not\leqslant_{\mathcal{L}} \mathcal{B}(F,K)$. The cases 2, 3 and 4 correspond to Propositions 30–32 proved below. ◀

▶ **Proposition 30.** *Let $E,F \subset \mathcal{X}$, $x \in E$, $y \in F$ and $S,K \in \mathbb{N}$. Then*

$$\mathcal{A}(E,x,S) \leqslant_{\mathcal{L}} \mathcal{A}(F,y,K) \iff F \subset E \wedge x = y \wedge S \leqslant K.$$

Proof. We note $t_1 = \mathcal{A}(E,x,S)$ and $t_2 = \mathcal{A}(F,y,K)$. Let us suppose $F \subset E$, $x = y$ and $S \leqslant K$. Let σ be a valuation.

- If there exists $y \in F$ such that $\sigma(y) = 0$, then $[\![t_2]\!]_\sigma = 0$ and since $F \subset E$, $[\![t_1]\!]_\sigma = 0$.
- Else, $[\![t_1]\!]_\sigma \leqslant \sigma(x) + S \leqslant \sigma(x) + K = [\![t_2]\!]_\sigma$.

In both cases, $[\![t_1]\!]_\sigma \leqslant [\![t_2]\!]_\sigma$ hence $t_1 \leqslant_{\mathcal{L}} t_2$.

Now, we show the other implication by contraposition.

- If there exists $z \in F$ such that $z \notin E$, we take σ such that $\sigma(z) = 0$ and forall $j \neq z$, $\sigma(j) = 1$. We note that $z \neq x$ (since $z \notin E$ and $x \in E$) hence $\sigma(x) = 1$. Then, $[\![t_1]\!]_\sigma = S+1 > 0 = [\![t_2]\!]_\sigma$.
- If $x \neq y$ we take σ such that $\sigma(x) = K+2$, $\sigma(y) = 1$ and forall $z \neq x$ and $z \neq y$, $\sigma(z) = 1$. Then, $[\![t_1]\!]_\sigma = K+S+2 > K+1 = [\![t_2]\!]_\sigma$.
- If $S > K$ we take σ such that forall z, $\sigma(z) = 1$. Then, $[\![t_1]\!]_\sigma = S+1 > K+1 = [\![t_2]\!]_\sigma$.

◀

▶ **Proposition 31.** *Let $E,F \subset \mathcal{X}$ and $S,K \in \mathbb{N}$. Then*

$$\mathcal{B}(E,S) \leqslant_{\mathcal{L}} \mathcal{B}(F,K) \iff F \subset E \wedge S \leqslant K.$$

Proof. We note $t_1 = \mathcal{B}(E,S)$ and $t_2 = \mathcal{B}(F,K)$. Let us suppose $F \subset E$ and $S \leqslant K$. Let σ be a valuation.

- If there exists $y \in F$ such that $\sigma(y) = 0$, then $[\![t_2]\!]_\sigma = 0$ and since $F \subset E$, $[\![t_1]\!]_\sigma = 0$.
- Else, $[\![t_1]\!]_\sigma \leqslant K \leqslant S = [\![t_2]\!]_\sigma$.

In both cases, $[\![t_1]\!]_\sigma \leqslant [\![t_2]\!]_\sigma$ hence $t_1 \leqslant_{\mathcal{L}} t_2$.

Now, we show the other implication by contraposition.

- If there exists $y \in F$ such that $y \notin E$, we take σ such that $\sigma(y) = 0$ and forall $z \neq y$, $\sigma(z) = 1$. Then, $[\![t_1]\!]_\sigma = S > 0 = [\![t_2]\!]_\sigma$.
- If $S < K$ we take σ such that forall y, $\sigma(y) = 1$. Then, $[\![t_2]\!]_\sigma = K > S = [\![t_1]\!]_\sigma$.

◀

▶ **Proposition 32.** *Let $E,F \subset \mathcal{X}$, $x \in E$ and $K,S \in \mathbb{N}$. Then*

$$\mathcal{B}(E,S) \leqslant_{\mathcal{L}} \mathcal{A}(F,x,K) \iff (F \subset E \wedge S \leqslant K+1).$$

Proof. We note $t_1 = \mathcal{B}(E, S)$ and $t_2 = \mathcal{A}(F, x, K)$. Let us suppose $F \subset E$ and $S \leqslant K + 1$. Let σ be a valuation.

- If there exists $y \in F$ such that $\sigma(y) = 0$, then $[\![t_2]\!]_\sigma = 0$ and since $F \subset E$, $[\![t_1]\!]_\sigma = 0$.
- Else, $\sigma(x) \geq 1$ (because $x \in F$) and then $[\![t_2]\!]_\sigma = \sigma(x) + K \geq 1 + K \geq S \geq [\![t_1]\!]_\sigma$.

In both cases, $[\![t_1]\!]_\sigma \leqslant [\![t_2]\!]_\sigma$ hence $t_1 \leqslant_{\mathcal{L}} t_2$.

Now, we show the other implication by contraposition. First, we note that $S > 0$.

- If there exists $y \in F$ such that $y \notin E$, we take σ such that $\sigma(y) = 0$ and forall $z \neq y$, $\sigma(z) = 1$. Then, $[\![t_1]\!]_\sigma = K > 0 = [\![t_2]\!]_\sigma$.
- If $S > K + 1$ we take σ such that forall y, $\sigma(y) = 1$. Then, $[\![t_1]\!]_\sigma = S > K + 1 = [\![t_1]\!]_\sigma$.

◀

As a corollary, we get that the sublevel equivalence is a syntactic equality, which is quite natural; the uniqueness property would be impossible otherwise.

▶ **Corollary 33.** *Let $t_1, t_2 \in \mathcal{L}_s$. Then $t_1 =_{\mathcal{L}} t_2 \iff t_1 = t_2$.*

Proof. We have $t_1 =_{\mathcal{L}} t_2 \iff t_1 \leqslant_{\mathcal{L}} t_2 \wedge t_2 \leqslant_{\mathcal{L}} t_1$, and we conclude with the Theorem 29. ◀

And we can show the uniqueness of the minimal representation. First, we show that two equivalent minimal representations have the same $\mathcal{A}$.

▶ **Proposition 34.** *Let $t_1, t_2 \in \mathcal{L}_r$ such that $t_1 =_{\mathcal{L}} t_2$. Then*

$$\mathcal{A}(E, x, S) \in t_1 \iff \mathcal{A}(E, x, S) \in t_2.$$

Proof. Let us note

$$t_1 = \max(\mathcal{A}(E_0, x_0, S_0), \dots, \mathcal{A}(E_n, x_n, S_n), \mathcal{B}(G_0, T_0), \dots, \mathcal{B}(G_p, T_p))$$
$$t_2 = \max(\mathcal{A}(F_0, y_0, K_0), \dots, \mathcal{A}(F_m, y_m, K_m), \mathcal{B}(H_0, L_0), \dots, \mathcal{B}(H_q, L_q)).$$

Let $\mathcal{A}(E, x, S)$ be an sublevel of t_1. We consider σ such that

$$\sigma(y) = \begin{cases} \max(S_0, \dots, S_n, K_0, \dots, K_m, T_0, \dots, T_p, L_0, \dots, L_q) + 1 & \text{if } y = x \\ 1 & \text{if } y \in E \setminus \{x\} \\ 0 & \text{else} \end{cases}$$

We have $[\![t_1]\!]_\sigma = [\![\mathcal{A}(E, x, S)]\!]_\sigma = S + \sigma(x)$ and then $[\![t_2]\!]_\sigma = S + \sigma(x)$. Then, there exists $\mathcal{A}(F, y, K)$ in t_2 such that $F \subset E \cup \{x\} = E$ (else F contains a variable z such that $\sigma(z) = 0$) and $\sigma(y) + K = \sigma(x) + S$ or there exists $\mathcal{B}(F, K)$ in t_2 such that $K = \sigma(x) + S$. Since $\sigma(x) > \max(S_0, \dots, S_n, K_0, \dots, K_m)$, we deduce that it is the first case and $y = x$.

Then, there is (F, x, S) in t_2 with $F \subset E$ and $\sigma(y) + K = \sigma(x) + S$. If $F \subsetneq E$, then by the same reasoning, we show that there exists $\mathcal{A}(G, x, S) \in t_1$ with $G \subset F \subsetneq E$. But, by Definition 28, it is impossible to have $\mathcal{A}(E, x, S)$ and $\mathcal{A}(G, x, S)$ in t_1 with $G \subset E$ since they are comparable.

Then $E = F$ and $\mathcal{A}(E, x, S)$ is also an element of t_2. ◀

And we show the same for the $\mathcal{B}$.

▶ **Proposition 35.** *Let $t_1, t_2 \in \mathcal{L}_r$ such that $t_1 =_{\mathcal{L}} t_2$. Then*

$$\mathcal{B}(E, S) \in t_1 \iff \mathcal{B}(E, S) \in t_2.$$

Proof. Let us note

$$t_1 = \max(\mathcal{A}(E_0, x_0, S_0), \dots, \mathcal{A}(E_n, x_n, S_n), \mathcal{B}(G_0, T_0), \dots, \mathcal{B}(G_p, T_p))$$
$$t_2 = \max(\mathcal{A}(F_0, y_0, K_0), \dots, \mathcal{A}(F_m, y_m, K_m), \mathcal{B}(H_0, L_0), \dots, \mathcal{B}(H_q, L_q)).$$

We show the result by induction on E. Let $\mathcal{B}(E, S)$ be a sublevel of t_1. If $E = \emptyset$, we consider σ the zero function. Then, $[\![t_1]\!]_\sigma = S$, hence $[\![t_2]\!]_\sigma = S$. Since $S > 0$, it follows that $\mathcal{B}(\emptyset, S)$ is a sublevel of t_2.

In the induction case, we consider σ such that $\sigma(x) = 1$ if $x \in E$ and $\sigma(x) = 0$ otherwise, hence $[\![t_1]\!]_\sigma = S$. Then, $[\![t_2]\!]_\sigma = S$ and since $S > 0$, there exists $\mathcal{A}(F, x, K)$ in t_2 such that $F \subset E$ and $\sigma(x) + K = S$ or there exists $\mathcal{B}(F, S)$ in t_2 such that $F \subset E$ and $K = S$.

In the first case, we have $x \in F \subset E$, then $\sigma(x) = 1$ and $K = S - 1$. Then, by Proposition 34, $\mathcal{A}(F, x, S - 1) \in t_1$ which is impossible by Definition 28 since it would be comparable with $\mathcal{B}(E, S) \in t_1$.

Then, we have $\mathcal{B}(F, S) \in t_2$. If $F \subsetneq E$, we apply the induction hypothesis and obtain $\mathcal{B}(F, S) \in t_1$, impossible because it would be comparable with $\mathcal{B}(E, S)$.

Then $E = F$ and $\mathcal{B}(E, S)$ is also an element of t_2. ◀

We immediately obtain that equivalence of minimal representations is the syntactic equality.

▶ **Proposition 36.** *For all $t_1, t_2 \in \mathcal{L}_r$, $t_1 =_\mathcal{L} t_2 \iff t_1 = t_2$.*

Proof. The reverse implication is trivial and the first one is a consequences of the Propositions 34 and 35. ◀

And finally, we obtain the main theorem: the existence and uniquennes of a minimal representation for each level. First, we show the intuitive property that the minimal representation of a maximum of sublevels is formed with elements of these sublevels.

▶ **Proposition 37.** *For all $u_1, \dots, u_n \subset \mathcal{L}_s$, there exists a unique $\{v_1, \dots, v_m\} \subseteq \{u_1, \dots, u_n\}$ such that $\max(u_1, \dots, u_n) = \max(v_1, \dots, v_m)$ and $\max(v_1, \dots, v_m) \in \mathcal{L}_r$.*

Proof. We apply the following procedure. We consider $E = \{u_1, \dots, u_n\}$. While there exists $u, v \in E$ such taht $u \leqslant_\mathcal{L} v$, we remove u from E. The Proposition 36 permits to obtain the uniqueness of this minial representation. ◀

▶ **Theorem 38** (Representation). *For all $t \in \mathcal{L}$, there exists a unique $\{u_1, \dots, u_n\} \subset \mathcal{L}_s$ such that $t =_\mathcal{L} \max(u_1, \dots, u_n)$. We say that $\max(u_1, \dots, u_n)$ is the minimal representation of t and we denote it as $\mathrm{repr}(t)$.*

Proof. By Theorem 26, there exists $u_1, \dots, u_n \in \mathcal{L}_\mathrm{s}$ such that $t =_\mathcal{L} \max(u_1, \dots, u_n)$, and by Proposition 37, there exists a unique minimal representation of $\max(u_1, \dots, u_n)$. ◀

Having a unique minimal representation is useful to have a faithful embedding of $\mathcal{L}$ in the $\lambda\Pi/\equiv$, which is done in Section 4. Moreover, this representation also gives us an easy way to compare two terms. Indeed, a sublevel can be compared to a level using the representation.

▶ **Lemma 39.** *Let $u, v_1, \dots, v_n \in \mathcal{L}_s$. Then $u \leqslant_\mathcal{L} \max(v_1, \dots, v_n)$ if and only if there exists i such that $u \leqslant_\mathcal{L} v_i$.*

Proof. The reverse implication is trivial. We show the direct implication by contraposition. We suppose that forall i, $u \not\leqslant_\mathcal{L} v_i$.

If $u = \mathcal{B}(E, S)$, we consider σ such that $\sigma(x) = 1$ if $x \in E$ and 0 otherwise. Then, forall v_i, we have either

- $v_i = \mathcal{A}(F, x, K)$ or $v_i = \mathcal{B}(F, K)$ and $F \not\subset E$ hence $[\![v_i]\!]_\sigma = 0 < S = [\![u]\!]_\sigma$,
- or $v_i = \mathcal{A}(F, x, K)$ with $F \subseteq E$ and $K < S - 1$ hence $[\![v_i]\!]_\sigma = K + 1 < S = [\![u]\!]_\sigma$,
- or $v_i = \mathcal{B}(F, K)$ with $K < S - 1$ hence $[\![v_i]\!]_\sigma = K < S = [\![t]\!]_\sigma$.

Then $u \not\leqslant_{\mathcal{L}} \max(v_1, \dots, v_n)$.

If $u = \mathcal{A}(E, x, S)$, then, each v_i is of the form $\mathcal{A}(F_i, x_i, K_i)$ or $\mathcal{B}(F_i, K_i)$, we consider M the maximum of these K_i and σ such that $\sigma(x) = M + 2$, $\sigma(y) = 1$ if $y \in E \setminus \{x\}$ and 0 otherwise. Then, forall v_i, we have either

- $v_i = \mathcal{B}(F, K)$ hence $[\![v_i]\!]_\sigma \leqslant K < S + M + 2 = [\![u]\!]_\sigma$,
- or $v_i = \mathcal{A}(F, y, K)$ and $F \not\subset E$ hence $[\![v_i]\!]_\sigma = 0 < S = [\![u]\!]_\sigma$,
- or $v_i = \mathcal{A}(F, y, K)$ with $F \subseteq E$ and $x \neq y$, hence $[\![v_i]\!]_\sigma = K + 1 < S + M + 2 = [\![u]\!]_\sigma$,
- or $v_i = \mathcal{A}(F, x, K)$ with $F \subseteq E$ and $K < S$, hence $[\![v_i]\!]_\sigma = K + M + 2 < S + M + 2 = [\![u]\!]_\sigma$.

Then $u \not\leqslant_{\mathcal{L}} \max(v_1, \dots, v_n)$. ◀

And we use this lemma to compare two levels, for instance 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 40.** *Let $u_1, \dots, u_n, v_1, \dots, v_m \in \mathcal{L}_s$. Then, $\max(u_1, \dots, u_n) \leqslant_{\mathcal{L}} \max(v_1, \dots, v_m)$ if and only if forall i, there exists j such that $u_i \leqslant_{\mathcal{L}} v_j$.*

Proof. If $\max(u_1, \dots, u_n) \leqslant_{\mathcal{L}} \max(v_1, \dots, v_n)$, then for all i, $u_i \leqslant_{\mathcal{L}} \max(v_1, \dots, v_n)$ and by the Lemma 39, there exists v_j such that $u_i \leqslant_{\mathcal{L}} v_j$. The reverse implication is trivial. ◀

One can note that the Lemma 39 gives us a new proof of the uniqueness property stated in Proposition 36.

Proof. Let $u = \max(u_1, \dots, u_n)$ and $v = \max(v_1, \dots, v_m)$ be two minimal representations such that $u =_{\mathcal{L}} v$. We want to show that forall i, there exists j such that $u_i = v_j$.

We have $u_i \leqslant_{\mathcal{L}} u \leqslant_{\mathcal{L}} v$, hence by Lemma 39, there exists v_j such that $u_i \leqslant_{\mathcal{L}} v_j$. In the same way, there exists u_k such that $v_j \leqslant_{\mathcal{L}} u_k$. Then, by Definition 28, $i = k$ (because $u_1, \dots, u_n$ are incomparable), and then $u_i =_{\mathcal{L}} v_j$ hence $u_i = v_j$ by Corollary 33. ◀

This shows that there is a link between the Lemma 39 and the uniqueness property. In fact, this lemma should be understood as an *independence* lemma. Indeed, if we consider $\max(u_1, \dots, u_n)$ as a linear combination of $u_1, \dots, u_n$, then this lemma 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: $\mathcal{L}_{\mathrm{s}}$ is a 'linearly independent' family (uniqueness of the minimal representation) which generates at least $\mathcal{L}$ (for instance, $\max(\mathcal{A}(\{x\}, y, 0))$ or $\max(\mathcal{B}(\{x\}, 1))$ are not equivalent to any level).

4 A rewriting system for this universe representation

This section is dedicated to the implementation of this representation in the $\lambda\Pi/\equiv$.

4.1 Basic tools

Here, we define the very basic term that we will used. The booleans and the natural numbers, to begin with, are necessary.

▶ **Definition 41** (Booleans). *We define a type* $\mathtt{B}$*, with constructors* $\mathtt{true}\colon \mathtt{B}$*,* $\mathtt{false}\colon \mathtt{B}$ *and functions* $\mathtt{and}\colon \mathtt{B} \to \mathtt{B}$*,* $\mathtt{or}\colon \mathtt{B} \to \mathtt{B}$ *and* $\mathtt{not}\colon \mathtt{B} \to \mathtt{B}$*.* $\mathtt{B}$ *is interpreted as the Booleans,* $\mathtt{true}$*,* $\mathtt{false}$*,* $\mathtt{and}$*,* $\mathtt{or}$ *and* $\mathtt{not}$ *as* $\top$*,* $\bot$*, the conjunction, the disjunction and the negation.*

▶ **Definition 42** (Natural numbers). *We define a type* $\mathtt{N}$*, with constructors* $\mathtt{0}\colon \mathtt{N}$*,* $\mathtt{s}\colon \mathtt{N} \to \mathtt{N}$ *and functions* $\mathtt{+}\colon \mathtt{N} \to \mathtt{N} \to \mathtt{N}$*,* $\mathtt{<=_N}\colon \mathtt{N} \to \mathtt{N} \to \mathtt{B}$*,* $\mathtt{=_N}\colon \mathtt{N} \to \mathtt{N} \to \mathtt{B}$*,* $\mathtt{<_N}\colon \mathtt{N} \to \mathtt{N} \to \mathtt{B}$ *(with infix notation) and* $\mathtt{max_N}\colon \mathtt{N} \to \mathtt{N} \to \mathtt{N}$*.* $\mathtt{N}$ *is interpreted as* $\mathbb{N}$*,* $\mathtt{0}$ *as* 0*,* $\mathtt{s}$*,* $\mathtt{+}$*,* $\mathtt{<=_N}$*,* $\mathtt{=_N}$ *and* $\mathtt{max_N}$ *as* 0*,* s*,* $+$*,* $\leq$*,* $=$ *and* max*.*

Moreover, we define a *if-then-else* structure. It is not really necessary, but will be very convenient to facilitate the writing of some rules.

▶ **Definition 43.** *Let* T *be a type. We define the function* $\mathtt{ite}_T\colon \mathtt{B} \to T \to T \to T$ *such that* $\forall u, v \in T$*,* $\mathtt{ite}_T\ \mathtt{true}\ u\ v \hookrightarrow u$ *and* $\mathtt{ite}_T\ \mathtt{false}\ u\ v \hookrightarrow v$*. For convenience reasons, we will denote* $\mathtt{ite}_T\ b\ u\ v$ *by* $\mathtt{if}\ b\ \mathtt{then}\ u\ \mathtt{else}\ v$*.*

And finally, we show how to define a type of sets for all ordered types. It will be used for set of natural numbers (to define the sublevels), but also for set of sublevels (to define the representations).

▶ **Definition 44** (Sets). *Let* T *be a type equipped with a total order function* $\mathtt{leq}_T\colon T \to T \to \mathtt{B}$*. Then, we define a type* $\mathtt{S}[T]$ *corresponding to the finite set of elements of* T *with a constructor* $\{\}_T\colon \mathtt{S}[T]$ *and the functions (all with infix notation)* $\ll_T\colon \mathtt{S}[T] \to T \to \mathtt{S}[T]$*,* $\mathtt{++}_T\colon \mathtt{S}[T] \to \mathtt{S}[T] \to \mathtt{S}[T]$*,* $\mathtt{in}_T\colon T \to \mathtt{S}[T] \to \mathtt{B}$*,* $\mathtt{sub}_T\colon \mathtt{S}[T] \to \mathtt{S}[T] \to \mathtt{B}$ *and* $\mathtt{=_{S[T]}}\colon \mathtt{S}[T] \to \mathtt{S}[T] \to \mathtt{B}$*.*

$\mathtt{S}[T]$ *is interpreted as the set of finite subsets of* T*,* $\{\}_T$ *as the empty set,* $\ll_T$ *as the function that adds an element to a set,* $\mathtt{++}_T$*,* $\mathtt{in}_T$*,* $\mathtt{sub}_T$ *and* $\mathtt{=_{S[T]}}$ *as* $\cup$*,* $\in$*,* $\subseteq$ *and* $=$*.*

For convenience reasons, we will denote the term $\{\}_T \ll_T x_1 \ll_T \cdots \ll_T x_n$ *by* $\{x_1, \ldots, x_n\}_T$*. In particular,* $\{x\}_T$ *will represent a singleton.*

$\mathtt{S}[T]$ is implemented as a sorted list of elements of T with the constructors $\{\}_T\colon \mathtt{S}[T]$ and $::_T\colon T \to \mathtt{S}[T] \to \mathtt{S}[T]$, and the function $\ll_T$ adds an element to a list while keeping the uniqueness and order properties. Then, a set will be built through $\{\}_T$ and $\ll_T$ and the constructor $::_T$ will only be used in patterns of rewrite rules.

▶ **Definition 45** (Set order). *Let* T *be a type with a total order function* $\mathtt{leq}_T$*. Then, we define a total order* $\mathtt{leq_{S[T]}}$ *by considering the lexicographic order on sorted word of* T*. Besides, we define the corresponding strict total order* $\mathtt{lq_{S[T]}}$*.*

We do not give the rules for these elements since they are quite basic. However, the Dedukti implementation is available if necesary.

4.2 Level encoding

In order to implement the sets of sublevels and to compare two sublevels, we should be able to compare and sort level variables. That is why we use a deep encoding where each variable is encoded as a natural number. We denote by $\gamma\colon \mathcal{X} \to \mathbb{N}$ a bijection that associates each variable to a natural number.

▶ **Definition 46.** *We define* $\mathtt{L}$*, the type of the levels together with the constructors* $\mathtt{0_L}\colon \mathtt{L}$*,* $\mathtt{s_L}\colon \mathtt{L} \to \mathtt{L}$*,* $\mathtt{max_L}\colon \mathtt{L} \to \mathtt{L} \to \mathtt{L}$*,* $\mathtt{imax_L}\colon \mathtt{L} \to \mathtt{L} \to \mathtt{L}$ *and* $\mathtt{var_L}\colon \mathtt{N} \to \mathtt{L}$*. We define inductively a translation function* $|\cdot|\colon \mathcal{L} \to \mathtt{L}$ *with*

$$|0| = \mathtt{0_L} \qquad |s(t)| = \mathtt{s_L}\ |t| \qquad |x| = \mathtt{var_L}\ |\gamma(x)|_{\mathtt{N}}$$
$$|\max(u, v)| = \mathtt{max_L}\ |u|\ |v| \qquad |\mathrm{imax}(u, v)| = \mathtt{imax_L}\ |u|\ |v|$$

▶ **Definition 47.** *We define* $\mathtt{L_S}$*, the type of the sublevels, with the constructors* $\mathtt{a}\colon \mathtt{S[N]} \to \mathtt{N} \to \mathtt{N} \to \mathtt{L_S}$ *and* $\mathtt{b}\colon \mathtt{S[N]} \to \mathtt{N} \to \mathtt{L_S}$*. We define a translation function* $|\cdot|_s \colon \mathcal{L}_s \to \mathtt{L_S}$ *by* $|\mathcal{A}(E,x,S)|_s = \mathtt{a}\ |E|\ |x|\ |S|$ *and* $|\mathcal{B}(E,S)|_s = \mathtt{b}\ |E|\ |S|$*.*

In order to define $\mathtt{S[L_S]}$, we need a total order on $\mathtt{L_S}$. We consider the order where all the $\mathtt{a}$ are before the $\mathtt{b}$, and the $\mathtt{a}$ (respectively) the $\mathtt{b}$ are sorted according to the lexicographic order:

- $\mathtt{a}\ E\ x\ S \leqslant \mathtt{b}\ F\ K$,
- $\mathtt{a}\ E\ x\ S \leqslant \mathtt{a}\ |F|\ |y|\ |K|$ is the lexicographic order between (E,x,S) and (F,y,K) (which is a total order since $\mathtt{N}$ and $\mathtt{S[N]}$ are equipped with a total order by Definition 45),
- $\mathtt{b}\ E\ S \leqslant \mathtt{b}\ F\ K$ is the lexicographic order between (E,S) and (F,K).

▶ **Definition 48** (Total order on sublevels)**.** *We define a total* $\mathtt{leq_{L_s}}$ *on* $\mathtt{L_S}$ *(and its corresponding strict order* $\mathtt{lq_{L_s}}$*). Indeed, we consider these rewrite rules.*

$$(\mathtt{a}\ E\ x\ S)\ \mathtt{leq_{L_s}}\ (\mathtt{b}\ F\ K) \longrightarrow \mathtt{true}$$
$$(\mathtt{a}\ E\ x\ S)\ \mathtt{leq_{L_s}}\ (\mathtt{a}\ F\ y\ K) \longrightarrow \mathtt{or}\ (E\ \mathtt{lq_{S[N]}}\ F)\ \mathtt{and}\ (E =_{\mathtt{S[N]}} F)\ ((\mathtt{or}\ (x <_{\mathtt{N}} y)\ (\mathtt{and}\ (x =_{\mathtt{N}} y)\ (S <=_{\mathtt{N}} K))))$$
$$\mathtt{b}\ E\ S\ \mathtt{leq_{L_s}}\ \mathtt{b}\ F\ K \longrightarrow \mathtt{or}\big(E\ \mathtt{lq_{S[N]}}\ F, \mathtt{and}(E =_{\mathtt{S[N]}} F, S <=_{\mathtt{N}} K)\big)$$

And we can then define the representations.

▶ **Definition 49.** *We define a function* $\mathtt{max_{L_s}}\colon \mathtt{S[L_S]} \to \mathtt{L}$ *that embeds a set of* $\mathtt{L_S}$ *into a* $\mathtt{L}$ *and a translation function* $|\cdot|_r \colon \mathcal{L}_r \to \mathtt{L}$ *defined by* $|\max(u_1,\dots,u_n)|_r = \mathtt{max_{L_s}}\big(\{|u_1|_{\mathtt{s}},\dots,|u_n|_{\mathtt{s}}\}_T\big)$*.*

Now, we have defined all the types and the elements of $\mathcal{L}_r$ have translations in $\lambda\Pi/\equiv$. The next step is to provide rewrite rules that transform a level into its minimal representation. The cases of $\mathtt{0_L}$ and of variables are easy.

$$\mathtt{0_L} \longrightarrow \mathtt{max_{L_s}}(\{\}_{\mathtt{L_s}}) \qquad \mathtt{var_L}(x) \longrightarrow \mathtt{max_{L_s}}\big(\{\mathtt{a}(\{x\}_{\mathtt{N}}, x, 0)\}_{\mathtt{L_s}}\big)$$

For the other cases, and in particular for the max, the sublevel comparison will be useful, so we define it.

▶ **Definition 50** (Sublevels comparison)**.** *We define* $<=_{\mathtt{L_s}}\colon \mathtt{L_S} \to \mathtt{L_S} \to \mathtt{B}$ *interpreted as* $\leqslant_{\mathcal{L}}$ *with these rewrite rules.*

$$\mathtt{a}(E,x,S) <=_{\mathtt{L_s}} \mathtt{b}(F,K) \longrightarrow \mathtt{false} \tag{1}$$
$$\mathtt{b}(E,S) <=_{\mathtt{L_s}} \mathtt{b}(F,K) \longrightarrow \mathtt{and}(F\ \mathtt{sub_N}\ E, S <=_{\mathtt{N}} K) \tag{2}$$
$$\mathtt{b}(E,\mathtt{s_L}(S)) <=_{\mathtt{L_s}} \mathtt{a}(F,y,K) \longrightarrow \mathtt{and}(F\ \mathtt{sub_N}\ E, S <=_{\mathtt{N}} K+1) \tag{3}$$
$$\mathtt{a}(E,x,S) <=_{\mathtt{L_s}} \mathtt{a}(F,y,K) \longrightarrow \mathtt{and}(F\ \mathtt{sub_N}\ E, \mathtt{and}(x =_{\mathtt{N}} y, S <=_{\mathtt{N}} K)) \tag{4}$$

▶ **Proposition 51.** *Let* $u, v \in \mathcal{L}_s$*. Then,* $u \leqslant_{\mathcal{L}} v \iff |u| <=_{\mathtt{L_s}} |v| \hookrightarrow^* \mathtt{true}$*.*

Proof. Each rule (i) corresponds to the case i of the Theorem 29. ◀

And now, we can give the rules to normalize the other levels.

4.3 The successor

We show how to compute the minimal representation of $s(\max(u_1, \dots, u_n))$.

▶ **Definition 52.** *We define* s_s *on the sublevels by* $\mathrm{s}_s(\mathcal{A}(E, x, S)) = \mathcal{A}(E, x, S+1)$ *and* $\mathrm{s}_s(\mathcal{B}(E, S)) = \mathcal{B}(E, S+1)$. *Then* $\mathrm{s}_s(u) =_{\mathcal{L}} s(u)$.

▶ **Proposition 53.** $\mathrm{repr}(s(\max(\emptyset))) = \max(\mathcal{B}(\emptyset), s(0))$ *and for all* $n > 0$ *and* $t = \max(u_1, \dots, u_n) \in \mathcal{L}_r$, $\mathrm{repr}(s(t)) = \max(\mathrm{s}_s(u_1), \dots, \mathrm{s}_s(u_n))$.

We create a function $\mathtt{s}_{\mathtt{L}_\mathtt{s}} : \mathtt{S[L_S]} \to \mathtt{S[L_S]}$ corresponding to s_s.

$$\mathtt{s}_{\mathtt{L}_\mathtt{s}}(\{\}_{\mathtt{L}_\mathtt{s}}) \longrightarrow \{\}_{\mathtt{L}_\mathtt{s}} \qquad \mathtt{s}_{\mathtt{L}_\mathtt{s}}(\mathtt{b}(E, S) ::_{\mathtt{L}_\mathtt{s}} q) \longrightarrow \mathtt{s}_{\mathtt{L}_\mathtt{s}}(q) \ll_{\mathtt{L}_\mathtt{s}} \mathtt{b}(E, \mathtt{s}(S))$$
$$\mathtt{s}_{\mathtt{L}_\mathtt{s}}(\mathtt{a}(E, x, S) ::_{\mathtt{L}_\mathtt{s}} q) \longrightarrow \mathtt{s}_{\mathtt{L}_\mathtt{s}}(q) \ll_{\mathtt{L}_\mathtt{s}} \mathtt{a}(E, x, \mathtt{s}(S))$$

And we compute the minimal representation of a successor according to Proposition 53.

$$\mathtt{s}_\mathtt{L}(\mathtt{max}_{\mathtt{L}_\mathtt{s}}(\{\}_{\mathtt{L}_\mathtt{s}})) \longrightarrow \mathtt{max}_{\mathtt{L}_\mathtt{s}}(\mathtt{b}(\{\}_{\mathtt{L}_\mathtt{s}}, \mathtt{s}(0))) \qquad \mathtt{s}_\mathtt{L}(\mathtt{max}_{\mathtt{L}_\mathtt{s}}(u ::_{\mathtt{L}_\mathtt{s}} q)) \longrightarrow \mathtt{max}_{\mathtt{L}_\mathtt{s}}(\mathtt{s}_{\mathtt{L}_\mathtt{s}}(u ::_{\mathtt{L}_\mathtt{s}} q))$$

▶ **Proposition 54.** *For all* $t \in \mathcal{L}_r$, $\mathtt{s}_\mathtt{L}(|t|_r) \hookrightarrow^* |\mathrm{repr}(s(t))|_r$.

4.4 The maximum

For the maximum, we define $\max_\mathrm{s}$, that adds a sublevel to a minimal representation.

▶ **Definition 55.** *We define inductively* $\max_s : \mathcal{P}(\mathcal{L}_s) \times \mathcal{L}_s \to \mathcal{P}(\mathcal{L}_s)$ *inductively defined by* $\max_s(\emptyset, u) = \{u\}$ *and*

$$\max_s(\{u_1, \dots, u_n\} \cup \{u\}, v) = \begin{cases} \{u_1, \dots, u_n, u\} & \text{if } v \leqslant_{\mathcal{L}} u, \\ \{u_1, \dots, u_n, v\} & \text{if } u \leqslant_{\mathcal{L}} v, \\ \max_s(\{u_1, \dots, u_n\}, v) \cup \{u\} & \text{else.} \end{cases}$$

▶ **Proposition 56.** *For all minimal representations* $t = \max(u_1, \dots, u_n) \in \mathcal{L}_r$ *and* $v \in \mathcal{L}_s$, $\mathrm{repr}(\max(u_1, \dots, u_n, v)) = \max_s(\{u_1, \dots, u_n\}, v)$.

We implement $\max_\mathrm{s}$ as a function $\mathtt{maxhelper} : \mathtt{S[L_S]} \to \mathtt{L_S} \to \mathtt{S[L_S]}$ with these rewrite rules.

$$\mathtt{maxhelper}(\{\}_{\mathtt{L}_\mathtt{s}}, u) \longrightarrow \{u\}_{\mathtt{L}_\mathtt{s}}$$
$$\mathtt{maxhelper}(u ::_{\mathtt{L}_\mathtt{s}} E, v) \longrightarrow \mathtt{if}\ v <=_{\mathtt{L}_\mathtt{s}} u\ \mathtt{then}\ E \ll_{\mathtt{L}_\mathtt{s}} u$$
$$\mathtt{else\ if}\ u <=_{\mathtt{L}_\mathtt{s}} v\ \mathtt{then}\ E \ll_{\mathtt{L}_\mathtt{s}} v\ \mathtt{else}\ \mathtt{maxhelper}(E, v) \ll_{\mathtt{L}_\mathtt{s}} u$$

And we compute the minimal representation for $\mathtt{max}_\mathtt{L}$ according to Proposition 56.

$$\mathtt{max}_\mathtt{L}(\mathtt{max}_{\mathtt{L}_\mathtt{s}}(E), \mathtt{max}_{\mathtt{L}_\mathtt{s}}(\{\}_{\mathtt{L}_\mathtt{s}})) \longrightarrow \mathtt{max}_{\mathtt{L}_\mathtt{s}}(E)$$
$$\mathtt{max}_\mathtt{L}(\mathtt{max}_{\mathtt{L}_\mathtt{s}}(E), \mathtt{max}_{\mathtt{L}_\mathtt{s}}(u ::_{\mathtt{L}_\mathtt{s}} F)) \longrightarrow \mathtt{max}_\mathtt{L}(\mathtt{maxhelper}(E, u), \mathtt{max}_{\mathtt{L}_\mathtt{s}}(F))$$

▶ **Proposition 57.** *For all* $t_1, t_2 \in \mathcal{L}_r$, $\mathtt{max}_\mathtt{L}(|t_1|_r, |t_2|_r) \hookrightarrow^* |\mathrm{repr}(\max(t_1, t_2))|_r$.

4.5 The impredicative maximum

To begin, we study $\mathrm{imax}(u, v)$ where $u, v \in \mathcal{L}_\mathrm{s}$.

▶ **Proposition 58.** *Let* $f(E, S)$ *be either* $\mathcal{A}(E, x, S)$ *or* $\mathcal{B}(E, S)$ *and* $g(F, K)$ *be either* $\mathcal{A}(F, x, K)$ *or* $\mathcal{B}(F, K)$. *Then*

$$\mathrm{imax}(f(E, S), g(F, K)) =_{\mathcal{L}} \max(f(E \cup F, S), g(F, K)).$$

Proof. We note $u = f(E, x, S)$ and $v = g(F, K)$. Let σ be a valuation.

- If there exists $y \in F$ such that $\sigma(y) = 0$, then $[\![\text{imax}(u, v)]\!]_\sigma = 0 = [\![\max(f(E \cup F, S), v)]\!]_\sigma$.
- Else, $[\![v]\!]_\sigma = K > 0$ and then $[\![\text{imax}(u, v)]\!]_\sigma = \max([\![u]\!]_\sigma, K)$. Besides, since $\sigma(y) \neq 0$ forall $y \in F$, there exists $y \in E \cup F$ such that $\sigma(y) = 0$ if and only if there exists $y \in E$ such that $\sigma(y) = 0$, hence $[\![u]\!]_\sigma = [\![f(E \cup F, S)]\!]_\sigma$.

Hence the result. ◀

We implement it as a function $\texttt{imax}_{\texttt{L}_\texttt{s}} : \texttt{L}_\texttt{S} \to \texttt{L}_\texttt{S} \to \texttt{L}$.

$$\begin{aligned}
\texttt{imax}_{\texttt{L}_\texttt{s}}(\texttt{a}(E, x, S), \texttt{b}(F, K)) &\longrightarrow \texttt{max}_\texttt{L}(\texttt{max}_{\texttt{L}_\texttt{s}}(\texttt{a}(E \mathbin{+\!\!+_\texttt{N}} F, x, S)), \texttt{max}_{\texttt{L}_\texttt{s}}(\texttt{b}(F, K))) \\
\texttt{imax}_{\texttt{L}_\texttt{s}}(\texttt{b}(E, S), \texttt{b}(F, K)) &\longrightarrow \texttt{max}_\texttt{L}(\texttt{max}_{\texttt{L}_\texttt{s}}(\texttt{b}(E \mathbin{+\!\!+_\texttt{N}} F, S)), \texttt{max}_{\texttt{L}_\texttt{s}}(\texttt{b}(F, K))) \\
\texttt{imax}_{\texttt{L}_\texttt{s}}(\texttt{b}(E, S), \texttt{a}(F, x, K)) &\longrightarrow \texttt{max}_\texttt{L}(\texttt{max}_{\texttt{L}_\texttt{s}}(\texttt{b}(E \mathbin{+\!\!+_\texttt{N}} F, S)), \texttt{max}_{\texttt{L}_\texttt{s}}(\texttt{a}(F, x, K))) \\
\texttt{imax}_{\texttt{L}_\texttt{s}}(\texttt{a}(E, x, S), \texttt{a}(F, y, K)) &\longrightarrow \texttt{max}_\texttt{L}(\texttt{max}_{\texttt{L}_\texttt{s}}(\texttt{a}(E \mathbin{+\!\!+_\texttt{N}} F, x, S)), \texttt{max}_{\texttt{L}_\texttt{s}}(\texttt{a}(F, y, K)))
\end{aligned}$$

Then, following the equalities $\text{imax}(0, t) =_\mathcal{L} t$ and $\text{imax}(t, 0) =_\mathcal{L} 0$, and Propositions 11 and 12, we define $\texttt{imax\_aux} : \texttt{L}_\texttt{S} \to \texttt{L} \to \texttt{L}$ and add these rewrite rules.

$$\begin{aligned}
\texttt{imax}_\texttt{L}(\texttt{max}_{\texttt{L}_\texttt{s}}(\{\}_{\texttt{L}_\texttt{s}}), t) &\longrightarrow t \\
\texttt{imax}_\texttt{L}(\texttt{max}_{\texttt{L}_\texttt{s}}(u ::_{\texttt{L}_\texttt{s}} q), t) &\longrightarrow \texttt{max}_\texttt{L}(\texttt{imax_aux}(u, t), \texttt{imax}_\texttt{L}(q, t)) \\
\texttt{imax_aux}(u, \texttt{max}_{\texttt{L}_\texttt{s}}(\{\}_{\texttt{L}_\texttt{s}})) &\longrightarrow \texttt{max}_{\texttt{L}_\texttt{s}}(\{\}_{\texttt{L}_\texttt{s}}) \\
\texttt{imax_aux}(u, \texttt{max}_{\texttt{L}_\texttt{s}}(v ::_{\texttt{L}_\texttt{s}} q)) &\longrightarrow \texttt{max}_\texttt{L}(\texttt{imax}_{\texttt{L}_\texttt{s}}(u, v), \texttt{imax_aux}(u, q))
\end{aligned}$$

▶ **Proposition 59.** *For all $t_1, t_2 \in \mathcal{L}_r$, $\texttt{imax}_\texttt{L}(|t_1|_r, |t_2|_r) \hookrightarrow^* |\text{repr}(\text{imax}(t_1, t_2))|_r$.*

Proof. By Propositions 57 and 58, for all $u, v \in \mathcal{L}_\text{s}$, $\texttt{imax}_{\texttt{L}_\texttt{s}}(|u|_\text{s}, |v|_\text{s}) \hookrightarrow^* |\text{repr}(\text{imax}(u, v))|$. Then, by induction on $t \in \mathcal{L}_r$ and using Proposition 57, we show that forall $u \in \mathcal{L}_\text{s}$, $\texttt{imax\_aux}(|u|_\text{s}, |t|_\text{r}) \hookrightarrow^* |\text{repr}(\text{imax}(u, t))|_\text{r}$. And finally, we show the result by induction on t_2 using Propositions 11 and 12 and the equivalences $\text{imax}(0, t) =_\mathcal{L} t$ and $\text{imax}(t, 0) =_\mathcal{L} 0$. ◀

4.6 Implementing the substitution

Since we use a deep encoding, β-reduction cannot be used for substitution. Then, we implement a substitution function. First, we implement $\texttt{eval}_{\texttt{L}_\texttt{s}} : \texttt{L}_\texttt{S} \to \texttt{N} \to \texttt{N} \to \texttt{L}$ for the sublevels following the semantic given in the Definition 21.

$$\begin{aligned}
\texttt{eval}_{\texttt{L}_\texttt{s}}(\texttt{b}(E, S), y, n) &\longrightarrow \texttt{if}\,\texttt{and}(y\,\texttt{in}_\texttt{N}\,E, n =_\texttt{N} 0)\,\texttt{then}\,\texttt{max}_{\texttt{L}_\texttt{s}}(\{\}_{\texttt{L}_\texttt{s}}) \\
&\qquad\qquad \texttt{else}\,\texttt{max}_{\texttt{L}_\texttt{s}}(\texttt{b}(E \setminus_\texttt{N} y, S)) \\
\texttt{eval}_{\texttt{L}_\texttt{s}}(\texttt{a}(E, x, S), y, n) &\longrightarrow \texttt{if}\,\texttt{and}(y\,\texttt{in}_\texttt{N}\,E, n =_\texttt{N} 0)\,\texttt{then}\,\texttt{max}_{\texttt{L}_\texttt{s}}(\{\}_{\texttt{L}_\texttt{s}}) \\
&\qquad\qquad \texttt{else}\,\texttt{if}\,x =_\texttt{N} y\,\texttt{then}\,\texttt{max}_{\texttt{L}_\texttt{s}}(\texttt{b}(\{\}_\texttt{N}, S + n)) \\
&\qquad\qquad \texttt{else}\,\texttt{max}_{\texttt{L}_\texttt{s}}(\texttt{a}(E \setminus_\texttt{N} y, x, S))
\end{aligned}$$

Then, we create a function $\texttt{eval}_\texttt{L} : \texttt{L} \to \texttt{N} \to \texttt{N} \to \texttt{L}$ that evaluate a level using the fact that $[\max(u_1, \ldots, u_n)]\{x \mapsto n\} = \max([u_1]\{x \mapsto n\}, \ldots, [u_n]\{x \mapsto n\})$.

$$\texttt{eval}_\texttt{L}(\{\}_{\texttt{L}_\texttt{s}}, y, n) \longrightarrow \{\}_{\texttt{L}_\texttt{s}} \qquad \texttt{eval}_\texttt{L}(u ::_{\texttt{L}_\texttt{s}} q, y, n) \longrightarrow \texttt{max}_\texttt{L}(\texttt{eval}_{\texttt{L}_\texttt{s}}(u, y, n), \texttt{eval}_\texttt{L}(q, y, n))$$

▶ **Proposition 60.** *Let $t \in \mathcal{L}$. Then, the normal form of $|[t]\{x \mapsto u\}|$ is $\texttt{eval}_\texttt{L}(|t|, |u|)$.*

4.7 Properties of the rewrite system

The rewrite system that we designed have strong properties. First, one can note that it is does not use any higher-order rewrite rule, hence it can be implemented in a first order system.

▶ **Theorem 61.** *The rewrite system is confluent and strongly normalizing.*

Proof. The termination has been proved with two termination checkers, TTT2 [26] and SizeChangeTool [19], and the confluence with CSI [34]. ◀

And of course, we show that it is sound relatively to the minimal representation.

▶ **Theorem 62** (Soundness). *Let $t \in \mathcal{L}$. Then, the normal form of $|t|$ is $|\mathrm{repr}(t)|_r$.*

Proof. We show that $t \hookrightarrow^* |\mathrm{repr}(t)|_{\mathrm{r}}$ by induction on t: $\mathtt{var}_{\mathtt{L}}(x) \hookrightarrow \mathtt{max}_{\mathtt{L_s}}\big(\{\mathtt{a}(\{x\}_{\mathtt{N}}, 0, x)\}_{\mathtt{L_s}}\big)$, $0 \hookrightarrow \mathtt{max}_{\mathtt{L_s}}\ \{\}_{\mathtt{L_s}}$, and we show the cases s, max and imax using Propositions 54, 57, and 59. Since no rewrite rule can be applied to $\mathtt{max}_{\mathtt{L_s}}$, the normal form of $|t|$ is $|\mathrm{repr}(t)|_{\mathrm{r}}$. ◀

Theorems 38 and 62 give us that the translations of two equivalent levels are convertible, and even more, they have the same normal form. In other words, this embedding faithfully represent the level equivalences.

Besides, one could note some drawbacks of this embedding. First, we do not have a back translation from $\lambda\Pi/\equiv$ to $\mathtt{L}$. There are two main reason for this.

1. $\mathtt{a}$ and $\mathtt{b}$ are not exact translations of $\mathcal{A}$ and $\mathcal{B}$.
2. $\mathcal{L}_r$ and $\mathcal{L}$ are not equivalent.

The first reason is linked to the restrictions that we added to $\mathcal{A}$ and $\mathcal{B}$. Here, it is possible to write terms $\mathtt{b}\ E\ \mathtt{0}$ or $\mathtt{a}\ E\ x\ k$ where x is not an element of E. In the same way, it is possible to write $\mathtt{max}_{\mathtt{L_s}}\ L$ while two sublevels of L are comparable.

A solution could be to add a dependent term as argument, to check these conditions. For instance, $\mathtt{a}$ would have the type $(E\colon \mathtt{S[N]}) \to (x\colon \mathtt{N}) \to \mathtt{Prf}\ (x\ \mathtt{in}_{\mathtt{N}}\ E) \to \mathtt{N} \to \mathtt{L_S}$ where $\mathtt{Prf}\colon \mathtt{B} \to \mathtt{Type}$ represents the proof of a proposition. We declare $\mathtt{I}\colon \mathtt{Prf\ true}$ and we use $\mathtt{a}\ E\ x\ k\ \mathtt{I}$ using the fact that $x\ \mathtt{in}_{\mathtt{N}}\ E$ reduces to $\mathtt{true}$ if and only if x is an element of E.

The second reason is not related to the embedding, but to the representation that we introduced. Indeed, we already noted that some minimal representations are not equivalent to any level ($\max(\mathcal{A}(\{x\}, y, 0))$ or $\max(\mathcal{B}(\{x\}, 1))$ as examples). Then, it could be a good idea to find a characterization of the elements of $\mathcal{L}_r$ that actually correspond to levels.

5 Conclusion

We introduced a new representation of the levels of the impredicative PTS where equivalent levels have the same representation. It provides us an easy procedure decision for the inequality problem in the imax-sucessor algebra, and it permits us to get a sound encoding of these levels in the $\lambda\Pi/\equiv$, in the sense that equivalent levels have convertible translations. Moreover, this encoding corresponds to a first-order, confluent and strongly normalizing rewrite system, and in particular it permits to decide level equality.

This encoding of the levels permits to encode CC^{∞} with universe polymorphism. Besides, we still have to study how this encoding behaves well together with encodings of inductive types or cumulativity in order to get a better encoding of Coq. The ideas mentioned at the end of Section 4 are also interesting. In particular, the characterization of the elements of $\mathcal{L}_r$ that are actually levels would lead to a better understanding of the imax-successor grammar.

Finally, this idea of representation, with the linear algebra analogy, could certainly be adapted to some sets of terms built over a maximum, a supremum or other similar operations. In addition to providing decision procedures, it would permit to define a concept similar to the basis of a vector space on these spaces on these sets, and then it seems to be an interesting direction to explore.

References

1. Ali Assaf. A calculus of constructions with explicit subtyping. In Hugo Herbelin, Pierre Letouzey, and Matthieu Sozeau, editors, *20th International Conference on Types for Proofs and Programs (TYPES 2014)*, volume 39 of *LIPICS*, Institut Henri Poincaré, Paris, France, May 2014. URL: https://hal.archives-ouvertes.fr/hal-01097401.
2. Ali Assaf. *A framework for defining computational higher-order logics*. Theses, École polytechnique, 09 2015. URL: https://pastel.archives-ouvertes.fr/tel-01235303.
3. 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 $\lambda\pi$-calculus modulo theory. 2016.
4. 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.
5. Henk Barendregt. Introduction to generalized type systems. *Journal of Functional Programming*, 1(2):125–154, 1991. doi:10.1017/S0956796800020025.
6. Henk Barendregt, S. Abramsky, D. Gabbay, T. Maibaum, and Henk (Hendrik) Barendregt. Lambda calculi with types. 10 2000.
7. Michael Beeson, Pierre Boutry, Gabriel Braun, Charly Gries, and Julien Narboux. GeoCoq, June 2018. URL: https://inria.hal.science/hal-01912024.
8. Stefano Berardi. *Type dependence and constructive mathematics*. PhD thesis, PhD thesis, Dipartimento di Informatica, Torino, Italy, 1990.
9. 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. URL: https://drops.dagstuhl.de/opus/volltexte/2022/16305, doi:10.4230/LIPIcs.FSCD.2022.24.
10. 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. URL: https://drops.dagstuhl.de/opus/volltexte/2021/14258, doi:10.4230/LIPIcs.FSCD.2021.20.
11. Mathieu Boespflug and Guillaume Burel. Coqine: Translating the calculus of inductive constructions into the $\lambda\pi$-calculus modulo. In *in "Second International Workshop on Proof Exchange for Theorem Proving*, 2012.
12. Mathieu Boespflug, Quentin Carbonneaux, and Olivier Hermant. The $\lambda\Pi$-calculus Modulo as a Universal Proof Language. *CEUR Workshop Proceedings*, 878, 06 2012.
13. Thierry Coquand and Christine Paulin. Inductively defined types. In Per Martin-Löf and Grigori Mints, editors, *COLOG-88*, pages 50–66, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.
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.

15 Denis Cousineau and Gilles Dowek. Embedding pure type systems in the lambda-pi-calculus modulo. pages 102–117, 06 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 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.

18 Michael Färber. Safe, fast, concurrent proof checking for the lambda-pi calculus modulo rewriting. pages 225–238, 01 2022. doi:10.1145/3497775.3503683.

19 Guillaume Genestier. SizeChangeTool: A Termination Checker for Rewriting Dependent Types. In Mauricio Ayala-Rincón, Silvia Ghilezan, and Jakob Grue Simonsen, editors, *HOR 2019 - 10th International Workshop on Higher-Order Rewriting*, Joint Proceedings of HOR 2019 and IWC 2019, pages 14–19, Dortmund, Germany, June 2019. URL: https://hal.science/hal-02442465.

20 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. URL: https://drops.dagstuhl.de/opus/volltexte/2020/12353, doi:10.4230/LIPIcs.FSCD.2020.31.

21 Yoan Géran. Stt∀ geocoq, 2021. URL: https://github.com/Karnaj/sttfa_geocoq_euclid.

22 Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *J. ACM*, 40(1):143–184, 1 1993. doi:10.1145/138027.138060.

23 Robert Harper and Robert Pollack. Type checking with universes. In *2nd International Joint Conference on Theory and Practice of Software Development*, TAPSOFT '89, page 107–136, NLD, 1991. Elsevier Science Publishers B. V.

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. URL: https://drops.dagstuhl.de/opus/volltexte/2020/12357, doi:10.4230/LIPIcs.FSCD.2020.35.

25 Gérard Huet. Extending the calculus of constructions with type: Type, 1988. Unpublished draft. URL: https://pauillac.inria.fr/~huet/PUBLIC/typtyp.pdf.

26 Martin Korp, Christian Sternagel, Harald Zankl, and Aart Middeldorp. Tyrolean termination tool 2. In Ralf Treinen, editor, *Rewriting Techniques and Applications, 20th International Conference, RTA 2009, Brasília, Brazil, June 29 - July 1, 2009, Proceedings*, volume 5595 of *Lecture Notes in Computer Science*, pages 295–304. Springer, 2009. doi:10.1007/978-3-642-02348-4_21.

27 Zhaohui Luo. Notes on universes in type theory, October, 2012. URL: https://www.cs.rhul.ac.uk/home/zhaohui/universes.pdf.

28 Christine Paulin-Mohring. Inductive definitions in the system coq rules and properties. In Marc Bezem and Jan Friso Groote, editors, *Typed Lambda Calculi and Applications*, pages 328–345, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

29 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.

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

31 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.

32 François Thiré. *Interoperability between proof systems using the logical framework Dedukti. (Interopérabilité entre systèmes de preuve en utilisant le cadre logique Dedukti)*. PhD thesis, École normale supérieure Paris-Saclay, Cachan, France, 2020. URL: https://tel.archives-ouvertes.fr/tel-03224039.

33 Vladimir Voevodsky. A universe polymorphic type system, October 22, 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.

34 Harald Zankl, Bertram Felgenhauer, and Aart Middeldorp. Csi: a confluence tool. volume 6803, pages 499–505, 07 2011. doi:10.1007/978-3-642-22438-6_38.