



HAL
open science

Encoding impredicative hierarchy of type universes with variables

Yoan Gérard

► **To cite this version:**

| Yoan Gérard. Encoding impredicative hierarchy of type universes with variables. 2023. hal-04311936

HAL Id: hal-04311936

<https://hal.science/hal-04311936>

Preprint submitted on 28 Nov 2023

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

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



Distributed under a Creative Commons Attribution 4.0 International License

1 Encoding impredicative hierarchy of type universes 2 with variables

3 Yoan Gérard ✉

4 Mines Paris - PSL, Centre de Recherche en Informatique

5 Université Paris-Saclay, Laboratoire Méthodes Formelles, ENS Paris-Saclay

6 — Abstract —

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

15 **2012 ACM Subject Classification** Theory of computation → Equational logic and rewriting; Theory
16 of computation → Type theory

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

18 **Digital Object Identifier** 10.4230/LIPIcs.CVIT.2016.23

19 **Supplementary Material** An implementation is available at [https://gitlab.crans.org/geran/
20 dedukti-level-implementation](https://gitlab.crans.org/geran/dedukti-level-implementation)

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

23 **1** Introduction

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

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

36 Pure Type Systems

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



© Yoan Gérard;

licensed under Creative Commons License CC-BY 4.0

42nd Conference on Very Important Topics (CVIT 2016).

Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:20

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

23:2 Encoding impredicative hierarchy of type universes with variables

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

44 ► **Definition 1.** A *Pure Type System (PTS)* is defined by a set of sorts \mathcal{S} (that we will also
45 call *universes*), a set of axioms $\mathcal{A} \subseteq \mathcal{S}^2$ and a set of rules $\mathcal{R} \subseteq \mathcal{S}^3$.

46 \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
47 possible quantifications and their typing rules. The terms are the following, where $s \in \mathcal{S}$ and
48 x is an element of a countable set of variables \mathcal{X} .

49 $t := s \mid x \mid \Pi x : t \cdot t \mid (\lambda x : t \cdot t) \mid t t$

50 and the typing rules are given in Figure 1.

$$\begin{array}{c}
 51 \quad (\text{EMPTY}) \frac{}{[] \text{ WF}} \quad (\text{DECL}) \frac{\Gamma \vdash A : s \quad x \notin \Gamma}{\Gamma, x : A \text{ WF}} \quad (\text{VAR}) \frac{\Gamma \text{ WF} \quad (x : A) \in \Gamma}{\Gamma \vdash x : A} \\
 52 \quad (\text{SORT}) \frac{}{\vdash s_1 : s_2} (s_1, s_2) \in \mathcal{A} \quad (\text{PROD}) \frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A \cdot B : s_3} (s_1, s_2, s_3) \in \mathcal{R} \\
 53 \quad (\text{APP}) \frac{\Gamma \vdash t : \Pi x : A \cdot B \quad \Gamma, \vdash u : A}{\Gamma \vdash t u : B[x := u]} \quad (\text{ABS}) \frac{\Gamma, x : A \vdash t : B \quad \Gamma \vdash \Pi x : A \cdot B : s}{\Gamma \vdash \lambda x \cdot t : \Pi x : A \cdot B} \\
 54 \quad (\text{CONV}) \frac{\Gamma \vdash B : s \quad \Gamma \vdash t : A \quad A \equiv_{\beta} B}{\Gamma \vdash t : B} s \in \mathcal{S}
 \end{array}$$

■ **Figure 1** Typing rules

55 ► **Definition 2 (Functional and full PTS).** A *PTS* is said to be *functional* if \mathcal{A} and \mathcal{R} are
56 *functional relations* from \mathcal{S} and $\mathcal{S} \times \mathcal{S}$ to \mathcal{S} , that is to say $(s_1, s_2) \in \mathcal{A} \wedge (s_1, s_3) \in \mathcal{A} \implies$
57 $s_2 = s_3$ and $(s_1, s_2, s_3) \in \mathcal{R} \wedge (s_1, s_2, s_4) \in \mathcal{R} \implies s_3 = s_4$.

58 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} .

59 The $\lambda\Pi$ -calculus modulo rewriting

60 $\lambda\Pi$, the extension of STLC with dependent types, is the language of the *Edinburgh Logical*
61 *Framework* (ELF) [22]. However, computation plays an essential role in type theories, then
62 in modern proof assistant, and $\lambda\Pi$ is not well-suited for this. To address this point, the
63 $\lambda\Pi$ -calculus modulo rewriting ($\lambda\Pi/\equiv$) [12] extends $\lambda\Pi$ by allowing user-defined higher-order
64 rewrite rules [16, 30] that can be used to define functions but also types. Types are then
65 identified modulo β and these rewrite rules.

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

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

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

- 75 1. for each sort s , symbols $U_s : \text{Type}$ and $\text{EL}_s : U_s \rightarrow \text{Type}$,
- 76 2. for each axiom (s_1, s_2) , a symbol $u_{s_1} : U_{s_2}$ and a rewrite rule $\text{EL}_{s_2}(s_1) \longrightarrow U_{s_1}$,

77 3. for each rule (s_1, s_2, s_3) , a symbol $\pi_{(s_1, s_2)}: (x: \mathbf{U}_{s_1}) \rightarrow (\mathbf{El}_{s_1}(x) \rightarrow \mathbf{U}_{s_2}) \rightarrow \mathbf{U}_{s_3}$ and a
 78 rewrite rule $\mathbf{El}_{s_3}(\pi_{(s_1, s_2)}(A, B)) \longrightarrow (x: \mathbf{El}_{s_1}(A)) \rightarrow \mathbf{El}_{s_2}(B)$.

79 \mathbf{u}_s corresponds to the sort s as a term, \mathbf{U}_s to the sort s as a type, and \mathbf{El}_s associates a sort
 80 (as term) of type s to its corresponding type, hence the rewrite rule added for each axiom,
 81 and $\pi_{(s_1, s_2)}AB$ is the term corresponding to the types of the function from A to $B(A)$, hence
 82 the rewrite rule added to obtain the type associated to this term.

83 Several systems have been encoded in $\lambda\Pi/\equiv$: HOL-LIGHT [31, 2], AGDA [20], MATITA
 84 [2], but also parts of COQ, on which we will come back to later. Besides, since there exist
 85 multiple implementations of the $\lambda\Pi/\equiv$ such as DEDUKTI [3], LAMBDAPI [24], or KONTROLI
 86 [18], these embeddings have been effectively implemented leading to translations from the
 87 proofs systems to these implementations, but also to translations from these implementations
 88 of $\lambda\Pi/\equiv$ to proof assistants [31, 32].

89 Coq's type system

90 The theory of COQ is based on CC extended with an infinite hierarchy of universes and an
 91 impredicative universe Prop. It corresponds to a slightly different version of the following
 92 PTS (where Prop is denoted as \mathbf{Type}_0).

93 ► **Definition 4** (Impredicative max). We define $\mathbf{imax}: \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ by $\mathbf{imax}(i, 0) = 0$ and
 94 $\mathbf{imax}(i, j + 1) = \max(i, j + 1)$.

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

97 ► **Remark 6.** One can also define a predicative PTS where the products from \mathbf{Type}_i to \mathbf{Type}_j
 98 are elements of $\mathbf{Type}_{\max(i, j)}$ instead of $\mathbf{Type}_{\mathbf{imax}(i, j)}$. This latter is the building on AGDA
 99 proof system while CC^∞ is the one of COQ but also of LEAN or MATITA.

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

- 104 ■ symbols $\mathbf{U}: \mathbb{N} \rightarrow \mathbf{Type}$ and $\mathbf{El}: (i: \mathbb{N}) \rightarrow \mathbf{U}(i) \rightarrow \mathbf{Type}$,
- 105 ■ a symbol $\mathbf{u}: (i: \mathbb{N}) \rightarrow \mathbf{U}(\mathbf{s}(i))$ and a rewrite rule $\mathbf{El} _ i \longrightarrow \mathbf{U} i$,
- 106 ■ a symbol $\pi: (i: \mathbb{N}) \rightarrow (j: \mathbb{N}) \rightarrow (A: \mathbf{U} i) \rightarrow (\mathbf{El} i A \rightarrow \mathbf{U} j): \mathbf{U} (\mathbf{imax} i j)$ and a rewrite rule
 107 $\mathbf{El} _ (\pi i j A B) \longrightarrow (x: \mathbf{El} i A) \rightarrow \mathbf{El} j (B x)$.

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

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

116 Level variables

117 We extend the syntax of the levels with variables.

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

$$119 \quad t := 0 \mid s(t) \mid \max(t, t) \mid \text{imax}(t, t) \mid x$$

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

122 ► **Definition 8** (Valuation). *A valuation is a function $\sigma: \mathcal{X} \rightarrow \mathbb{N}$.*

123 ► **Definition 9.** *Let $\sigma: \mathcal{X} \rightarrow \mathbb{N}$ be a valuation. We define inductively the value of a level t*
 124 *over σ , denoted as $\llbracket t \rrbracket_\sigma$ with*

$$125 \quad \llbracket 0 \rrbracket_\sigma = 0 \quad \llbracket s(t) \rrbracket_\sigma = s(\llbracket t \rrbracket_\sigma) \quad \llbracket x \rrbracket_\sigma = \sigma(x)$$

$$126 \quad \llbracket \max(t_1, t_2) \rrbracket_\sigma = \max(\llbracket t_1 \rrbracket_\sigma, \llbracket t_2 \rrbracket_\sigma) \quad \llbracket \text{imax}(t_1, t_2) \rrbracket_\sigma = \text{imax}(\llbracket t_1 \rrbracket_\sigma, \llbracket t_2 \rrbracket_\sigma)$$

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

132 ► **Definition 10** (Level comparison). *Let $t_1, t_2 \in \mathcal{L}$. We say that $t_1 \leq_{\mathcal{L}} t_2$ if for all valuations*
 133 *σ , $\llbracket t_1 \rrbracket_\sigma \leq \llbracket t_2 \rrbracket_\sigma$. In the same way, we say that $t_1 =_{\mathcal{L}} t_2$ if for all valuations σ , $\llbracket t_1 \rrbracket_\sigma = \llbracket t_2 \rrbracket_\sigma$.*
 134 *Hence $t_1 =_{\mathcal{L}} t_2$ if and only if $t_1 \leq_{\mathcal{L}} t_2$ and $t_2 \leq_{\mathcal{L}} t_1$.*

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

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

144 Related work

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

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

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

159 Contribution and outline

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

$$164 \quad \max(u_1, \dots, u_n) =_{\mathcal{L}} \max(v_1, \dots, v_m) \iff \{u_1, \dots, u_n\} = \{v_1, \dots, v_m\}.$$

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

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

174 2 Universe representation in impredicative hierarchy

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

178 2.1 Levels as maximum

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

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

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

- 184 ■ If $\llbracket v \rrbracket_{\sigma} = \llbracket w \rrbracket_{\sigma} = 0$, then $\llbracket \max(t_1, t_2) \rrbracket_{\sigma} = 0 = \llbracket t \rrbracket_{\sigma}$.
- 185 ■ If $\llbracket v \rrbracket_{\sigma} \neq 0$ and $\llbracket w \rrbracket_{\sigma} = 0$, then $\llbracket \max(t_1, t_2) \rrbracket_{\sigma} = \max(\llbracket u \rrbracket_{\sigma}, \llbracket v \rrbracket_{\sigma}) = \llbracket t \rrbracket_{\sigma}$.
- 186 ■ If $\llbracket v \rrbracket_{\sigma} = 0$ and $\llbracket w \rrbracket_{\sigma} \neq 0$, then $\llbracket \max(t_1, t_2) \rrbracket_{\sigma} = \max(\llbracket u \rrbracket_{\sigma}, \llbracket w \rrbracket_{\sigma}) = \llbracket t \rrbracket_{\sigma}$.
- 187 ■ Else, $\llbracket \max(t_1, t_2) \rrbracket_{\sigma} = \max(\llbracket u \rrbracket_{\sigma}, \llbracket v \rrbracket_{\sigma}, \llbracket w \rrbracket_{\sigma}) = \llbracket t \rrbracket_{\sigma}$.

188 ◀

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

190 **Proof.** Let σ be a valuation.

- 191 ■ If $\llbracket w \rrbracket_{\sigma} = 0$, then $\llbracket \max(\text{imax}(u, w), \text{imax}(v, w)) \rrbracket_{\sigma} = 0 = \llbracket \text{imax}(\max(u, v), w) \rrbracket_{\sigma}$.
- 192 ■ Else, $\llbracket \max(\text{imax}(u, w), \text{imax}(v, w)) \rrbracket_{\sigma} = \max(\llbracket u \rrbracket_{\sigma}, \llbracket v \rrbracket_{\sigma}, \llbracket w \rrbracket_{\sigma}) = \llbracket \text{imax}(\max(u, v), w) \rrbracket_{\sigma}$.

193 ◀

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

196 ► **Theorem 13.** *For all $t \in \mathcal{L}$, there exists u_1, \dots, u_n in the grammar $t := 0 \mid s(t) \mid \text{imax}(t, t) \mid$
 197 x such that $t =_{\mathcal{L}} \max(u_1, \dots, u_n)$.*

198 **2.2 Simplification of the levels**

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

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

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

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

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

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

211 **Proof.** Let σ be a valuation.

212 ■ If $\llbracket w \rrbracket_{\sigma} = 0$, then $\llbracket \text{max}(\text{imax}(u, w), \text{imax}(u, w)) \rrbracket_{\sigma} = 0 = \llbracket \text{imax}(u, \text{imax}(v, w)) \rrbracket_{\sigma}$.

213 ■ Else, $\llbracket \text{max}(\text{imax}(u, w), \text{imax}(v, w)) \rrbracket_{\sigma} = \text{max}(\llbracket u \rrbracket_{\sigma}, \llbracket v \rrbracket_{\sigma}, \llbracket w \rrbracket_{\sigma}) = \llbracket \text{imax}(u, \text{imax}(v, w)) \rrbracket_{\sigma}$.

214 ◀

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

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

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

222 **Proof.** Let σ be a valuation.

223 ■ If $\llbracket w \rrbracket_{\sigma} = 0$, then $\llbracket s(\text{imax}(v, w)) \rrbracket_{\sigma} = s(0) = \llbracket \text{max}(s(w), \text{imax}(s(v), w)) \rrbracket_{\sigma}$.

224 ■ Else $\llbracket s(\text{imax}(v, w)) \rrbracket_{\sigma} = s(\text{max}(\llbracket v \rrbracket_{\sigma}, \llbracket w \rrbracket_{\sigma})) = \llbracket \text{max}(s(w), \text{imax}(s(v), w)) \rrbracket_{\sigma}$.

225 ◀

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

227 ► **Theorem 18.** *For all $u \in \mathcal{L}$, there exists u_1, \dots, u_n in the grammar $t := s^k(x) \mid s^k(0) \mid$
228 $\text{imax}(t, x)$ such that $u =_{\mathcal{L}} \text{max}(u_1, \dots, u_n)$.*

229 ► **Remark 19.** For all t in the grammar of Theorem 18, there exists $x_1, \dots, x_n \in \mathcal{X}$, and
230 $v = s^k(0)$ or $v = s^k(x)$ such that $t = \text{imax}(\text{imax}(\text{imax}(\dots \text{imax}(v, x_1), x_2) \dots), x_{n-1}), x_n)$.

231 **2.3 Introducing new levels**

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

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

236 To better understand the issue, we should have a deep understanding of the semantic of
 237 $\text{imax}(x, y)$. It means that we always consider the value of y , but we only consider the value
 238 of x if y is not zero: the position of the variables is essential in the levels. But, taking into
 239 account $\text{imax}(y, x)$ leads to offset this, and the value of x and y are always considered.

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

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

$$245 \quad \llbracket \mathcal{A}(E, x, S) \rrbracket_\sigma = \begin{cases} 0 & \text{if there exists } y \in E \text{ such that } \sigma(y) = 0 \\ \llbracket x \rrbracket_\sigma + S & \text{else} \end{cases}$$

$$246 \quad \llbracket \mathcal{B}(E, S) \rrbracket_\sigma = \begin{cases} 0 & \text{if there exists } y \in E \text{ such that } \sigma(y) = 0 \\ S & \text{else} \end{cases}$$

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

249 ► **Proposition 22.** *If $t = \text{imax}(\text{imax}(\text{imax}(\dots \text{imax}(s^k(y), x_1) \dots)), x_{n-1}), x_n)$, then*

$$250 \quad t =_{\mathcal{L}} \max(\mathcal{A}(\emptyset, x_n, 0), \mathcal{A}(\{x_n\}, x_{n-1}, 0), \dots, \mathcal{A}(\{x_2, \dots, x_n\}, x_1, 0), \mathcal{A}(\{x_1, \dots, x_n\}, y, k)).$$

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

$$252 \quad t =_{\mathcal{L}} \max(\mathcal{A}(\emptyset, x_n, 0), \mathcal{A}(\{x_n\}, x_{n-1}, 0), \dots, \mathcal{A}(\{x_2, \dots, x_n\}, x_1, 0), \mathcal{B}(\{x_1, \dots, x_n\}, k)).$$

253 **Proof.** We show the result for the first case. Let σ be a valuation. If for all $1 \leq i \leq n$,
 254 $\sigma(x_i) \neq 0$, then

$$255 \quad \llbracket t \rrbracket_\sigma = \max(\sigma(x_n), \dots, \sigma(x_1), k + \sigma(y))$$

$$256 \quad \forall 1 < i \leq n, \llbracket \mathcal{A}(\{x_n, \dots, x_{i+1}\}, x_i, 0) \rrbracket_\sigma = \sigma(x_i)$$

$$257 \quad \llbracket \mathcal{A}(\{x_n, \dots, x_1\}, y, k) \rrbracket_\sigma = k + \sigma(y)$$

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

$$259 \quad \llbracket t \rrbracket_\sigma = \max(\sigma(x_n), \dots, \sigma(x_{i+1})) \quad \forall 1 < j \leq i, \llbracket \mathcal{A}(\{x_n, \dots, x_{j+1}\}, x_j, 0) \rrbracket_\sigma = 0$$

$$260 \quad \llbracket \mathcal{A}(\{x_n, \dots, x_1\}, y, k) \rrbracket_\sigma = 0 \quad \forall i < j \leq n, \llbracket \mathcal{A}(\{x_n, \dots, x_{j+1}\}, x_j, 0) \rrbracket_\sigma = \sigma(x_j)$$

261 hence the equality.

262 And the second case is very similar. Let σ be a valuation. If for all $1 \leq i \leq n$, $\sigma(x_i) \neq 0$,
 263 then

$$264 \quad \llbracket t \rrbracket_\sigma = \max(\sigma(x_n), \dots, \sigma(x_1), k)$$

$$265 \quad \forall 1 < i \leq n, \llbracket \mathcal{A}(\{x_n, \dots, x_{i+1}\}, x_i, 0) \rrbracket_\sigma = \sigma(x_i)$$

$$266 \quad \llbracket \mathcal{B}(\{x_n, \dots, x_1\}, k) \rrbracket_\sigma = k$$

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

$$268 \quad \llbracket t \rrbracket_\sigma = \max(\sigma(x_n), \dots, \sigma(x_{i+1})) \quad \forall 1 < j \leq i, \llbracket \mathcal{A}(\{x_n, \dots, x_{j+1}\}, x_j, 0) \rrbracket_\sigma = 0$$

$$269 \quad \llbracket \mathcal{B}(\{x_n, \dots, x_1\}, k) \rrbracket_\sigma = 0 \quad \forall i < j \leq n, \llbracket \mathcal{A}(\{x_n, \dots, x_{j+1}\}, x_j, 0) \rrbracket_\sigma = \sigma(x_j)$$

270 hence the equality. ◀

23:8 Encoding impredicative hierarchy of type universes with variables

271 The two equivalences are quite similar, the difference being in the very last subterm of
 272 the max which is a \mathcal{A} in the first case (we have to take into account $s^k(y)$ that is to say
 273 $k + y$) and a \mathcal{B} in the second one (k is taken into account with a \mathcal{B}).

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

277 ► **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
 278 valuation σ , $\llbracket t_1 \rrbracket_{\sigma} = \sigma(x) = \llbracket t_2 \rrbracket_{\sigma}$.

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

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

$$282 \quad \mathcal{A}(E, x, S) =_{\mathcal{L}} \max(\mathcal{A}(E \cup \{x\}, x, S), \mathcal{B}(E, S)).$$

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

284 ■ If there exists $y \in E$ such that $\sigma(y) = 0$, then $\llbracket t \rrbracket_{\sigma} = \llbracket u \rrbracket_{\sigma} = \llbracket v \rrbracket_{\sigma} = 0$.

285 ■ Else, if $\sigma(x) = 0$, then $\llbracket t \rrbracket_{\sigma} = S$, $\llbracket u \rrbracket_{\sigma} = 0$ and $\llbracket v \rrbracket_{\sigma} = S$.

286 ■ Else, $\sigma(x) \neq 0$, and then $\llbracket t \rrbracket_{\sigma} = \sigma(x) + S$, $\llbracket u \rrbracket_{\sigma} = \sigma(x) + S$ and $\llbracket v \rrbracket_{\sigma} = S$.

287 Hence the result. ◀

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

289 The second modification is related to the representation of 0. Indeed, for all $E \subset \mathcal{X}$,
 290 $\mathcal{B}(E, 0) =_{\mathcal{L}} 0$, and we should then keep at most one of them. However, since we already have
 291 $0 =_{\mathcal{L}} \max(\emptyset)$, we can remove all of them.

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

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

294 1. $\mathcal{A}(E, x, S) \in \mathcal{L}_s \iff x \in E$,

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

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

297 ► **Remark 27.** By convenience, we will note $u_i \in t$ if there exists $u_1, \dots, u_n \in \mathcal{L}_s$ such that
 298 $t =_{\mathcal{L}} \max(u_1, \dots, u_n)$.

299 **3 A minimal representation**

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

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

307 1. $t = \max(u_1, \dots, u_n)$,

308 2. for all $i \neq j$, u_i and u_j are incomparable.

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

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

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

$$315 \quad \mathcal{A}(E, x, S) \not\leq_{\mathcal{L}} \mathcal{B}(F, K) \tag{1}$$

$$316 \quad \mathcal{B}(E, S) \leq_{\mathcal{L}} \mathcal{B}(F, K) \iff F \subset E \wedge S \leq K \tag{2}$$

$$317 \quad \mathcal{B}(E, S) \leq_{\mathcal{L}} \mathcal{A}(F, x, K) \iff (F \subset E \wedge S \leq K + 1) \tag{3}$$

$$318 \quad \mathcal{A}(E, x, S) \leq_{\mathcal{L}} \mathcal{A}(F, y, K) \iff F \subset E \wedge x = y \wedge S \leq K \tag{4}$$

319 **Proof.** With σ such that $\sigma(x) = K + 1$ and $\sigma(y) = 1$ if $y \neq x$, we show the first case. Indeed,
 320 $\llbracket \mathcal{A}(E, x, S) \rrbracket_{\sigma} = K + 1 + S > K = \llbracket \mathcal{B}(F, K) \rrbracket_{\sigma}$ hence $\mathcal{A}(E, x, S) \not\leq_{\mathcal{L}} \mathcal{B}(F, K)$. The cases 2, 3
 321 and 4 correspond to Propositions 30–32 proved below. ◀

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

$$323 \quad \mathcal{A}(E, x, S) \leq_{\mathcal{L}} \mathcal{A}(F, y, K) \iff F \subset E \wedge x = y \wedge S \leq K.$$

324 **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
 325 $S \leq K$. Let σ be a valuation.

326 ■ If there exists $y \in F$ such that $\sigma(y) = 0$, then $\llbracket t_2 \rrbracket_{\sigma} = 0$ and since $F \subset E$, $\llbracket t_1 \rrbracket_{\sigma} = 0$.

327 ■ Else, $\llbracket t_1 \rrbracket_{\sigma} \leq \sigma(x) + S \leq \sigma(x) + K = \llbracket t_2 \rrbracket_{\sigma}$.

328 In both cases, $\llbracket t_1 \rrbracket_{\sigma} \leq \llbracket t_2 \rrbracket_{\sigma}$ hence $t_1 \leq_{\mathcal{L}} t_2$.

329 Now, we show the other implication by contraposition.

330 ■ If there exists $z \in F$ such that $z \notin E$, we take σ such that $\sigma(z) = 0$ and for all $j \neq z$,
 331 $\sigma(j) = 1$. We note that $z \neq x$ (since $z \notin E$ and $x \in E$) hence $\sigma(x) = 1$. Then,
 332 $\llbracket t_1 \rrbracket_{\sigma} = S + 1 > 0 = \llbracket t_2 \rrbracket_{\sigma}$.

333 ■ If $x \neq y$ we take σ such that $\sigma(x) = K + 2$, $\sigma(y) = 1$ and for all $z \neq x$ and $z \neq y$, $\sigma(z) = 1$.
 334 Then, $\llbracket t_1 \rrbracket_{\sigma} = K + S + 2 > K + 1 = \llbracket t_2 \rrbracket_{\sigma}$.

335 ■ If $S > K$ we take σ such that for all z , $\sigma(z) = 1$. Then, $\llbracket t_1 \rrbracket_{\sigma} = S + 1 > K + 1 = \llbracket t_2 \rrbracket_{\sigma}$.
 336 ◀

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

$$338 \quad \mathcal{B}(E, S) \leq_{\mathcal{L}} \mathcal{B}(F, K) \iff F \subset E \wedge S \leq K.$$

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

341 ■ If there exists $y \in F$ such that $\sigma(y) = 0$, then $\llbracket t_2 \rrbracket_{\sigma} = 0$ and since $F \subset E$, $\llbracket t_1 \rrbracket_{\sigma} = 0$.

342 ■ Else, $\llbracket t_1 \rrbracket_{\sigma} \leq K \leq S = \llbracket t_2 \rrbracket_{\sigma}$.

343 In both cases, $\llbracket t_1 \rrbracket_{\sigma} \leq \llbracket t_2 \rrbracket_{\sigma}$ hence $t_1 \leq_{\mathcal{L}} t_2$.

344 Now, we show the other implication by contraposition.

345 ■ If there exists $y \in F$ such that $y \notin E$, we take σ such that $\sigma(y) = 0$ and for all $z \neq y$,
 346 $\sigma(z) = 1$. Then, $\llbracket t_1 \rrbracket_{\sigma} = S > 0 = \llbracket t_2 \rrbracket_{\sigma}$.

347 ■ If $S < K$ we take σ such that for all y , $\sigma(y) = 1$. Then, $\llbracket t_2 \rrbracket_{\sigma} = K > S = \llbracket t_1 \rrbracket_{\sigma}$.
 348 ◀

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

$$350 \quad \mathcal{B}(E, S) \leq_{\mathcal{L}} \mathcal{A}(F, x, K) \iff (F \subset E \wedge S \leq K + 1).$$

23:10 Encoding impredicative hierarchy of type universes with variables

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

353 ■ If there exists $y \in F$ such that $\sigma(y) = 0$, then $\llbracket t_2 \rrbracket_\sigma = 0$ and since $F \subset E$, $\llbracket t_1 \rrbracket_\sigma = 0$.

354 ■ Else, $\sigma(x) \geq 1$ (because $x \in F$) and then $\llbracket t_2 \rrbracket_\sigma = \sigma(x) + K \geq 1 + K \geq S \geq \llbracket t_1 \rrbracket_\sigma$.

355 In both cases, $\llbracket t_1 \rrbracket_\sigma \leq \llbracket t_2 \rrbracket_\sigma$ hence $t_1 \leq_{\mathcal{L}} t_2$.

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

357 ■ If there exists $y \in F$ such that $y \notin E$, we take σ such that $\sigma(y) = 0$ and for all $z \neq y$,
 358 $\sigma(z) = 1$. Then, $\llbracket t_1 \rrbracket_\sigma = K > 0 = \llbracket t_2 \rrbracket_\sigma$.

359 ■ If $S > K + 1$ we take σ such that for all y , $\sigma(y) = 1$. Then, $\llbracket t_1 \rrbracket_\sigma = S > K + 1 = \llbracket t_2 \rrbracket_\sigma$.

360 ◀

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

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

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

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

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

$$368 \quad \mathcal{A}(E, x, S) \in t_1 \iff \mathcal{A}(E, x, S) \in t_2.$$

369 **Proof.** Let us note

$$370 \quad t_1 = \max(\mathcal{A}(E_0, x_0, S_0), \dots, \mathcal{A}(E_n, x_n, S_n), \mathcal{B}(G_0, T_0), \dots, \mathcal{B}(G_p, T_p))$$

$$371 \quad t_2 = \max(\mathcal{A}(F_0, y_0, K_0), \dots, \mathcal{A}(F_m, y_m, K_m), \mathcal{B}(H_0, L_0), \dots, \mathcal{B}(H_q, L_q)).$$

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

$$373 \quad \sigma(y) = \begin{cases} \max(S_0, \dots, S_n, K_0, \dots, K_m, T_0, \dots, T_p, L_0, \dots, L_q) + 1 & \text{if } y = x \\ 1 & \text{if } y \in E \setminus \{x\} \\ 0 & \text{else} \end{cases}$$

374 We have $\llbracket t_1 \rrbracket_\sigma = \llbracket \mathcal{A}(E, x, S) \rrbracket_\sigma = S + \sigma(x)$ and then $\llbracket t_2 \rrbracket_\sigma = S + \sigma(x)$. Then, there exists
 375 $\mathcal{A}(F, y, K)$ in t_2 such that $F \subset E \cup \{x\} = E$ (else F contains a variable z such that $\sigma(z) = 0$)
 376 and $\sigma(y) + K = \sigma(x) + S$ or there exists $\mathcal{B}(F, K)$ in t_2 such that $K = \sigma(x) + S$. Since
 377 $\sigma(x) > \max(S_0, \dots, S_n, K_0, \dots, K_m)$, we deduce that it is the first case and $y = x$.

378 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
 379 the same reasoning, we show that there exists $\mathcal{A}(G, x, S) \in t_1$ with $G \subset F \subsetneq E$. But, by
 380 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
 381 are comparable.

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

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

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

$$385 \quad \mathcal{B}(E, S) \in t_1 \iff \mathcal{B}(E, S) \in t_2.$$

386 **Proof.** Let us note

$$387 \quad t_1 = \max(\mathcal{A}(E_0, x_0, S_0), \dots, \mathcal{A}(E_n, x_n, S_n), \mathcal{B}(G_0, T_0), \dots, \mathcal{B}(G_p, T_p))$$

$$388 \quad t_2 = \max(\mathcal{A}(F_0, y_0, K_0), \dots, \mathcal{A}(F_m, y_m, K_m), \mathcal{B}(H_0, L_0), \dots, \mathcal{B}(H_q, L_q)).$$

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

392 In the induction case, we consider σ such that $\sigma(x) = 1$ if $x \in E$ and $\sigma(x) = 0$ otherwise,
 393 hence $\llbracket t_1 \rrbracket_\sigma = S$. Then, $\llbracket t_2 \rrbracket_\sigma = S$ and since $S > 0$, there exists $\mathcal{A}(F, x, K)$ in t_2 such that
 394 $F \subset E$ and $\sigma(x) + K = S$ or there exists $\mathcal{B}(F, S)$ in t_2 such that $F \subset E$ and $K = S$.

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

398 Then, we have $\mathcal{B}(F, S) \in t_2$. If $F \subsetneq E$, we apply the induction hypothesis and obtain
 399 $\mathcal{B}(F, S) \in t_1$, impossible because it would be comparable with $\mathcal{B}(E, S)$.

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

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

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

404 **Proof.** The reverse implication is trivial and the first one is a consequences of the Proposi-
 405 tions 34 and 35. ◀

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

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

411 **Proof.** We apply the following procedure. We consider $E = \{u_1, \dots, u_n\}$. While there exists
 412 $u, v \in E$ such that $u \leq_{\mathcal{L}} v$, we remove u from E . The Proposition 36 permits to obtain the
 413 uniqueness of this minimal representation. ◀

414 ▶ **Theorem 38 (Representation).** For all $t \in \mathcal{L}$, there exists a unique $\{u_1, \dots, u_n\} \subset \mathcal{L}_s$ such
 415 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
 416 and we denote it as $\text{repr}(t)$.

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

419 Having a unique minimal representation is useful to have a faithful embedding of \mathcal{L} in the
 420 $\lambda\Pi/\equiv$, which is done in Section 4. Moreover, this representation also gives us an easy way to
 421 compare two terms. Indeed, a sublevel can be compared to a level using the representation.

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

424 **Proof.** The reverse implication is trivial. We show the direct implication by contraposition.
 425 We suppose that for all i , $u \not\leq_{\mathcal{L}} v_i$.

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

23:12 Encoding impredicative hierarchy of type universes with variables

- 428 ■ $v_i = \mathcal{A}(F, x, K)$ or $v_i = \mathcal{B}(F, K)$ and $F \not\subseteq E$ hence $\llbracket v_i \rrbracket_\sigma = 0 < S = \llbracket u \rrbracket_\sigma$,
429 ■ or $v_i = \mathcal{A}(F, x, K)$ with $F \subseteq E$ and $K < S - 1$ hence $\llbracket v_i \rrbracket_\sigma = K + 1 < S = \llbracket u \rrbracket_\sigma$,
430 ■ or $v_i = \mathcal{B}(F, K)$ with $K < S - 1$ hence $\llbracket v_i \rrbracket_\sigma = K < S = \llbracket t \rrbracket_\sigma$.

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

432 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
433 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
434 otherwise. Then, for all v_i , we have either

- 435 ■ $v_i = \mathcal{B}(F, K)$ hence $\llbracket v_i \rrbracket_\sigma \leq K < S + M + 2 = \llbracket u \rrbracket_\sigma$,
436 ■ or $v_i = \mathcal{A}(F, y, K)$ and $F \not\subseteq E$ hence $\llbracket v_i \rrbracket_\sigma = 0 < S = \llbracket u \rrbracket_\sigma$,
437 ■ or $v_i = \mathcal{A}(F, y, K)$ with $F \subseteq E$ and $x \neq y$, hence $\llbracket v_i \rrbracket_\sigma = K + 1 < S + M + 2 = \llbracket u \rrbracket_\sigma$,
438 ■ or $v_i = \mathcal{A}(F, x, K)$ with $F \subseteq E$ and $K < S$, hence $\llbracket v_i \rrbracket_\sigma = K + M + 2 < S + M + 2 = \llbracket u \rrbracket_\sigma$.
439 Then $u \not\leq_{\mathcal{L}} \max(v_1, \dots, v_n)$. ◀

440 And we use this lemma to compare two levels, for instance by comparing each sublevel
441 of the minimal representation of the first one to the second one. More generally, two
442 representations are compared in the following way.

443 ► **Theorem 40.** *Let $u_1, \dots, u_n, v_1, \dots, v_m \in \mathcal{L}_s$. Then, $\max(u_1, \dots, u_n) \leq_{\mathcal{L}} \max(v_1, \dots, v_m)$
444 if and only if for all i , there exists j such that $u_i \leq_{\mathcal{L}} v_j$.*

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

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

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

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

454 This shows that there is a link between the Lemma 39 and the uniqueness property. In
455 fact, this lemma should be understood as an *independence* lemma. Indeed, if we consider
456 $\max(u_1, \dots, u_n)$ as a linear combination of u_1, \dots, u_n , then this lemma states that the only
457 way to be smaller than a linear combination is to depend on and be smaller than one of the
458 elements of this combination.

459 This analogy provides a new point of view on our work: \mathcal{L}_s is a ‘linearly independent’
460 family (uniqueness of the minimal representation) which generates at least \mathcal{L} (for instance,
461 $\max(\mathcal{A}(\{x\}, y, 0)$ or $\max(\mathcal{B}(\{x\}, 1))$ are not equivalent to any level).

462 **4 A rewriting system for this universe representation**

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

464 **4.1 Basic tools**

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

467 ► **Definition 41** (Booleans). We define a type B , with constructors $\text{true}: B$, $\text{false}: B$ and
 468 functions $\text{and}: B \rightarrow B$, $\text{or}: B \rightarrow B$ and $\text{not}: B \rightarrow B$. B is interpreted as the Booleans, true ,
 469 false , and , or and not as \top , \perp , the conjunction, the disjunction and the negation.

470 ► **Definition 42** (Natural numbers). We define a type N , with constructors $0: N$, $s: N \rightarrow N$
 471 and functions $+: N \rightarrow N \rightarrow N$, $<=_N: N \rightarrow N \rightarrow B$, $=_N: N \rightarrow N \rightarrow B$, $<_N: N \rightarrow N \rightarrow B$ (with infix
 472 notation) and $\text{max}_N: N \rightarrow N \rightarrow N$. N is interpreted as \mathbb{N} , 0 as 0 , s , $+$, $<=_N$, $=_N$ and max_N as 0 ,
 473 s , $+$, \leq , $=$ and max .

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

476 ► **Definition 43.** Let T be a type. We define the function $\text{ite}_T: B \rightarrow T \rightarrow T \rightarrow T$ such that
 477 $\forall u, v \in T$, $\text{ite}_T \text{ true } u \ v \hookrightarrow u$ and $\text{ite}_T \text{ false } u \ v \hookrightarrow v$. For convenience reasons, we will
 478 denote $\text{ite}_T b \ u \ v$ by *if b then u else v* .

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

482 ► **Definition 44** (Sets). Let T be a type equipped with a total order function $\text{leq}_T: T \rightarrow$
 483 $T \rightarrow B$. Then, we define a type $S[T]$ corresponding to the finite set of elements of T with a
 484 constructor $\{\}_T: S[T]$ and the functions (all with infix notation) $\ll_T: S[T] \rightarrow T \rightarrow S[T]$,
 485 $++_T: S[T] \rightarrow S[T] \rightarrow S[T]$, $\text{in}_T: T \rightarrow S[T] \rightarrow B$, $\text{sub}_T: S[T] \rightarrow S[T] \rightarrow B$ and
 486 $=_{S[T]}: S[T] \rightarrow S[T] \rightarrow B$.

487 $S[T]$ is interpreted as the set of finite subsets of T , $\{\}_T$ as the empty set, \ll_T as the
 488 function that adds an element to a set, $++_T$, in_T , sub_T and $=_{S[T]}$ as \cup , \in , \subseteq and $=$.

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

491 $S[T]$ is implemented as a sorted list of elements of T with the constructors $\{\}_T: S[T]$
 492 and $::_T: T \rightarrow S[T] \rightarrow S[T]$, and the function \ll_T adds an element to a list while keeping
 493 the uniqueness and order properties. Then, a set will be built through $\{\}_T$ and \ll_T and the
 494 constructor $::_T$ will only be used in patterns of rewrite rules.

495 ► **Definition 45** (Set order). Let T be a type with a total order function leq_T . Then, we
 496 define a total order $\text{leq}_{S[T]}$ by considering the lexicographic order on sorted word of T .
 497 Besides, we define the corresponding strict total order $\text{lq}_{S[T]}$.

498 We do not give the rules for these elements since they are quite basic. However, the
 499 DEDUKTI implementation is available if necessary.

500 4.2 Level encoding

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

505 ► **Definition 46.** We define L , the type of the levels together with the constructors $0_L: L$,
 506 $s_L: L \rightarrow L$, $\text{max}_L: L \rightarrow L \rightarrow L$, $\text{imax}_L: L \rightarrow L \rightarrow L$ and $\text{var}_L: N \rightarrow L$. We define inductively a
 507 translation function $|\cdot|: \mathcal{L} \rightarrow L$ with

$$508 \quad |0| = 0_L \quad |s(t)| = s_L |t| \quad |x| = \text{var}_L |\gamma(x)|_N$$

$$509 \quad |\text{max}(u, v)| = \text{max}_L |u| |v| \quad |\text{imax}(u, v)| = \text{imax}_L |u| |v|$$

23:14 Encoding impredicative hierarchy of type universes with variables

510 ► **Definition 47.** We define L_S , the type of the sublevels, with the constructors $\mathbf{a}: S[N] \rightarrow$
 511 $N \rightarrow N \rightarrow L_S$ and $\mathbf{b}: S[N] \rightarrow N \rightarrow L_S$. We define a translation function $|\cdot|_s: \mathcal{L}_s \rightarrow L_S$ by
 512 $|\mathcal{A}(E, x, S)|_s = \mathbf{a} |E| |x| |S|$ and $|\mathcal{B}(E, S)|_s = \mathbf{b} |E| |S|$.

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

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

520 ► **Definition 48** (Total order on sublevels). We define a total leq_{L_S} on L_S (and its corresponding
 521 strict order 1q_{L_S}). Indeed, we consider these rewrite rules.

$$\begin{aligned}
 522 \quad & (\mathbf{a} E x S) \text{leq}_{L_S} (\mathbf{b} F K) \longrightarrow \text{true} \\
 523 \quad & (\mathbf{a} E x S) \text{leq}_{L_S} (\mathbf{a} F y K) \longrightarrow \text{or} (E \text{1q}_{S[N]} F) \\
 & \quad \quad \quad \text{and} (E =_{S[N]} F) ((\text{or} (x <_N y) (\text{and} (x =_N y) (S <=_N K)))) \\
 524 \quad & \quad \quad \quad \text{and} (E =_{S[N]} F) ((\text{or} (x <_N y) (\text{and} (x =_N y) (S <=_N K)))) \\
 525 \quad & \mathbf{b} E S \text{leq}_{L_S} \mathbf{b} F K \longrightarrow \text{or} (E \text{1q}_{S[N]} F, \text{and}(E =_{S[N]} F, S <=_N K))
 \end{aligned}$$

526 And we can then define the representations.

527 ► **Definition 49.** We define a function $\text{max}_{L_S}: S[L_S] \rightarrow L$ that embeds a set of L_S into a L and a
 528 translation function $|\cdot|_r: \mathcal{L}_r \rightarrow L$ defined by $|\text{max}(u_1, \dots, u_n)|_r = \text{max}_{L_S}(\{|u_1|_s, \dots, |u_n|_s\}_T)$.

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

$$532 \quad 0_L \longrightarrow \text{max}_{L_S}(\{\}_{L_S}) \quad \text{var}_L(x) \longrightarrow \text{max}_{L_S}(\{\mathbf{a}(\{x\}_N, x, 0)\}_{L_S})$$

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

535 ► **Definition 50** (Sublevels comparison). We define $<=_L: L_S \rightarrow L_S \rightarrow B$ interpreted as $\leq_{\mathcal{L}}$
 536 with these rewrite rules.

$$537 \quad \mathbf{a}(E, x, S) <=_L \mathbf{b}(F, K) \longrightarrow \text{false} \tag{1}$$

$$538 \quad \mathbf{b}(E, S) <=_L \mathbf{b}(F, K) \longrightarrow \text{and}(F \text{sub}_N E, S <=_N K) \tag{2}$$

$$539 \quad \mathbf{b}(E, \mathbf{s}_L(S)) <=_L \mathbf{a}(F, y, K) \longrightarrow \text{and}(F \text{sub}_N E, S <=_N K + 1) \tag{3}$$

$$540 \quad \mathbf{a}(E, x, S) <=_L \mathbf{a}(F, y, K) \longrightarrow \text{and}(F \text{sub}_N E, \text{and}(x =_N y, S <=_N K)) \tag{4}$$

541 ► **Proposition 51.** Let $u, v \in \mathcal{L}_s$. Then, $u \leq_{\mathcal{L}} v \iff |u| <=_L |v| \hookrightarrow^* \text{true}$.

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

543 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 $s_s(\mathcal{A}(E, x, S)) = \mathcal{A}(E, x, S + 1)$ and $s_s(\mathcal{B}(E, S)) = \mathcal{B}(E, S + 1)$. Then $s_s(u) =_{\mathcal{L}} s(u)$.

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

We create a function $s_{L_s} : \mathcal{S}[L_s] \rightarrow \mathcal{S}[L_s]$ corresponding to s_s .

$$s_{L_s}(\{ \}_{L_s}) \longrightarrow \{ \}_{L_s} \quad s_{L_s}(\mathbf{b}(E, S) ::_{L_s} q) \longrightarrow s_{L_s}(q) \ll_{L_s} \mathbf{b}(E, s(S))$$

$$s_{L_s}(\mathbf{a}(E, x, S) ::_{L_s} q) \longrightarrow s_{L_s}(q) \ll_{L_s} \mathbf{a}(E, x, s(S))$$

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

$$s_L(\max_{L_s}(\{ \}_{L_s})) \longrightarrow \max_{L_s}(\mathbf{b}(\{ \}_{L_s}, s(0))) \quad s_L(\max_{L_s}(u ::_{L_s} q)) \longrightarrow \max_{L_s}(s_{L_s}(u ::_{L_s} q))$$

► **Proposition 54.** For all $t \in \mathcal{L}_r$, $s_L(|t|_r) \hookrightarrow^* |\text{repr}(s(t))|_r$.

4.4 The maximum

For the maximum, we define \max_s , that adds a sublevel to a minimal representation.

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

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

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

We implement \max_s as a function $\text{maxhelper} : \mathcal{S}[L_s] \rightarrow L_s \rightarrow \mathcal{S}[L_s]$ with these rewrite rules.

$$\text{maxhelper}(\{ \}_{L_s}, u) \longrightarrow \{u\}_{L_s}$$

$$\text{maxhelper}(u ::_{L_s} E, v) \longrightarrow \text{if } v \leq_{L_s} u \text{ then } E \ll_{L_s} u$$

$$\text{else if } u \leq_{L_s} v \text{ then } E \ll_{L_s} v \text{ else } \text{maxhelper}(E, v) \ll_{L_s} u$$

And we compute the minimal representation for \max_L according to Proposition 56.

$$\max_L(\max_{L_s}(E), \max_{L_s}(\{ \}_{L_s})) \longrightarrow \max_{L_s}(E)$$

$$\max_L(\max_{L_s}(E), \max_{L_s}(u ::_{L_s} F)) \longrightarrow \max_L(\text{maxhelper}(E, u), \max_{L_s}(F))$$

► **Proposition 57.** For all $t_1, t_2 \in \mathcal{L}_r$, $\max_L(|t_1|_r, |t_2|_r) \hookrightarrow^* |\text{repr}(\max(t_1, t_2))|_r$.

4.5 The impredicative maximum

To begin, we study $\text{imax}(u, v)$ where $u, v \in \mathcal{L}_s$.

► **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

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

23:16 Encoding impredicative hierarchy of type universes with variables

577 **Proof.** We note $u = f(E, x, S)$ and $v = g(F, K)$. Let σ be a valuation.
578 ■ If there exists $y \in F$ such that $\sigma(y) = 0$, then $\llbracket \text{imax}(u, v) \rrbracket_\sigma = 0 = \llbracket \max(f(E \cup F, S), v) \rrbracket_\sigma$.
579 ■ Else, $\llbracket v \rrbracket_\sigma = K > 0$ and then $\llbracket \text{imax}(u, v) \rrbracket_\sigma = \max(\llbracket u \rrbracket_\sigma, K)$. Besides, since $\sigma(y) \neq 0$ for all
580 $y \in F$, there exists $y \in E \cup F$ such that $\sigma(y) = 0$ if and only if there exists $y \in E$ such
581 that $\sigma(y) = 0$, hence $\llbracket u \rrbracket_\sigma = \llbracket f(E \cup F, S) \rrbracket_\sigma$.
582 Hence the result. ◀

583 We implement it as a function $\text{imax}_{L_S} : L_S \rightarrow L_S \rightarrow L$.

584 $\text{imax}_{L_S}(\mathbf{a}(E, x, S), \mathbf{b}(F, K)) \longrightarrow \max_L(\max_{L_S}(\mathbf{a}(E \text{ ++}_N F, x, S)), \max_{L_S}(\mathbf{b}(F, K)))$
585 $\text{imax}_{L_S}(\mathbf{b}(E, S), \mathbf{b}(F, K)) \longrightarrow \max_L(\max_{L_S}(\mathbf{b}(E \text{ ++}_N F, S)), \max_{L_S}(\mathbf{b}(F, K)))$
586 $\text{imax}_{L_S}(\mathbf{b}(E, S), \mathbf{a}(F, x, K)) \longrightarrow \max_L(\max_{L_S}(\mathbf{b}(E \text{ ++}_N F, S)), \max_{L_S}(\mathbf{a}(F, x, K)))$
587 $\text{imax}_{L_S}(\mathbf{a}(E, x, S), \mathbf{a}(F, y, K)) \longrightarrow \max_L(\max_{L_S}(\mathbf{a}(E \text{ ++}_N F, x, S)), \max_{L_S}(\mathbf{a}(F, y, K)))$
588

589 Then, following the equalities $\text{imax}(0, t) =_{\mathcal{L}} t$ and $\text{imax}(t, 0) =_{\mathcal{L}} 0$, and Propositions 11
590 and 12, we define $\text{imax_aux} : L_S \rightarrow L \rightarrow L$ and add these rewrite rules.

591 $\text{imax}_L(\max_{L_S}(\{\}_L), t) \longrightarrow t$
592 $\text{imax}_L(\max_{L_S}(u ::_{L_S} q), t) \longrightarrow \max_L(\text{imax_aux}(u, t), \text{imax}_L(q, t))$
593 $\text{imax_aux}(u, \max_{L_S}(\{\}_L)) \longrightarrow \max_{L_S}(\{\}_L)$
594 $\text{imax_aux}(u, \max_{L_S}(v ::_{L_S} q)) \longrightarrow \max_L(\text{imax}_{L_S}(u, v), \text{imax_aux}(u, q))$

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

596 **Proof.** By Propositions 57 and 58, for all $u, v \in \mathcal{L}_s$, $\text{imax}_{L_S}(|u|_s, |v|_s) \hookrightarrow^* |\text{repr}(\text{imax}(u, v))|_r$.
597 Then, by induction on $t \in \mathcal{L}_r$ and using Proposition 57, we show that for all $u \in \mathcal{L}_s$,
598 $\text{imax_aux}(|u|_s, |t|_r) \hookrightarrow^* |\text{repr}(\text{imax}(u, t))|_r$. And finally, we show the result by induction on
599 t_2 using Propositions 11 and 12 and the equivalences $\text{imax}(0, t) =_{\mathcal{L}} t$ and $\text{imax}(t, 0) =_{\mathcal{L}} 0$. ◀

600 4.6 Implementing the substitution

601 Since we use a deep encoding, β -reduction cannot be used for substitution. Then, we
602 implement a substitution function. First, we implement $\text{eval}_{L_S} : L_S \rightarrow N \rightarrow N \rightarrow L$ for the
603 sublevels following the semantic given in the Definition 21.

604 $\text{eval}_{L_S}(\mathbf{b}(E, S), y, n) \longrightarrow \text{if and}(y \text{ in}_N E, n =_N 0) \text{ then } \max_{L_S}(\{\}_L)$
605 $\text{else } \max_{L_S}(\mathbf{b}(E \setminus_N y, S))$
606 $\text{eval}_{L_S}(\mathbf{a}(E, x, S), y, n) \longrightarrow \text{if and}(y \text{ in}_N E, n =_N 0) \text{ then } \max_{L_S}(\{\}_L)$
607 $\text{else if } x =_N y \text{ then } \max_{L_S}(\mathbf{b}(\{\}_N, S + n))$
608 $\text{else } \max_{L_S}(\mathbf{a}(E \setminus_N y, x, S))$

609 Then, we create a function $\text{eval}_L : L \rightarrow N \rightarrow N \rightarrow L$ that evaluate a level using the fact that
610 $[\max(u_1, \dots, u_n)]\{x \mapsto n\} = \max([u_1]\{x \mapsto n\}, \dots, [u_n]\{x \mapsto n\})$.

611 $\text{eval}_L(\{\}_L, y, n) \longrightarrow \{\}_L \quad \text{eval}_L(u ::_{L_S} q, y, n) \longrightarrow \max_L(\text{eval}_{L_S}(u, y, n), \text{eval}_L(q, y, n))$

612 ► **Proposition 60.** Let $t \in \mathcal{L}$. Then, the normal form of $||t]\{x \mapsto u\}|$ is $\text{eval}_L(|t|, |u|)$.

613 4.7 Properties of the rewrite system

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

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

618 **Proof.** The termination has been proved with two termination checkers, $\mathsf{T}\mathsf{T}_2$ [26] and
619 $\mathsf{SizeChangeTool}$ [19], and the confluence with CSI [34]. ◀

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

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

622 **Proof.** We show that $t \hookrightarrow^* |\mathsf{repr}(t)|_r$ by induction on t : $\mathsf{var}_L(x) \hookrightarrow \mathsf{max}_{L_S}(\{\mathsf{a}(\{x\}_N, 0, x)\}_{L_S})$,
623 $0 \hookrightarrow \mathsf{max}_{L_S} \{\}_{L_S}$, and we show the cases s , max and imax using Propositions 54, 57, and 59.
624 Since no rewrite rule can be applied to max_{L_S} , the normal form of $|t|$ is $|\mathsf{repr}(t)|_r$. ◀

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

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

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

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

635 A solution could be to add a dependent term as argument, to check these conditions.
636 For instance, a would have the type $(E: \mathsf{S}[N]) \rightarrow (x: N) \rightarrow \mathsf{Prf} (x \mathsf{in}_N E) \rightarrow N \rightarrow L_S$ where
637 $\mathsf{Prf}: \mathsf{B} \rightarrow \mathsf{Type}$ represents the proof of a proposition. We declare $\mathsf{I}: \mathsf{Prf} \mathsf{true}$ and we use
638 $\mathsf{a} E x k \mathsf{I}$ using the fact that $x \mathsf{in}_N E$ reduces to true if and only if x is an element of E .

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

643 5 Conclusion

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

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

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

660 — References —

- 661 **1** Ali Assaf. A calculus of constructions with explicit subtyping. In Hugo Herbelin, Pierre
 662 Letouzey, and Matthieu Sozeau, editors, *20th International Conference on Types for Proofs
 663 and Programs (TYPES 2014)*, volume 39 of *LIPICS*, Institut Henri Poincaré, Paris, France,
 664 May 2014. URL: <https://hal.archives-ouvertes.fr/hal-01097401>.
- 665 **2** Ali Assaf. *A framework for defining computational higher-order logics*. Theses, École polytech-
 666 nique, 09 2015. URL: <https://pastel.archives-ouvertes.fr/tel-01235303>.
- 667 **3** Ali Assaf, Guillaume Burel, Raphaël Cauderlier, David Delahaye, Gilles Dowek, Catherine
 668 Dubois, Frédéric Gilbert, Pierre Halmagrand, Olivier Hermant, and Ronan Saillard. *Dedukti :*
 669 *a logical framework based on the $\lambda\pi$ -calculus modulo theory*. 2016.
- 670 **4** Ali Assaf, Gilles Dowek, Jean-Pierre Jouannaud, and Jiaxiang Liu. Encoding Proofs in
 671 *Dedukti: the case of Coq proofs*. In *Proceedings Hammers for Type Theories*, Proc. Higher-
 672 Order rewriting Workshop, Coimbra, Portugal, July 2016. Easy Chair. URL: [https://inria.
 673 hal.science/hal-01330980](https://inria.hal.science/hal-01330980).
- 674 **5** Henk Barendregt. Introduction to generalized type systems. *Journal of Functional Program-*
 675 *ming*, 1(2):125–154, 1991. doi:10.1017/S0956796800020025.
- 676 **6** Henk Barendregt, S. Abramsky, D. Gabbay, T. Maibaum, and Henk (Hendrik) Barendregt.
 677 *Lambda calculi with types*. 10 2000.
- 678 **7** Michael Beeson, Pierre Boutry, Gabriel Braun, Charly Gries, and Julien Narboux. *GeoCoq*,
 679 June 2018. URL: <https://inria.hal.science/hal-01912024>.
- 680 **8** Stefano Berardi. *Type dependence and constructive mathematics*. PhD thesis, PhD thesis,
 681 Dipartimento di Informatica, Torino, Italy, 1990.
- 682 **9** Frédéric Blanqui. Encoding Type Universes Without Using Matching Modulo Associativity and
 683 Commutativity. In Amy P. Felty, editor, *7th International Conference on Formal Structures for
 684 Computation and Deduction (FSCD 2022)*, volume 228 of *Leibniz International Proceedings in
 685 Informatics (LIPIcs)*, pages 24:1–24:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-
 686 Zentrum für Informatik. URL: [https://drops.dagstuhl.
 687 de/opus/volltexte/2022/16305](https://drops.dagstuhl.de/opus/volltexte/2022/16305), doi:10.4230/LIPIcs.FSCD.2022.24.
- 688 **10** Frédéric Blanqui, Gilles Dowek, Émilie Grienemberger, Gabriel Hondet, and François Thiré.
 689 Some Axioms for Mathematics. In Naoki Kobayashi, editor, *6th International Conference
 690 on Formal Structures for Computation and Deduction (FSCD 2021)*, volume 195 of *Leibniz
 691 International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:19, Dagstuhl, Germany,
 692 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: [https://drops.dagstuhl.
 693 de/opus/volltexte/2021/14258](https://drops.dagstuhl.de/opus/volltexte/2021/14258), doi:10.4230/LIPIcs.FSCD.2021.20.
- 694 **11** Mathieu Boespflug and Guillaume Burel. Coqine: Translating the calculus of inductive
 695 constructions into the $\lambda\pi$ -calculus modulo. In *in "Second International Workshop on Proof
 696 Exchange for Theorem Proving*, 2012.
- 697 **12** Mathieu Boespflug, Quentin Carbonneaux, and Olivier Hermant. The $\lambda\Pi$ -calculus Modulo as
 698 a Universal Proof Language. *CEUR Workshop Proceedings*, 878, 06 2012.
- 699 **13** Thierry Coquand and Christine Paulin. Inductively defined types. In Per Martin-Löf and
 700 Grigori Mints, editors, *COLOG-88*, pages 50–66, Berlin, Heidelberg, 1990. Springer Berlin
 701 Heidelberg.
- 702 **14** Judicaël Courant. Explicit universes for the calculus of constructions. In Victor A. Carreño,
 703 César A. Muñoz, and Sofiène Tahar, editors, *Theorem Proving in Higher Order Logics*, pages
 704 115–130, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

- 705 15 Denis Cousineau and Gilles Dowek. Embedding pure type systems in the lambda-pi-calculus
706 modulo. pages 102–117, 06 2007. doi:10.1007/978-3-540-73228-0_9.
- 707 16 Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In Jan van Leeuwen, editor,
708 *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages
709 243–320. Elsevier and MIT Press, 1990. doi:10.1016/b978-0-444-88074-1.50011-1.
- 710 17 Gaspard Férey. *Higher-Order Confluence and Universe Embedding in the Logical Framework*.
711 (*Confluence d'ordre supérieur et encodage d'univers dans le Logical Framework*. PhD thesis,
712 École normale supérieure Paris-Saclay, France, 2021. URL: [https://lmf.cnrs.fr/downloads/
713 Perso/Ferey-thesis.pdf](https://lmf.cnrs.fr/downloads/Perso/Ferey-thesis.pdf).
- 714 18 Michael Färber. Safe, fast, concurrent proof checking for the lambda-pi calculus modulo
715 rewriting. pages 225–238, 01 2022. doi:10.1145/3497775.3503683.
- 716 19 Guillaume Genestier. SizeChangeTool: A Termination Checker for Rewriting Dependent Types.
717 In Mauricio Ayala-Rincón, Silvia Ghilezan, and Jakob Grue Simonsen, editors, *HOR 2019*
718 *- 10th International Workshop on Higher-Order Rewriting*, Joint Proceedings of HOR 2019
719 and IWC 2019, pages 14–19, Dortmund, Germany, June 2019. URL: [https://hal.science/
720 hal-02442465](https://hal.science/hal-02442465).
- 721 20 Guillaume Genestier. Encoding Agda Programs Using Rewriting. In Zena M. Ariola, editor, *5th*
722 *International Conference on Formal Structures for Computation and Deduction (FSCD 2020)*,
723 volume 167 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:17,
724 Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: [https://drops.
725 dagstuhl.de/opus/volltexte/2020/12353](https://drops.dagstuhl.de/opus/volltexte/2020/12353), doi:10.4230/LIPIcs.FSCD.2020.31.
- 726 21 Yoan Gérard. *Stt∀ geocoq*, 2021. URL: https://github.com/Karnaj/sttfa_geocoq_euclid.
- 727 22 Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *J. ACM*,
728 40(1):143–184, 1 1993. doi:10.1145/138027.138060.
- 729 23 Robert Harper and Robert Pollack. Type checking with universes. In *2nd International Joint*
730 *Conference on Theory and Practice of Software Development*, TAPSOFT '89, page 107–136,
731 NLD, 1991. Elsevier Science Publishers B. V.
- 732 24 Gabriel Hondet and Frédéric Blanqui. The New Rewriting Engine of Dedukti (System
733 Description). In Zena M. Ariola, editor, *5th International Conference on Formal Structures for*
734 *Computation and Deduction (FSCD 2020)*, volume 167 of *Leibniz International Proceedings in*
735 *Informatics (LIPIcs)*, pages 35:1–35:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-
736 Zentrum für Informatik. URL: <https://drops.dagstuhl.de/opus/volltexte/2020/12357>,
737 doi:10.4230/LIPIcs.FSCD.2020.35.
- 738 25 Gérard Huet. Extending the calculus of constructions with type: Type, 1988. Unpublished
739 draft. URL: <https://pauillac.inria.fr/~huet/PUBLIC/typtyp.pdf>.
- 740 26 Martin Korp, Christian Sternagel, Harald Zankl, and Aart Middeldorp. Tyrolean ter-
741 mination tool 2. In Ralf Treinen, editor, *Rewriting Techniques and Applications, 20th*
742 *International Conference, RTA 2009, Brasilia, Brazil, June 29 - July 1, 2009, Proceed-*
743 *ings*, volume 5595 of *Lecture Notes in Computer Science*, pages 295–304. Springer, 2009.
744 doi:10.1007/978-3-642-02348-4_21.
- 745 27 Zhaohui Luo. Notes on universes in type theory, October, 2012. URL: [https://www.cs.rhul.
746 ac.uk/home/zhaohui/universes.pdf](https://www.cs.rhul.ac.uk/home/zhaohui/universes.pdf).
- 747 28 Christine Paulin-Mohring. Inductive definitions in the system coq rules and properties. In
748 Marc Bezem and Jan Friso Groote, editors, *Typed Lambda Calculi and Applications*, pages
749 328–345, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.
- 750 29 Matthieu Sozeau and Nicolas Tabareau. Universe polymorphism in coq. In Gerwin Klein and
751 Ruben Gamboa, editors, *Interactive Theorem Proving*, pages 499–514, Cham, 2014. Springer
752 International Publishing.
- 753 30 Terese. *Term rewriting systems.*, volume 55 of *Cambridge tracts in theoretical computer science*.
754 Cambridge University Press, 2003.
- 755 31 François Thiré. Sharing a library between proof assistants: Reaching out to the HOL family. In
756 Frédéric Blanqui and Giselle Reis, editors, *Proceedings of the 13th International Workshop on*

23:20 Encoding impredicative hierarchy of type universes with variables

- 757 *Logical Frameworks and Meta-Languages: Theory and Practice, FMTP@FSCD 2018, Oxford,*
758 *UK, 7th July 2018*, volume 274 of *EPTCS*, pages 57–71, 2018. doi:10.4204/EPTCS.274.5.
- 759 32 François Thiré. *Interoperability between proof systems using the logical framework De-*
760 *dukti. (Interopérabilité entre systèmes de preuve en utilisant le cadre logique Dedukti).*
761 PhD thesis, École normale supérieure Paris-Saclay, Cachan, France, 2020. URL: <https://tel.archives-ouvertes.fr/tel-03224039>.
762
- 763 33 Vladimir Voevodsky. A universe polymorphic type system, October 22, 2014. An
764 unfinished unreleased manuscript. URL: [https://www.math.ias.edu/Voevodsky/files/](https://www.math.ias.edu/Voevodsky/files/files-annotated/Dropbox/Unfinished_papers/Type_systems/UPTS_current/Universe_polymorphic_type_sytem.pdf)
765 [files-annotated/Dropbox/Unfinished_papers/Type_systems/UPTS_current/Universe_](https://www.math.ias.edu/Voevodsky/files/files-annotated/Dropbox/Unfinished_papers/Type_systems/UPTS_current/Universe_polymorphic_type_sytem.pdf)
766 [polymorphic_type_sytem.pdf](https://www.math.ias.edu/Voevodsky/files/files-annotated/Dropbox/Unfinished_papers/Type_systems/UPTS_current/Universe_polymorphic_type_sytem.pdf).
- 767 34 Harald Zankl, Bertram Felgenhauer, and Aart Middeldorp. Csi: a confluence tool. volume
768 6803, pages 499–505, 07 2011. doi:10.1007/978-3-642-22438-6_38.