Normalisation for First-Class Universe Levels

Citation for the original published paper (version of record):
Danielsson, N., Favier, N., Kubánek, O. (2026). Normalisation for First-Class Universe Levels. Proceedings of the ACM on Programming Languages, 10: 64-88. http://dx.doi.org/10.1145/3776645

RESEARCH-ARTICLE

Normalisation for First-Class Universe Levels

NILS ANDERS DANIELSSON, Chalmers University of Technology, Gothenburg, Vastra Gotaland, Sweden

NAÏM CAMILLE FAVIER, Chalmers University of Technology, Gothenburg, Vastra Gotaland, Sweden

ONDŘEJ KUBÁNEK, Chalmers University of Technology, Gothenburg, Vastra Gotaland, Sweden

Published: 08 January 2026
Accepted: 06 November 2025
Received: 10 July 2025

Normalisation for First-Class Universe Levels

NILS ANDERS DANIELSSON, University of Gothenburg and Chalmers University of Technology, Sweden
NAÏM CAMILLE FAVIER, Chalmers University of Technology and University of Gothenburg, Sweden
ONDŘEJ KUBÁNEK, Chalmers University of Technology, Sweden

Various mechanisms are available for managing universe levels in proof assistants based on type theory. The Agda proof assistant implements a strong form of universe polymorphism in which universe levels are internalised as a type, making levels first-class objects and permitting higher-rank quantification via ordinary Π-types. We prove normalisation and decidability of equality and type-checking for a type theory with first-class universe levels inspired by Agda. We also show that level primitives can safely be erased in extracted programs. Our development is formalised in Agda itself and builds upon previous work which uses logical relations on extrinsically typed syntax.

CCS Concepts: • **Theory of computation** → **Type theory**; *Logic and verification.*

Additional Key Words and Phrases: dependent type theory, universe polymorphism, first-class universe levels, metatheory, formalisation, Agda

ACM Reference Format:
Nils Anders Danielsson, Naïm Camille Favier, and Ondřej Kubánek. 2026. Normalisation for First-Class Universe Levels. *Proc. ACM Program. Lang.* 10, POPL, Article 3 (January 2026), 25 pages. https://doi.org/10.1145/3776645

1 Introduction

Universes, or types of types, are a key feature of dependent type theory. From a programming perspective, quantifying over a universe enables a form of *polymorphism*: for instance, one can define a polymorphic identity function with the type $(A : \mathcal{U}) \to A \to A$, or a uniform list type former with the type $\mathcal{U} \to \mathcal{U}$, thus avoiding the need to duplicate these constructions for every type. In addition, they make it possible to define families of types by *case analysis* on terms, as in the following definition of the family of finite n-element types by recursion on the type of natural numbers:

$$\begin{aligned}&\mathsf{Fin} : \mathbb{N} \to \mathcal{U}\\ &\mathsf{Fin}\ 0 \quad\ = \bot\\ &\mathsf{Fin}\ (1 + n) = \top + \mathsf{Fin}\ n\end{aligned}$$

This is a form of *large elimination*, which makes it possible to express non-uniform dependency of types on terms, sometimes called *full-spectrum* dependent types [McKinna 2006; Angiuli and Gratzer 2025]. Large elimination also increases the strength of dependent type theory: using the family Fin defined above, one can prove that $0 \neq 1$; by contrast, it is consistent with a plain version of Martin-Löf type theory without universes that all terms are equal [Smith 1988]. Furthermore,

Authors' Contact Information: Nils Anders Danielsson, Department of Computer Science and Engineering, University of Gothenburg and Chalmers University of Technology, Gothenburg, Sweden, nad@cse.gu.se; Naïm Camille Favier, Department of Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden, naimf@chalmers.se; Ondřej Kubánek, Chalmers University of Technology, Gothenburg, Sweden, kubanek0ondrej@gmail.com.

ACM 2475-1421/2026/1-ART3
https://doi.org/10.1145/3776645

from the perspective of the formalisation of mathematics, universes make it possible to talk about the *type* of e.g. groups, allowing mathematical structures to be manipulated as first-class objects.

This work is concerned with the metatheoretical properties of universes with *first-class universe levels*, a particular incarnation of the notion of universe in dependent type theory implemented in the Agda proof assistant [The Agda Team 2025].

Universe hierarchies. Since a universe is a type of types, one might expect a universe to be an element of itself. However, Girard [1972] showed that Russell's paradox in naïve set theory carries over to type theory: the assumption of a type $\mathcal{U}$ of *all* types, such that $\mathcal{U} : \mathcal{U}$, is inconsistent [Coquand 1986]. This has led to stratified theories with a hierarchy of universes $\mathcal{U}_\ell : \mathcal{U}_{\ell+1}$, where the universe level ℓ usually ranges over ordinal numbers less than a given ordinal. See for instance Coquand [2019] for a recent presentation of proofs of canonicity and normalisation for dependent type theory with a hierarchy of universes indexed by the natural numbers (the ordinal ω).

Universe polymorphism. While universes enable definitions quantifying over types, universe hierarchies introduce a further need for *universe-polymorphic* definitions in order to avoid duplication across universe levels. For example, a universe-polymorphic identity function could be given the type $\forall \ell.(A : \mathcal{U}_\ell) \to A \to A$, which quantifies over a universe level ℓ *and* a type in the corresponding universe; this can then be instantiated as the identity function for any type $A : \mathcal{U}_0$, as well as the identity function for $\mathcal{U}_0$ itself, $\mathcal{U}_0 \to \mathcal{U}_0$, etc.

By allowing level quantification only in top-level definitions, one obtains prenex, or rank-1, universe polymorphism. An implicit form of rank-1 universe polymorphism known as *typical ambiguity* is described by Huet [1988] and Harper and Pollack [1991]; this approach involves inferring a consistent assignment of universe levels as part of an elaboration phase. A system with explicit universe polymorphism, based on work by Sozeau and Tabareau [2014], is implemented in the proof assistant Rocq[1] [The Rocq Development Team 2025].

First-class levels. In Agda, finite universe levels are represented by a *type* Level within the theory, so that level quantification can be achieved using ordinary dependent function types, as in the level-polymorphic identity function $(\ell : \mathsf{Level}) \to (A : \mathcal{U}_\ell) \to A \to A$.[2] This is a seemingly very powerful feature that allows not only *higher-rank* universe polymorphism (for example, proving a theorem under the assumption that every universe is univalent [The Univalent Foundations Program 2013; Bezem et al. 2023]), but also e.g. storing universe levels in data structures. Allais [2019] made use of the latter feature to implement level-polymorphic n-ary functions, but noted that its metatheoretical consequences were unknown. While at least one type theory with first-class levels is now known to be consistent and admit closed term computation (canonicity) [Chan and Weirich 2025], there has been little work about properties related to *open* computation, such as normalisation and decidability of equality, properties that are relevant for implementation in proof assistants.

Universe-polymorphic types such as that of the identity function above quantify over all finite universe levels, so they are "too big" to live in any finite universe themselves. In Agda 2.6.1 [The Agda Team 2020], these types are given the type $\mathcal{U}_\omega$, which is not an element of any universe, so there are only $\omega + 1$ universe levels. The next version, Agda 2.6.2 [The Agda Team 2021], has a second countable hierarchy $\mathcal{U}_\omega : \mathcal{U}_{\omega+1} : \cdots$, hence a hierarchy of $\omega \cdot 2$ universes, of which only the first ω are internal and thus subject to universe polymorphism.

[1] Formerly known as Coq.
[2] Agda's default syntax uses Set instead of $\mathcal{U}$.

1.1 Contributions

We describe a dependent type theory with a hierarchy of universes à la Russell indexed by a type of levels (Section 2). Our theory of levels is closely inspired by Agda 2.6.1's hierarchy of $\omega + 1$ universes and features a least level $\mathbf{0}$, a successor function $-^+$, and a supremum operator $\sqcup$ obeying the following equalities judgementally: left and right identity for $\mathbf{0}$, associativity, commutativity, distributivity with respect to $-^+$, idempotence, and subsumption;[3] to put it more concisely, levels form a bounded sup-semilattice with an inflationary endomorphism of sup-semilattices $-^+$.

We prove consistency, canonicity, weak head normalisation, and injectivity of type formers for this type theory using logical relations (Section 3). We further prove decidability of equality and (for a fragment of the language) typing, and deep normalisation (Section 4). We build upon the work of Abel et al. [2017]; all the main results are formalised in Agda, extending a formalisation based on that of Abel et al. [2023]. Like the previous work, we make use of induction-recursion in the metatheory, but we do not make essential use of Agda's first-class levels:[4] most of our development is parametrised by a single universe level for the underlying type of the modality, so our proof could be replayed in a universe-monomorphic setting with just a handful of universes. To our knowledge, this is the first proof of normalisation for a type theory with higher-rank universe polymorphism or first-class levels.

We also prove that extraction to an untyped λ-calculus is sound when level primitives are erased (Section 5), building upon the results of Abel et al. [2023].

1.2 Related Work

Chan and Weirich [2025] prove consistency, canonicity and other properties for TTBFL, a type theory with *bounded* first-class universe levels: unlike the type theory presented here, this allows quantifying over levels bounded by a given level. However, as far as we know it is still unclear whether TTBFL enjoys normalisation, and the technique described in the present work does not straightforwardly extend to bounded levels (see Section 6). Their work is formalised in the proof assistant Lean [The Lean Developers 2025].

Bezem et al. [2023] present a type theory with higher-rank universe polymorphism realised as a separate form of quantification rather than with first-class types. Their theory of levels is quite similar to ours: it also consists of a sup-semilattice with an inflationary endomorphism (but no least level, so that every construction involving universes must be universe-polymorphic). Bezem et al. conjecture that normalisation holds. They also conjecture that, if a term has type $\mathbb{N}$ in a context with only level variables, then the term is convertible to a numeral. We prove a similar result for contexts where all assumptions have type Level or type $\mathcal{U}_t$ for some t (Corollary 3.13). Bezem et al. [2023] also present a variant of the type theory with level constraint assumptions.

Kovács [2022] investigates the semantics of type theories with generalised universe hierarchies (defined as types equipped with well-founded, transitive relations) and first-class universe levels within the framework of categories with families [Dybjer 1996] and builds inductive-recursive models of these theories, partially formalised in Agda. While canonicity plausibly follows from a gluing argument reusing the model used to prove consistency, normalisation is less clear. Kovács also discusses the universe features implemented in several proof assistants.

Hou (Favonia) et al. [2023] discuss further examples of universe hierarchies (notably, *non-well-founded* ones). They show, using a syntactic interpretation argument, that if certain type theories

[3] That is $\ell \sqcup \ell^+ = \ell^+$, or in other words $\ell \leq \ell^+$ for the partial order induced by $\sqcup$.
[4] We use Agda's standard library [The Agda Community 2025], which does use first-class levels [Allais 2019], but we expect that it would be possible to switch to a library without essential use of first-class levels.

with rank-1 universe polymorphism are inconsistent, then the same applies to a certain universe-monomorphic type theory with a hierarchy of ω universes.

Courant [2002] proves normalisation and other metatheoretical properties for a type theory in which contexts can introduce universe levels with bounds by translating it into a type theory without universe polymorphism.

Jang et al. [2025] formalise a version of Martin-Löf type theory with a cumulative hierarchy of ω universes without universe polymorphism in Rocq. They prove soundness and completeness for a normalisation-by-evaluation algorithm, and extract it into an executable OCaml type-checker.

Our proof of soundness of extraction (Theorem 5.1) builds on the work of Abel et al. [2023]. In that work the target of extraction is a language that uses call-by-name. We additionally support call-by-value. In the call-by-name setting we remove erased function arguments entirely (unlike Abel et al.). In the call-by-value setting it is not always safe to do so: Letouzey [2003] points out that if additionally the eliminator for the empty type is replaced with code that throws an exception, then that exception can be triggered, and Mishra-Linger [2008, Section 5.2.3] presents an example that involves using an inconsistent, erasable assumption to type code that becomes non-terminating after extraction.

There is a large body of work on erasure for dependently typed languages, but less about erasure in the presence of universe polymorphism. We are not aware of any prior work on correctness of erasure for dependent type theory with first-class universe levels. Sozeau et al. [2025] present a mechanised proof of correctness of erasure for a variant of Rocq's core language that includes support for universe polymorphism. Their extraction method is type-based: types and certain kinds of proofs are erased. In contrast, our extraction function is defined for a *graded* type theory, which allows programmers to mark certain things as erasable. The extraction function erases type constructors and level primitives, but a function argument of type $\mathcal{U}_t$ or Level might not be erased if it is not marked as erasable. Another difference is that Sozeau et al. prove statements of the form "if a term has a value, then…" (roughly), whereas we prove that "every well-typed and well-resourced term has a value and…" (given certain assumptions).

2 A Type Theory with First-Class Universe Levels

The formalisation of Abel et al. [2023] builds on work by Abel et al. [2017], later extended by Gaëtan Gilbert, Wojciech Nawrocki and Oskar Eriksson. It features Π-types, strong and weak Σ-types, an empty type $\bot$, a strong unit type, natural numbers, and one universe closed under those type formers, with η-rules for Π-types, strong Σ-types, and the unit type. Later the first-named author of this paper extended the formalisation with identity types, and Oskar Eriksson added a weak unit type. We extend the formalisation further with a type of levels, a hierarchy of universes indexed by levels, and Lift types with η-rules. Our formalisation is available online [Danielsson et al. 2025].

In order to keep our presentation focused, we mainly describe a fragment of our type theory with only levels, universes, Lift types and Π-types, and refer the reader to Abel et al. [2017, 2023] and to the formalisation for a complete treatment of the other type formers. Our presentation also differs somewhat from the formalisation. For instance, we do not include grade annotations for Π-types [Abel et al. 2023] until Section 5, and although we present well-scoped syntax, we do not typically state conditions like "the context and the type are indexed by the same natural number" (see the source code for the full details of such conditions). As another example, the formalisation provides optional support for equality reflection, and some results are proved under the assumption that equality reflection is not allowed: in this text we instead work with a type theory without support for equality reflection.

$$\frac{i \in \{0, \ldots, n-1\}}{x_i \in \mathsf{Term}\, n} \qquad \frac{}{\mathsf{Level} \in \mathsf{Term}\, n} \qquad \frac{}{\mathbf{0} \in \mathsf{Term}\, n} \qquad \frac{t \in \mathsf{Term}\, n}{t^+ \in \mathsf{Term}\, n} \qquad \frac{t, u \in \mathsf{Term}\, n}{t \sqcup u \in \mathsf{Term}\, n}$$

$$\frac{t \in \mathsf{Term}\, n}{\mathcal{U}_t \in \mathsf{Term}\, n} \qquad \frac{t \in \mathsf{Term}\, n \qquad A \in \mathsf{Term}\, n}{\mathsf{Lift}_t\, A \in \mathsf{Term}\, n} \qquad \frac{u \in \mathsf{Term}\, n}{\mathsf{lift}\, u \in \mathsf{Term}\, n} \qquad \frac{t \in \mathsf{Term}\, n}{\mathsf{lower}\, t \in \mathsf{Term}\, n}$$

$$\frac{A \in \mathsf{Term}\, n \qquad B \in \mathsf{Term}\, (1+n)}{\Pi\, A\, B \in \mathsf{Term}\, n} \qquad \frac{t, u \in \mathsf{Term}\, n}{t\, u \in \mathsf{Term}\, n} \qquad \frac{t \in \mathsf{Term}\, (1+n)}{\lambda\, t \in \mathsf{Term}\, n} \qquad \ldots$$

Fig. 1. Selected raw expressions.

Throughout the paper, some definitions and theorems are hyperlinked to the formalisation. The file `README.lagda.md` in the formalisation contains counterparts to all the links, so if the links no longer work, then one can use that file instead.

2.1 Syntax

We start by describing (in Figure 1) the syntax of well-scoped expressions $\boxed{\mathsf{Term}\, n}$: terms and types in a context of length $n \in \mathbb{N}$. We use $\in$ to denote type membership in the metatheory.

Contexts of length $n \in \mathbb{N}$, written $\boxed{\mathsf{Con}\, n}$, are given by the following rules:

$$\frac{}{\cdot \in \mathsf{Con}\, 0} \qquad \frac{\Gamma \in \mathsf{Con}\, n \qquad A \in \mathsf{Term}\, n}{(\Gamma, A) \in \mathsf{Con}\, (1+n)}$$

Variables in a context $\Gamma \in \mathsf{Con}\, n$ are represented using de Bruijn [1972] indices ranging over a finite set with n elements. Expressions admit substitution and weakening operations, both written $t[\sigma]$. We omit the definition of substitutions and weakenings, and only mention the weakening $\mathbf{p}$ that takes terms in a context to a context with a new variable at the end.

If $k \in \mathbb{N}$, we may write $k + t$ for the kth iterated application of $-^+$ to t, and $\downarrow k$ for $k + \mathbf{0}$: for example, $\downarrow 1$ denotes $\mathbf{0}^+$. We may also use the syntax $(a : A) \to B\, a$ and $\lambda\, a.\ f\, a$ in examples, instead of de Bruijn indices. We reserve the names ℓ, ℓ' for external levels and use t, u, v for level expressions.

2.2 Typing Rules

There are five typing judgements, defined simultaneously: well-formed contexts $\boxed{\vdash \Gamma}$, well-formed types $\boxed{\Gamma \vdash A}$, equal types $\boxed{\Gamma \vdash A = B}$, well-typed terms $\boxed{\Gamma \vdash t : A}$, and equal terms $\boxed{\Gamma \vdash t = u : A}$. These definitions also make use of a judgement for variables $\boxed{\Gamma \ni x_i : A}$.

Figure 2 describes some key typing rules for our small fragment. We omit some bureaucratic details: the rules for reflexivity, symmetry, transitivity and syntactic congruence of judgemental equality, and certain presuppositions used in the formalisation (rules with omitted presuppositions are admissible). We also omit the rules for terms of Π-types. We write $\Gamma \vdash t, u : A$ to abbreviate the conjunction of judgements $\Gamma \vdash t : A$ and $\Gamma \vdash u : A$.

The equality rules for levels express the fact that well-typed level expressions in context Γ form a sup-semilattice with respect to $\sqcup$ with a least element $\mathbf{0}$ and an inflationary endomorphism of sup-semilattices $-^+$, via the equivalent algebraic characterisation of sup-semilattices as idempotent, commutative monoids. We recover the ordering on levels by defining the judgement $\boxed{\Gamma \vdash t \leq u : \mathsf{Level}}$ as $\Gamma \vdash t \sqcup u = u : \mathsf{Level}$. That $-^+$ is an endomorphism of sup-semilattices means

$$\frac{}{\vdash \cdot} \qquad \frac{\Gamma \vdash A}{\vdash \Gamma, A} \qquad \frac{}{\Gamma, A \ni x_0 : A[\mathbf{p}]} \qquad \frac{\Gamma \ni x_i : A}{\Gamma, B \ni x_{i+1} : A[\mathbf{p}]} \qquad \textsc{Var}\ \frac{\vdash \Gamma \qquad \Gamma \ni x_i : A}{\Gamma \vdash x_i : A}$$

$$\textsc{Univ}\ \frac{\Gamma \vdash A : \mathcal{U}_t}{\Gamma \vdash A} \qquad \textsc{Univ-Eq}\ \frac{\Gamma \vdash A = B : \mathcal{U}_t}{\Gamma \vdash A = B} \qquad \textsc{Conv}\ \frac{\Gamma \vdash t : A \qquad \Gamma \vdash A = B}{\Gamma \vdash t : B} \qquad \textsc{Conv-Eq}\ \frac{\Gamma \vdash t = u : A \qquad \Gamma \vdash A = B}{\Gamma \vdash t = u : B}$$

$$\textsc{Level}\ \frac{\vdash \Gamma}{\Gamma \vdash \mathsf{Level} : \mathcal{U}_0} \qquad \textsc{Lzero}\ \frac{\vdash \Gamma}{\Gamma \vdash \mathbf{0} : \mathsf{Level}} \qquad \textsc{Lsuc}\ \frac{\Gamma \vdash t : \mathsf{Level}}{\Gamma \vdash t^+ : \mathsf{Level}} \qquad \textsc{Lsup}\ \frac{\Gamma \vdash t, u : \mathsf{Level}}{\Gamma \vdash t \sqcup u : \mathsf{Level}} \qquad \textsc{U}\ \frac{\Gamma \vdash t : \mathsf{Level}}{\Gamma \vdash \mathcal{U}_t : \mathcal{U}_{t^+}}$$

$$\textsc{Id-L}\ \frac{\Gamma \vdash t : \mathsf{Level}}{\Gamma \vdash \mathbf{0} \sqcup t = t : \mathsf{Level}} \qquad \textsc{Assoc}\ \frac{\Gamma \vdash t, u, v : \mathsf{Level}}{\Gamma \vdash (t \sqcup u) \sqcup v = t \sqcup (u \sqcup v) : \mathsf{Level}} \qquad \textsc{Comm}\ \frac{\Gamma \vdash t, u : \mathsf{Level}}{\Gamma \vdash t \sqcup u = u \sqcup t : \mathsf{Level}}$$

$$\textsc{Idem}\ \frac{\Gamma \vdash t : \mathsf{Level}}{\Gamma \vdash t \sqcup t = t : \mathsf{Level}}$$

$$\textsc{Distrib}\ \frac{\Gamma \vdash t, u : \mathsf{Level}}{\Gamma \vdash t^+ \sqcup u^+ = (t \sqcup u)^+ : \mathsf{Level}} \qquad \textsc{Sub}\ \frac{\Gamma \vdash t : \mathsf{Level}}{\Gamma \vdash t \sqcup t^+ = t^+ : \mathsf{Level}}$$

$$\textsc{Lift}\ \frac{\Gamma \vdash u : \mathsf{Level} \qquad \Gamma \vdash A}{\Gamma \vdash \mathsf{Lift}_u\, A} \qquad \textsc{Lift-U}\ \frac{\Gamma \vdash t, u : \mathsf{Level} \qquad \Gamma \vdash A : \mathcal{U}_t}{\Gamma \vdash \mathsf{Lift}_u\, A : \mathcal{U}_{t \sqcup u}}$$

$$\textsc{lift}\ \frac{\Gamma \vdash t : \mathsf{Level} \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{lift}\ u : \mathsf{Lift}_t\, A} \qquad \textsc{lower}\ \frac{\Gamma \vdash u : \mathsf{Lift}_t\, A}{\Gamma \vdash \mathsf{lower}\ u : A}$$

$$\textsc{Lift-}\beta\ \frac{\Gamma \vdash u : A}{\Gamma \vdash \mathsf{lower}\ (\mathsf{lift}\ u) = u : A} \qquad \textsc{Lift-}\eta\ \frac{\Gamma \vdash t, u : \mathsf{Lift}_v\, A \qquad \Gamma \vdash \mathsf{lower}\ t = \mathsf{lower}\ u : A}{\Gamma \vdash t = u : \mathsf{Lift}_v\, A}$$

$$\textsc{Pi}\ \frac{\Gamma, A \vdash B}{\Gamma \vdash \Pi\, A\, B} \qquad \textsc{Pi-U}\ \frac{\Gamma \vdash t : \mathsf{Level} \qquad \Gamma \vdash A : \mathcal{U}_t \qquad \Gamma, A \vdash B : \mathcal{U}_{t[\mathbf{p}]}}{\Gamma \vdash \Pi\, A\, B : \mathcal{U}_t} \qquad \ldots$$

Fig. 2. Selected typing rules.

that it distributes over $\sqcup$, which is expressed by the rule Distrib; that it is inflationary means that we have $\Gamma \vdash t \leq t^+ : \mathsf{Level}$, which is exactly the content of the subsumption rule Sub. The idempotence of $\sqcup$ ensures that the order is reflexive; a standard argument shows that it is also transitive and antisymmetric. The right identity rule is derivable from the left identity rule and commutativity, so we do not include it.

The following alternative typing rule for Lift is admissible, as one would expect:

$$\frac{\textsc{Lift-}\leq \qquad \Gamma \vdash t \leq u : \mathsf{Level} \qquad \Gamma \vdash A : \mathcal{U}_t}{\Gamma \vdash \mathsf{Lift}_u\, A : \mathcal{U}_u}$$

Notice a crucial difference between the two rules Pi and Pi-U: in the Pi-U rule, the level of the type family $\Gamma, A \vdash B$ must be a term t in context Γ, so that it cannot depend on the index of type A. This excludes all non-trivial examples of universe-polymorphic types, like the type of the universe-polymorphic identity function $(t : \mathsf{Level}) \to (A : \mathcal{U}_t) \to A \to A$: this type is only well-typed by the Pi rule, and does not live in any universe.

Comparison with Agda. We compare our system with Agda's type theory:

- Agda's Π- and Σ-type formers are *heterogeneous*: given a type A and a type family B at universe levels t and u respectively, they form a type at level $t \sqcup u$. By contrast, our Π- and Σ-type formers are homogeneous: when they are used to create elements of a universe the two levels must be the same. For instance, the type $\Pi\, \mathcal{U}_0\, \mathcal{U}_{0^+}$, which is a well-formed type in the empty context, does not have a type. Heterogeneous type formers can be defined using the homogeneous ones and Lift, but Π-types defined this way require level annotations, unlike Agda's Π-types which also work with types not belonging to any universe.
- In Agda 2.6.1 [The Agda Team 2020], types like $(t : \mathsf{Level}) \to \mathsf{Set}_t$ are assigned the type Set_ω, which is not an element of any universe (ignoring any bugs in the Agda implementation). In contrast, our type theory does *not* have a universe $\mathcal{U}_\omega$; types that would live in this universe are instead simply well-formed types ($\Gamma \vdash A$).
- In Agda 2.6.2 [The Agda Team 2021], the universe Set_ω is extended to a second countable hierarchy $\mathsf{Set}_\omega : \mathsf{Set}_{\omega+1} : \cdots$. This has the benefit that every type is contained in a universe; furthermore, since one cannot quantify over levels above ω, the second hierarchy is not internalised, avoiding the need for an additional universe $\mathsf{Set}_{\omega\cdot 2}$. We expect that our formalisation could be extended similarly (see Section 6), but with homogeneous Π- and Σ-types it would presumably still not be the case that every type is contained in a universe.
- Like Agda, our type theory is not cumulative:[5] for example, the base type of natural numbers does not have type $\mathcal{U}_{t^+}$ for any t. Instead, types can be explicitly lifted to higher universes using Lift.
- Agda 2.6.4 [The Agda Team 2023] introduces a `--level-universe` flag which makes Level live in a separate universe instead of Set_0. Similarly, our formalisation has a parameter to replace the conclusion of the Level rule with $\Gamma \vdash \mathsf{Level}$. A notable difference is that, in this restricted setting, Agda disallows forming the identity type of Level, whereas our type theory allows it (but that identity type does not live in a universe). With the exception of that observation the parameter does not affect the results in this paper.

[5] Agda has optional, experimental support for cumulativity using the `--cumulativity` flag.

$$\textsc{Lsup-L}\ \frac{\Gamma \vdash t \longrightarrow t' : \mathsf{Level} \qquad \Gamma \vdash u : \mathsf{Level}}{\Gamma \vdash t \sqcup u \longrightarrow t' \sqcup u : \mathsf{Level}} \qquad \textsc{Lsup-R}\ \frac{\Gamma \vdash t : \mathsf{Level} \qquad \Gamma \vdash u \longrightarrow u' : \mathsf{Level}}{\Gamma \vdash t^+ \sqcup u \longrightarrow t^+ \sqcup u' : \mathsf{Level}}$$

$$\textsc{Id-L}\ \frac{\Gamma \vdash t : \mathsf{Level}}{\Gamma \vdash \mathbf{0} \sqcup t \longrightarrow t : \mathsf{Level}} \qquad \textsc{Id-R}\ \frac{\Gamma \vdash t : \mathsf{Level}}{\Gamma \vdash t^+ \sqcup \mathbf{0} \longrightarrow t^+ : \mathsf{Level}} \qquad \textsc{Distrib}\ \frac{\Gamma \vdash t, u : \mathsf{Level}}{\Gamma \vdash t^+ \sqcup u^+ \longrightarrow (t \sqcup u)^+ : \mathsf{Level}}$$

$$\textsc{Lift-}\beta\ \frac{\Gamma \vdash u : A}{\Gamma \vdash \mathsf{lower}\ (\mathsf{lift}\ u) \longrightarrow u : A} \qquad \textsc{Lower}\ \frac{\Gamma \vdash t \longrightarrow u : \mathsf{Lift}_v\ A}{\Gamma \vdash \mathsf{lower}\ t \longrightarrow \mathsf{lower}\ u : A} \qquad \ldots$$

Fig. 3. Selected reduction rules.

2.3 Reduction Rules

We also extend the weak head reduction relation $\boxed{\Gamma \vdash t \longrightarrow u : A}$ of Abel et al. [2017, 2023] with the rules in Figure 3,[6] which ensure that every level expression in the empty context reduces to some canonical level $\downarrow k$ (this will follow from our normalisation argument). The rules are designed so that weak head reduction remains deterministic and stable under substitution.

One may see the rules for $\sqcup$ as a specification of the maximum function on natural numbers by recursion on the left argument, then on the right argument. Note that the Distrib rule goes "backwards" from what one might expect from a normalisation procedure for levels, as the goal is only to reduce closed levels to canonical form. A separate normalisation procedure going the other way, used to decide judgemental equality of levels, will be described in Section 4. Similarly we do not include reduction rules corresponding to e.g. associativity or commutativity, those properties are handled by the decision procedure.

In light of these reduction rules we carve out three syntactic categories of expressions: atomic neutrals ($\mathsf{Neutral}^{\mathsf{a}}\ n$), neutrals ($\mathsf{Neutral}\ n$), and weak head normal forms ($\mathsf{Whnf}\ n$), defined in Figure 4 for our small fragment. Normals and neutrals are standard ingredients of normalisation arguments based on Tait's computability method [Tait 1967; Altenkirch et al. 1995]. We use the term weak head normal form for "constructor" applications and neutral terms, where neutral terms are variables with a stack of eliminators (here we consider $\sqcup$ as an eliminator). Atomic neutrals are neutral expressions that are not applications of $\sqcup$; this distinction only matters at type Level. The reason for considering stuck applications of $\sqcup$ separately will become apparent in Section 3.

A similar syntactic category of stuck *lists* appears in the work of Allais et al. [2013, Figure 3], which extends type theory with definitional equalities for lists, including e.g. associativity of the list concatenation operator ++. Unlike us they only have a rule for applications of ++ to a stuck list on the left, not on the right, in accordance with the fact that the definition of ++ uses recursion on the left argument, whereas the reduction rules for $\sqcup$ in Section 2.3 "recurse" on both arguments.

We use the notation $\boxed{\Gamma \vdash - \longrightarrow^* - : A}$ for the reflexive, transitive closure of $\Gamma \vdash - \longrightarrow - : A$, and we define type reduction $\boxed{\Gamma \vdash A \longrightarrow B}$ by "there is some t such that $\Gamma \vdash A \longrightarrow B : \mathcal{U}_t$". Let $\Gamma \vdash \mathcal{J}$ stand for any of the following typing judgements: $\Gamma \vdash A$, $\Gamma \vdash A = B$, $\Gamma \vdash t : A$ or $\Gamma \vdash t = u : A$. We have the following properties (Abel et al. [2017, Theorem 3.26] prove some of these properties after the fundamental lemma, but we structure our proofs a little differently):

[6] This time we omit one presupposition.

$$\frac{t \in \mathsf{Neutral}^{\mathsf{a}}\, n}{t \in \mathsf{Neutral}\, n} \qquad \frac{t \in \mathsf{Neutral}\, n}{t \in \mathsf{Whnf}\, n}$$

$$\frac{i \in \{0, \ldots, n-1\}}{x_i \in \mathsf{Neutral}^{\mathsf{a}}\, n} \qquad \frac{t \in \mathsf{Neutral}^{\mathsf{a}}\, n}{\mathsf{lower}\, t \in \mathsf{Neutral}^{\mathsf{a}}\, n} \qquad \frac{t \in \mathsf{Neutral}^{\mathsf{a}}\, n \qquad u \in \mathsf{Term}\, n}{t\, u \in \mathsf{Neutral}^{\mathsf{a}}\, n}$$

$$\frac{t \in \mathsf{Neutral}\, n \qquad u \in \mathsf{Term}\, n}{t \sqcup u \in \mathsf{Neutral}\, n} \qquad \frac{t \in \mathsf{Term}\, n \qquad u \in \mathsf{Neutral}\, n}{t^+ \sqcup u \in \mathsf{Neutral}\, n}$$

$$\frac{}{\mathsf{Level} \in \mathsf{Whnf}\, n} \qquad \frac{}{\mathbf{0} \in \mathsf{Whnf}\, n} \qquad \frac{t \in \mathsf{Term}\, n}{t^+ \in \mathsf{Whnf}\, n} \qquad \frac{t \in \mathsf{Term}\, n}{\mathcal{U}_t \in \mathsf{Whnf}\, n}$$

$$\frac{t, A \in \mathsf{Term}\, n}{\mathsf{Lift}_t\, A \in \mathsf{Whnf}\, n} \qquad \frac{u \in \mathsf{Term}\, n}{\mathsf{lift}\, u \in \mathsf{Whnf}\, n} \qquad \frac{A \in \mathsf{Term}\, n \qquad B \in \mathsf{Term}\, (1+n)}{\Pi\, A\, B \in \mathsf{Whnf}\, n}$$

$$\frac{t \in \mathsf{Term}\, (1+n)}{\lambda\, t \in \mathsf{Whnf}\, n} \qquad \ldots$$

Fig. 4. Atomic neutrals, neutrals, and weak head normal forms.

Lemma 2.1. *(Well-formedness) Inhabited typing and reduction judgements have well-formed arguments.*

- $\Gamma \vdash \mathcal{J}$ *implies* $\vdash \Gamma$,
- $\Gamma \vdash A = B$ *implies* $\Gamma \vdash A$ *and* $\Gamma \vdash B$,
- $\Gamma \vdash t : A$ *implies* $\Gamma \vdash A$,
- $\Gamma \vdash t = u : A$ *implies* $\Gamma \vdash t : A$ *and* $\Gamma \vdash u : A$,
- $\Gamma \vdash A \longrightarrow^* B$ *implies* $\Gamma \vdash A$ *and* $\Gamma \vdash B$, *and*
- $\Gamma \vdash t \longrightarrow^* u : A$ *implies* $\Gamma \vdash t : A$ *and* $\Gamma \vdash u : A$.

3 A Logical Relation

In order to prove properties of terms in open contexts, we use a Kripke logical relation extending the one formalised by Abel et al. [2017, 2023]. This consists of judgements $\Gamma \Vdash \mathcal{J}$, called *reducibility predicates*, capturing the intended observations for the elements of each type: for example, the predicate for natural numbers says that every reducible element of type $\mathbb{N}$ reduces to 0, the successor of a reducible natural number, or a neutral term. Our aim will then be to prove a *fundamental lemma* (Lemma 3.3) stating that the reducibility predicates form a model of our type theory, in the sense that well-typed objects are reducible (e.g. $\Gamma \vdash t : A$ implies $\Gamma \Vdash t : A$).

More precisely, we model the internal universe hierarchy using an external hierarchy of $\omega + 1$ inductive-recursive universes: we define (in Section 3.2) judgements $\Gamma \Vdash_\ell \mathcal{J}$, where ℓ is either a natural number or ω and (as above) $\mathcal{J}$ stands for any of the four typing judgements (excluding context well-formedness). We do this by well-founded induction on ℓ using an inductive-recursive "universe operator" which defines the judgements $\Gamma \Vdash_\ell \mathcal{J}$ assuming that the judgements $\Gamma \Vdash_{\ell'} \mathcal{J}'$ are defined for all $\ell' < \ell$ and for all $\mathcal{J}'$. By contrast, the logical relations of Abel et al. [2017] only use two external levels, 0 and 1, as they only have a single universe.

Following Abel et al. [2017], our logical relation is parametrised by a family of *generic equality relations*: $\boxed{\Gamma \vdash A \cong B}$ for types, $\boxed{\Gamma \vdash t \cong u : A}$ for terms, and $\boxed{\Gamma \vdash t \sim u : A}$ for atomic neutral terms. These generic equality relations are partial equivalence relations closed under weakening, type conversion and (for the first two) weak head expansion, and they are contained in judgemental equality (the third relation is also contained in the second). They also satisfy variants of various judgemental equality rules (e.g. associativity of $\sqcup$); we refer the reader to the formalisation and the previous work for more details. The fundamental lemma is instantiated twice, first with judgemental equality, then a second time with *algorithmic equality* relations (some further motivation for this design is given by Adjedj et al. [2024, Section 4.4]).

3.1 Reducible Levels and Neutrals

The proof of Abel et al. [2017] extends directly to a hierarchy of ω universes instead of a single one where, informally, an element $A : \mathcal{U}_\ell$ of a universe is reducible if (among other conditions) A is a reducible *type* at level ℓ, and $\mathcal{U}_\ell$ is only reducible at levels strictly greater than ℓ.

The introduction of internal levels complicates things: when should $\mathcal{U}_t$ be reducible? We can find a hint in the canonicity argument of Sterling [2019, Section 5] for an algebraic presentation of a type theory with a hierarchy of universes à la Russell and a sort of levels: in the case of canonicity for *closed* terms, a level $\cdot \vdash t : \mathsf{Level}$ is reducible if there is a natural number $t^\bullet$ (called its *realiser*) such that t is judgementally equal to $\downarrow t^\bullet$, and a type $A : \mathcal{U}_t$ is interpreted as a $t^\bullet$-small family over the closed terms of A (that is, a family valued in the $t^\bullet$-th metatheoretical universe).

In our setting with open terms a level might not be judgementally equal to $\downarrow n$ for some natural number n. However, we will still define a (recursive) function $\uparrow$ that assigns a realiser to each reducible level t, and use this to define when $\mathcal{U}_t$ is reducible. How should levels that reduce to neutral expressions be handled? In order to model the Conv rule, judgementally equal levels should have equal realisers; it is thus natural to require that $\uparrow$ is a homomorphism of sup-semilattices from reducible level terms up to judgemental equality to $(\mathbb{N}, \leq)$. This suggests that the reducibility predicate for a neutral level of the form $t \sqcup u$ should include information about t and u.

We start by defining reducible atomic neutral terms of any type $\boxed{\Gamma \Vdash \mathsf{neNf}\ t : A}$ and reducible levels $\boxed{\Gamma \Vdash \mathsf{Lvl}\ t}$; these definitions are independent of the external level ℓ. The judgement $\Gamma \Vdash \mathsf{neNf}\ t : A$ holds iff t is atomic neutral and $\Gamma \vdash t \sim t : A$. The judgement $\Gamma \Vdash \mathsf{Lvl}\ t$ holds iff $\Gamma \vdash t \longrightarrow^* \bar{t} : \mathsf{Level}$ with $\bar{t}$ in weak head normal form and furthermore $\Gamma \Vdash \mathsf{Lvl_w}\ \bar{t}$; the judgements $\boxed{\Gamma \Vdash \mathsf{Lvl_w}\ t}$ for levels in weak head normal form (WHNF) and $\boxed{\Gamma \Vdash \mathsf{Lvl_n}\ t}$ for neutral levels are defined mutually with $\Gamma \Vdash \mathsf{Lvl}\ t$ by the following inductive rules, which mirror the structure of normal and neutral levels given in Figure 4:

$$\textsc{Ne}\ \frac{\Gamma \Vdash \mathsf{Lvl_n}\ t}{\Gamma \Vdash \mathsf{Lvl_w}\ t} \qquad \textsc{Atom}\ \frac{\Gamma \Vdash \mathsf{neNf}\ t : \mathsf{Level}}{\Gamma \Vdash \mathsf{Lvl_n}\ t}$$

$$\textsc{Lzero}\ \frac{}{\Gamma \Vdash \mathsf{Lvl_w}\ \mathbf{0}} \qquad \textsc{Lsuc}\ \frac{\Gamma \Vdash \mathsf{Lvl}\ t}{\Gamma \Vdash \mathsf{Lvl_w}\ t^+} \qquad \textsc{Lsup-L}\ \frac{\Gamma \Vdash \mathsf{Lvl_n}\ t \qquad \Gamma \Vdash \mathsf{Lvl}\ u}{\Gamma \Vdash \mathsf{Lvl_n}\ t \sqcup u} \qquad \textsc{Lsup-R}\ \frac{\Gamma \Vdash \mathsf{Lvl}\ t \qquad \Gamma \Vdash \mathsf{Lvl_n}\ u}{\Gamma \Vdash \mathsf{Lvl_n}\ t^+ \sqcup u}$$

Given a reducible level t, we define the *realiser* $\boxed{\uparrow t}$ of t as a natural number (one of the first ω external levels) by the following specification, where $\sqcup$ also denotes the maximum of two natural numbers:

$$\frac{\Gamma \vdash t \longrightarrow^* \bar{t} : \mathsf{Level}}{\uparrow t = \uparrow \bar{t}} \qquad \uparrow \mathbf{0} = 0 \qquad \uparrow t^+ = 1 + \uparrow t \qquad \uparrow (t \sqcup u) = (\uparrow t) \sqcup (\uparrow u) \qquad \frac{t \in \mathsf{Neutral}^{\mathsf{a}}\ n}{\uparrow t = 0}$$

$$\textsc{Ne}\ \frac{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t = u}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{w}}\ t = u} \qquad \textsc{Atom}\ \frac{\Gamma \Vdash \mathsf{neNf}\ t = u : \mathsf{Level}}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t = u}$$

$$\textsc{Lzero}\ \frac{}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{w}}\ \mathbf{0} = \mathbf{0}} \qquad \textsc{Lsuc}\ \frac{\Gamma \Vdash \mathsf{Lvl}\ t = u}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{w}}\ t^+ = u^+} \qquad \textsc{Sub}\ \frac{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t \qquad \Gamma \Vdash \mathsf{Lvl}\ t \sqcup u = u}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{w}}\ t \sqcup u^+ = u^+}$$

$$\textsc{Sym}\ \frac{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{w}}\ t = u}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{w}}\ u = t} \qquad \textsc{Trans}\ \frac{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{w}}\ t = u \qquad \Gamma \Vdash \mathsf{Lvl}_{\mathsf{w}}\ u = v}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{w}}\ t = v}$$

$$\textsc{Lsup-L}\ \frac{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t = t' \qquad \Gamma \Vdash \mathsf{Lvl}\ u = u'}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t \sqcup u = t' \sqcup u'} \qquad \textsc{Lsup-R}\ \frac{\Gamma \Vdash \mathsf{Lvl}\ t = t' \qquad \Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ u = u'}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t^+ \sqcup u = t'^+ \sqcup u'} \qquad \textsc{Zero-R}\ \frac{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t \sqcup \mathbf{0} = t}$$

$$\textsc{Assoc-1}\ \frac{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t \qquad \Gamma \Vdash \mathsf{Lvl}\ u, v}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ (t \sqcup u) \sqcup v = t \sqcup (u \sqcup v)} \qquad \textsc{Assoc-2}\ \frac{\Gamma \Vdash \mathsf{Lvl}\ t \qquad \Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ u \qquad \Gamma \Vdash \mathsf{Lvl}\ v}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ (t^+ \sqcup u) \sqcup v = t^+ \sqcup (u \sqcup v)}$$

$$\textsc{Assoc-3}\ \frac{\Gamma \Vdash \mathsf{Lvl}\ t, u \qquad \Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ v}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ (t \sqcup u)^+ \sqcup v = t^+ \sqcup (u^+ \sqcup v)} \qquad \textsc{Comm-1}\ \frac{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t, u' \qquad \Gamma \Vdash \mathsf{Lvl}\ t = t' \qquad \Gamma \Vdash \mathsf{Lvl}\ u = u'}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t \sqcup u = u' \sqcup t'}$$

$$\textsc{Comm-2}\ \frac{\Gamma \Vdash \mathsf{Lvl}\ t \qquad \Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ u \qquad \Gamma \Vdash \mathsf{Lvl}\ t^+ = t'}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t^+ \sqcup u = u \sqcup t'} \qquad \textsc{Idem}\ \frac{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t \qquad \Gamma \Vdash \mathsf{Lvl}\ t = t'}{\Gamma \Vdash \mathsf{Lvl}_{\mathsf{n}}\ t \sqcup t' = t}$$

Fig. 5. Reducible level equality.

In reality, $\uparrow t$ is defined by recursion on a witness of $\Gamma \Vdash \mathsf{Lvl}\ t$, but we omit the witnesses for ease of notation; this is justified by a proof-irrelevance lemma. Note how we make use of the stratification of neutral level expressions into stuck applications of $\sqcup$ and atomic neutrals in the definition of level reducibility to ensure that the equation $\uparrow(t \sqcup u) = (\uparrow t) \sqcup (\uparrow u)$ holds: if $t \sqcup u$ were treated as an atomic neutral, then we would not be able to recursively compute $\uparrow t$ and $\uparrow u$.

The choice to define the realiser of any atomic neutral level as 0 may be surprising, but it makes no difference: any other natural number would do, and in our formalisation we make this definition *abstract* [The Agda Team 2025, Section 3.1] to ensure that it does not affect the rest of the development (alternatively, we could make this value a parameter to the proof, thus realising atomic neutrals as a *metatheoretical* neutral value). Informally, the important properties of $\uparrow$ have to do with the *relations* between levels: $\Gamma \Vdash \mathsf{Lvl}\ t = u$ implies $\uparrow t = \uparrow u$ and $\Gamma \Vdash \mathsf{Lvl}\ t \sqcup u = u$ implies $\uparrow t \leq \uparrow u$.[7]

[7] Observe that, for all $k < \omega$ and Γ, we have $\Gamma \Vdash \mathsf{Lvl} \downarrow k$ and $\uparrow \downarrow k = k$. Incidentally, the choice to realise atomic neutral levels as 0 means that we also have $\Gamma \vdash \downarrow \uparrow t \leq t : \mathsf{Level}$ for all reducible levels t, so that $\uparrow$ is a *coreflection* of the internal poset of levels in context Γ onto $(\mathbb{N}, \leq)$: it is part of a Galois connection $\downarrow \dashv \uparrow$ in which the left adjoint $\downarrow$ is an order embedding. This fact plays no role in our proof.

The reducible equality judgement for atomic neutrals $\boxed{\Gamma \Vdash \mathsf{neNf}\ t = u : A}$ holds iff t and u are both atomic neutral terms and $\Gamma \vdash t \sim u : A$. The reducible level equality judgement $\boxed{\Gamma \Vdash \mathsf{Lvl}\ t = u}$ holds iff $\Gamma \vdash t \longrightarrow^* \overline{t} : \mathsf{Level}$ and $\Gamma \vdash u \longrightarrow^* \overline{u} : \mathsf{Level}$, with $\overline{t}$ and $\overline{u}$ in WHNF and furthermore $\Gamma \Vdash \mathsf{Lvl}_\mathsf{w}\ \overline{t} = \overline{u}$; the judgements $\boxed{\Gamma \Vdash \mathsf{Lvl}_\mathsf{w}\ t = u}$ for levels in WHNF and $\boxed{\Gamma \Vdash \mathsf{Lvl}_\mathsf{n}\ t = u}$ for neutral levels are defined mutually with $\Gamma \Vdash \mathsf{Lvl}\ t = u$ by the inductive rules in Figure 5. Those rules are essentially refinements of the typing rules according to the syntactic categories we have mentioned, and are informed by the proofs that the typing rules preserve reducibility. For example, the rule for the associativity of $\sqcup$ is split into three rules corresponding to different ways in which the expressions can be neutral. The rule Sub stands out as the only case in which a neutral level is equated with a non-neutral level in WHNF. We modified it slightly compared to the typing rule Sub in order to prove that $\sqcup$ respects equality ($\Gamma \Vdash \mathsf{Lvl}\ t = u$) in its first argument. One may read it as saying that $t \leq u$ implies $t \leq u^+$ (for neutral t).

We spell out two illustrative lemmas, one for terms and one for equalities:

Lemma 3.1. *(Reducibility for Lsup) If* $\Gamma \Vdash \mathsf{Lvl}\ t$ *and* $\Gamma \Vdash \mathsf{Lvl}\ u$*, then* $\Gamma \Vdash \mathsf{Lvl}\ t \sqcup u$.

Proof. By recursion on $\Gamma \Vdash \mathsf{Lvl}\ t$:

- If $\Gamma \vdash t \longrightarrow^* \mathbf{0} : \mathsf{Level}$, then $\Gamma \vdash t \sqcup u \longrightarrow^* u : \mathsf{Level}$, hence $\Gamma \Vdash \mathsf{Lvl}\ t \sqcup u$ by weak head expansion.
- If $\Gamma \vdash t \longrightarrow^* t'^+ : \mathsf{Level}$ with $\Gamma \Vdash \mathsf{Lvl}\ t'$, then, by recursion on $\Gamma \Vdash \mathsf{Lvl}\ u$:
  - If $\Gamma \vdash u \longrightarrow^* \mathbf{0} : \mathsf{Level}$, then $\Gamma \vdash t \sqcup u \longrightarrow^* t'^+ : \mathsf{Level}$, hence $\Gamma \Vdash \mathsf{Lvl}\ t \sqcup u$ by Lsuc.
  - If $\Gamma \vdash u \longrightarrow^* u'^+ : \mathsf{Level}$ with $\Gamma \Vdash \mathsf{Lvl}\ u'$, then $\Gamma \vdash t \sqcup u \longrightarrow^* (t' \sqcup u')^+ : \mathsf{Level}$ and $\Gamma \Vdash \mathsf{Lvl}\ t' \sqcup u'$ by the induction hypothesis, hence $\Gamma \Vdash \mathsf{Lvl}\ t \sqcup u$ by Lsuc.
  - If $\Gamma \vdash u \longrightarrow^* m : \mathsf{Level}$ with $\Gamma \Vdash \mathsf{Lvl}_\mathsf{n}\ m$, then $\Gamma \vdash t \sqcup u \longrightarrow^* t'^+ \sqcup m : \mathsf{Level}$, hence $\Gamma \Vdash \mathsf{Lvl}\ t \sqcup u$ by Lsup-R.
- If $\Gamma \vdash t \longrightarrow^* n : \mathsf{Level}$ with $\Gamma \Vdash \mathsf{Lvl}_\mathsf{n}\ n$, then $\Gamma \vdash t \sqcup u \longrightarrow^* n \sqcup u : \mathsf{Level}$, hence $\Gamma \Vdash \mathsf{Lvl}\ t \sqcup u$ by Lsup-L.

Note that the structure of this proof is similar to a usual recursive definition of the maximum operator $\sqcup$ for natural numbers, with additional cases for neutral levels. □

Lemma 3.2. *(Reducibility for Idem) If* $\Gamma \Vdash \mathsf{Lvl}\ t$*, then* $\Gamma \Vdash \mathsf{Lvl}\ t \sqcup t = t$.

Proof. By recursion on $\Gamma \Vdash \mathsf{Lvl}\ t$:

- If $\Gamma \vdash t \longrightarrow^* \mathbf{0} : \mathsf{Level}$, then the result follows by the equality rule Lzero.
- If $\Gamma \vdash t \longrightarrow^* t'^+ : \mathsf{Level}$ with $\Gamma \Vdash \mathsf{Lvl}\ t'$, then it suffices to show $\Gamma \Vdash \mathsf{Lvl}_\mathsf{w}\ (t' \sqcup t')^+ = t'^+$, which follows by the induction hypothesis and the equality rule Lsuc.
- If $\Gamma \vdash t \longrightarrow^* n : \mathsf{Level}$ with $\Gamma \Vdash \mathsf{Lvl}_\mathsf{n}\ n$, then it suffices to show $\Gamma \Vdash \mathsf{Lvl}_\mathsf{n}\ n \sqcup t = n$. Since we have $\Gamma \Vdash \mathsf{Lvl}\ t = n$ by weak head expansion, this follows by the equality rule Idem. □

3.2 Reducibility

We are now ready to define the other reducibility judgements. As in the work of Abel et al. [2017, 2023], the main judgement $\boxed{\Gamma \Vdash_\ell A}$ is defined inductively, while the judgements $\boxed{\Gamma \Vdash_\ell A = B \mathbin{/} \mathscr{A}}$ and $\boxed{\Gamma \Vdash_\ell t = u : A \mathbin{/} \mathscr{A}}$ are defined by simultaneous *recursion* on a witness $\mathscr{A}$ of $\Gamma \Vdash_\ell A$. This *inductive-recursive* [Dybjer 2000] definition is our main source of metatheoretical strength compared to the object theory, which allows us to stay clear of Gödelian limitations. To reduce redundancy, we define the term reducibility predicate $\boxed{\Gamma \Vdash_\ell t : A \mathbin{/} \mathscr{A}}$ as $\Gamma \Vdash_\ell t = t : A \mathbin{/} \mathscr{A}$. Unlike the syntax, our model is cumulative by construction: $\Gamma \Vdash_\ell A$ implies $\Gamma \Vdash_{\ell'} A$ for $\ell \leq \ell'$, and similarly for the other relations.

Neutrals. Types which reduce to a reflexive atomic neutral type are reducible:

$$\frac{\Gamma \vdash A \longrightarrow^* N \qquad N \in \mathsf{Neutral}^{\mathsf{a}}\, n \qquad \Gamma \vdash N \cong N}{\Gamma \Vdash_\ell A}$$

For a derivation $\mathscr{N}$ built using this rule, we define:

- $\Gamma \Vdash_\ell A = B \;/\; \mathscr{N}$ iff there is an atomic neutral M such that $\Gamma \vdash B \longrightarrow^* M$ and $\Gamma \vdash N \cong M$.
- $\Gamma \Vdash_\ell t = u : A \;/\; \mathscr{N}$ iff there are terms k, m such that $\Gamma \vdash t \longrightarrow^* k : N$, $\Gamma \vdash u \longrightarrow^* m : N$, and $\Gamma \Vdash \mathsf{neNf}\; k = m : N$.

Levels. Types which reduce to Level are reducible at every level:

$$\frac{\Gamma \vdash A \longrightarrow^* \mathsf{Level}}{\Gamma \Vdash_\ell A}$$

For a derivation $\mathscr{L}$ built using this rule, we define:

- $\Gamma \Vdash_\ell A = B \;/\; \mathscr{L}$ iff $\Gamma \vdash B \longrightarrow^* \mathsf{Level}$.
- $\Gamma \Vdash_\ell t = u : A \;/\; \mathscr{L}$ iff $\Gamma \Vdash \mathsf{Lvl}\; t = u$.

Thus $\Gamma \Vdash_\ell t : A \;/\; \mathscr{L}$ is defined as $\Gamma \Vdash \mathsf{Lvl}\; t = t$, which is logically equivalent to $\Gamma \Vdash \mathsf{Lvl}\; t$ by level reflexivity and well-formedness.

Universes. Types which reduce to $\mathcal{U}_t$ for some reducible level t such that $\uparrow t < \ell$ are reducible at level ℓ:

$$\frac{\Gamma \vdash U \longrightarrow^* \mathcal{U}_t \qquad \Gamma \Vdash \mathsf{Lvl}\; t \qquad \uparrow t < \ell}{\Gamma \Vdash_\ell U}$$

As an example, $\mathcal{U}_0$ is reducible at level 1 in any well-formed context; more generally, $\mathcal{U}_t$ is reducible at level $1 + \uparrow t$ for any reducible level t. For a derivation $\mathscr{U}$ built using this rule, we define:

- $\Gamma \Vdash_\ell U = V \;/\; \mathscr{U}$ iff there exists u such that $\Gamma \vdash V \longrightarrow^* \mathcal{U}_u$ and $\Gamma \Vdash \mathsf{Lvl}\; t = u$.
- $\Gamma \Vdash_\ell A = B : U \;/\; \mathscr{U}$ iff

(1) $\Gamma \vdash A \longrightarrow^* \overline{A} : \mathcal{U}_t$ and $\Gamma \vdash B \longrightarrow^* \overline{B} : \mathcal{U}_t$, where $\overline{A}$ and $\overline{B}$ are types in WHNF (applications of type constructors or atomic neutral terms) such that $\Gamma \vdash \overline{A} \cong \overline{B} : \mathcal{U}_t$ and

(2) $\mathscr{A} \in \Gamma \Vdash_{\uparrow t} A$ and $\Gamma \Vdash_{\uparrow t} B$ and $\Gamma \Vdash_{\uparrow t} A = B \;/\; \mathscr{A}$.

Note that $\Gamma \Vdash_{\uparrow t} \mathcal{J}$ is well-defined by the induction hypothesis, since $\uparrow t < \ell$.

Lift types. Types which reduce to $\mathsf{Lift}_t\, F$ are reducible at level ℓ iff t and F are:

$$\frac{\Gamma \vdash A \longrightarrow^* \mathsf{Lift}_t\, F \qquad \Gamma \Vdash \mathsf{Lvl}\; t \qquad \mathscr{F} \in \Gamma \Vdash_\ell F}{\Gamma \Vdash_\ell A}$$

Note that this definition does not involve any actual lifting: from the point of view of reducibility, Lift is essentially transparent. Instead, lifting will follow from the cumulativity of the logical relation when proving the relevant case of the fundamental lemma (Lemma 3.6). We do not need to require that $\uparrow t \leq \ell$ either. For a derivation $\mathscr{L}$ built using this rule, we define:

- $\Gamma \Vdash_\ell A = B \;/\; \mathscr{L}$ iff $\Gamma \vdash B \longrightarrow^* \mathsf{Lift}_u\, G$, with $\Gamma \Vdash \mathsf{Lvl}\; t = u$ and $\Gamma \Vdash_\ell F = G \;/\; \mathscr{F}$.
- $\Gamma \Vdash_\ell t = u : A \;/\; \mathscr{L}$ iff $\Gamma \vdash t \longrightarrow^* \overline{t} : \mathsf{Lift}_t\, F$ and $\Gamma \vdash u \longrightarrow^* \overline{u} : \mathsf{Lift}_t\, F$, where $\overline{t}$ and $\overline{u}$ are in WHNF, and furthermore $\Gamma \Vdash_\ell \mathsf{lower}\, \overline{t} = \mathsf{lower}\, \overline{u} : F \;/\; \mathscr{F}$.

Π-types. We have not made any changes to the definition of reducibility for Π-types in order to support the universe hierarchy,[8] and omit the definitions for type equality and terms. The definition makes use of the notion of a well-formed weakening, $\Delta \vdash \rho : \Gamma$, and the operation $\Uparrow$ that lifts such a weakening to a weakening that satisfies $\Delta, A[\rho] \vdash \rho\Uparrow : \Gamma, A$ (assuming that $\Delta \vdash A[\rho]$):

$$\frac{\begin{array}{c}\Gamma \vdash A \longrightarrow^* \Pi\, B\, C \qquad \Gamma \vdash \Pi\, B\, C \cong \Pi\, B\, C \qquad \mathscr{B} \in \forall \Delta, \Delta \vdash \rho : \Gamma.\, \Delta \Vdash_\ell B[\rho] \\ \mathscr{C} \in \forall \Delta, \Delta \vdash \rho : \Gamma, \Delta \Vdash_\ell t : B[\rho] \;/\; \mathscr{B}\, \Delta\, \rho.\, \Delta \Vdash_\ell C[\rho\Uparrow][t/x_0] \\ \forall \Delta, \Delta \vdash \rho : \Gamma, t \in \Delta \Vdash_\ell t : B[\rho] \;/\; \mathscr{B}\, \Delta\, \rho, \Delta \Vdash_\ell u : B[\rho] \;/\; \mathscr{B}\, \Delta\, \rho, \Delta \Vdash_\ell t = u : B[\rho] \;/\; \mathscr{B}\, \Delta\, \rho. \\ \Delta \Vdash_\ell C[\rho\Uparrow][t/x_0] = C[\rho\Uparrow][u/x_0] : \mathscr{C}\, \Delta\, \rho\, t\end{array}}{\Gamma \Vdash_\ell A}$$

The reducibility judgements for type and term equality are defined by recursion on type reducibility witnesses, but for a given type it does not matter which witness one chooses (up to logical equivalence). Below we use the notation $\boxed{\Gamma \Vdash_\ell t : A}$ for the type of pairs containing a witness $\mathscr{A} \in \Gamma \Vdash_\ell A$ and an element of $\Gamma \Vdash_\ell t : A \;/\; \mathscr{A}$, and similarly $\boxed{\Gamma \Vdash_\ell t = u : A}$ stands for pairs with a witness $\mathscr{A} \in \Gamma \Vdash_\ell A$ and an element of $\Gamma \Vdash_\ell t = u : A \;/\; \mathscr{A}$. The notation $\boxed{\Gamma \Vdash_\ell A = B}$ is used for triples containing a witness $\mathscr{A} \in \Gamma \Vdash_\ell A$, an element of $\Gamma \Vdash_\ell B$, and an element of $\Gamma \Vdash_\ell A = B \;/\; \mathscr{A}$.

3.3 Validity and the Fundamental Lemma

The reducibility judgements of Section 3.2 are designed so that $\Gamma \Vdash_\ell \mathcal{J}$ implies $\Gamma \vdash \mathcal{J}$: one might find it helpful to think of them as describing a *displayed model* over the syntax of our type theory for which the fundamental lemma will provide a *section* [Kaposi et al. 2019]. However, the reducibility judgements are not a priori sufficient to model type theory; in particular, if we tried to prove the fundamental lemma for them by direct induction on typing derivations, then we would have trouble with the cases for Π-types. Instead, like Abel et al. [2017], we define a family of *validity judgements* $\Gamma \Vdash^{\mathsf{v}}_\ell \mathcal{J}$ which close the reducibility judgements under reducible substitutions.[9]

We first define reducibly equal substitutions $\boxed{\Delta \Vdash^{\mathsf{s}} \sigma_1 = \sigma_2 : \Gamma}$, valid contexts $\boxed{\Vdash^{\mathsf{v}} \Gamma}$ and valid equal types $\boxed{\Gamma \Vdash^{\mathsf{v}}_\ell A = B}$ by mutual recursion on the context Γ. The valid type judgement $\boxed{\Gamma \Vdash^{\mathsf{v}}_\ell A}$ is defined as $\Gamma \Vdash^{\mathsf{v}}_\ell A = A$.

- $\Delta \Vdash^{\mathsf{s}} \cdot = \cdot : \cdot$ iff Δ is a well-formed context ($\vdash \Delta$).
- $\Delta \Vdash^{\mathsf{s}} \sigma_1, t = \sigma_2, u : \Gamma, A$ iff $\Delta \Vdash^{\mathsf{s}} \sigma_1 = \sigma_2 : \Gamma$ and furthermore there exists a level ℓ such that $\Gamma \Vdash^{\mathsf{v}}_\ell A$ and $\Delta \Vdash_\ell t = u : A[\sigma_1]$.
- $\Vdash^{\mathsf{v}} \cdot$ holds trivially.
- $\Vdash^{\mathsf{v}} \Gamma, A$ iff there exists a level ℓ such that $\Gamma \Vdash^{\mathsf{v}}_\ell A$.
- $\Gamma \Vdash^{\mathsf{v}}_\ell A = B$ iff $\Vdash^{\mathsf{v}} \Gamma$ and furthermore, for all contexts Δ and reducibly equal substitutions $\Delta \Vdash^{\mathsf{s}} \sigma_1 = \sigma_2 : \Gamma$, we have $\Delta \Vdash_\ell A[\sigma_1] = B[\sigma_2]$.

Valid terms are defined analogously:

- $\boxed{\Gamma \Vdash^{\mathsf{v}}_\ell t = u : A}$ iff $\Gamma \Vdash^{\mathsf{v}}_\ell A$ and furthermore, for all contexts Δ and reducibly equal substitutions $\Delta \Vdash^{\mathsf{s}} \sigma_1 = \sigma_2 : \Gamma$, we have $\Delta \Vdash_\ell t[\sigma_1] = u[\sigma_2] : A[\sigma_1]$.
- $\boxed{\Gamma \Vdash^{\mathsf{v}}_\ell t : A}$ iff $\Gamma \Vdash^{\mathsf{v}}_\ell t = t : A$.

[8] We changed it to accommodate the definition of unary reducibility for terms in terms of binary reducibility.
[9] The categorically-minded reader may recognise this as one of the central features of the *gluing* construction [Kaposi et al. 2019; Altenkirch and Kaposi 2017], although we do not attempt to make the connection precise.

We can now state the fundamental lemma:

Lemma 3.3. *(**Fundamental lemma**) Well-typed objects are valid:*

- $\vdash \Gamma$ *implies* $\Vdash^{\mathsf{v}} \Gamma$.
- $\Gamma \vdash A$ *implies* $\Gamma \Vdash^{\mathsf{v}}_{\ell} A$ *for some* ℓ.
- $\Gamma \vdash A = B$ *implies* $\Gamma \Vdash^{\mathsf{v}}_{\ell} A = B$ *for some* ℓ.
- $\Gamma \vdash t : A$ *implies* $\Gamma \Vdash^{\mathsf{v}}_{\ell} t : A$ *for some* ℓ.
- $\Gamma \vdash t = u : A$ *implies* $\Gamma \Vdash^{\mathsf{v}}_{\ell} t = u : A$ *for some* ℓ.

The fundamental lemma is proved by induction on the typing judgements. We have already presented some of the cases for the typing rules related to levels at the end of Section 3.1; they lift straightforwardly to validity judgements. We sketch some of the other cases that are affected by the addition of internal levels:

Lemma 3.4. *(Validity for U) If* $\Gamma \Vdash^{\mathsf{v}}_{\ell} t : \mathsf{Level}$, *then* $\Gamma \Vdash^{\mathsf{v}}_{\omega} \mathcal{U}_t : \mathcal{U}_{t^+}$.

Proof. By lifting the analogous statement about reducibility, which holds because $\uparrow t < \uparrow t^+ < \omega$, to validity. □

Lemma 3.5. *(Validity for Univ) If* $\Gamma \Vdash^{\mathsf{v}}_{\ell} A : \mathcal{U}_t$, *then* $\Gamma \Vdash^{\mathsf{v}}_{\ell} A$.

Proof. By lifting the analogous statement about reducibility, which holds by cumulativity of reducibility using the fact that $\Gamma \Vdash_{\uparrow t} A$ and $\uparrow t < \ell$, to validity. □

Lemma 3.6. *(Validity for Lift-U) If* $\Gamma \Vdash^{\mathsf{v}}_{\ell_1} t : \mathsf{Level}$, $\Gamma \Vdash^{\mathsf{v}}_{\ell_2} u : \mathsf{Level}$, *and* $\Gamma \Vdash^{\mathsf{v}}_{\ell_3} A : \mathcal{U}_t$, *then* $\Gamma \Vdash^{\mathsf{v}}_{\omega} \mathsf{Lift}_u\, A : \mathcal{U}_{t \sqcup u}$.

Proof. By lifting the analogous statement about reducibility, which holds by cumulativity of reducibility using the fact that $\Gamma \Vdash_{\uparrow t} A$ and $\uparrow t \leq \uparrow(t \sqcup u) < \omega$, to validity. □

Lemma 3.7. *(Validity for Pi-U) If* $\Gamma \vdash \Pi\, A\, B : \mathcal{U}_t$, $\Gamma \Vdash^{\mathsf{v}}_{\ell_1} t : \mathsf{Level}$, $\Gamma \Vdash^{\mathsf{v}}_{\ell_2} A : \mathcal{U}_t$, *and* $\Gamma, A \Vdash^{\mathsf{v}}_{\ell_3} B : \mathcal{U}_{t[\mathsf{p}]}$, *then* $\Gamma \Vdash^{\mathsf{v}}_{\omega} \Pi\, A\, B : \mathcal{U}_t$.

Proof. The proof uses the validity assumptions to derive reducible equality for A and B with weakenings and substitutions applied. It also uses the fact that level realisers are stable under weakening, i.e. that $\uparrow(t[\rho]) = \uparrow t$ for any well-formed weakening ρ. □

As an immediate consequence of the fundamental lemma we obtain the following result:

Corollary 3.8. *Well-typed objects are reducible:* $\Gamma \vdash \mathcal{J}$ *implies* $\Gamma \Vdash_{\ell} \mathcal{J}$ *for some* ℓ, *for each of the four typing judgements* $\mathcal{J}$.

Proof. This follows from the fact that, for a valid context Γ, the identity substitution is reducible ($\Gamma \Vdash^{\mathsf{s}} \mathrm{id} = \mathrm{id} : \Gamma$). □

Instantiating this corollary with judgemental equality as generic equality gives the following corollaries, among other results:

Corollary 3.9. *(Consistency) There is no closed term* t *such that* $\cdot \vdash t : \bot$, *where* $\bot$ *is the empty type.*

Corollary 3.10. *(Canonicity) For every closed term* $\cdot \vdash t : \mathbb{N}$, *there is an external natural number* n *such that* $\cdot \vdash t = \mathrm{suc}^n\, 0 : \mathbb{N}$.

Proof. Both consistency and canonicity follow from the fact that there are no neutral terms in the empty context. □

Corollary 3.11. *(Weak head normalisation) Every type* $\Gamma \vdash A$ *and every term* $\Gamma \vdash t : A$ *reduce to a weak head normal form.*

Corollary 3.12. *(Injectivity of and non-confusion for type formers)*

- $\Gamma \vdash \mathcal{U}_t = \mathcal{U}_u$ *implies* $\Gamma \vdash t = u : \mathsf{Level}$,
- $\Gamma \vdash \mathsf{Lift}_t\, A = \mathsf{Lift}_u\, B$ *implies* $\Gamma \vdash t = u : \mathsf{Level}$ *and* $\Gamma \vdash A = B$,
- $\Gamma \vdash \Pi\, A\, B = \Pi\, A'\, B'$ *implies* $\Gamma \vdash A = A'$ *and* $\Gamma, A \vdash B = B'$,
- $\Gamma \vdash \mathsf{Level} = \mathbb{N}$ *is not derivable*,
- $\Gamma \vdash \mathcal{U}_t = \mathbb{N}$ *is not derivable*,
- *and so on.*

We prove a generalisation of Corollary 3.10 for contexts containing only level and type variables:

Corollary 3.13. *For every context* Γ *consisting only of variables of type* Level *or* $\mathcal{U}_t$*, and for every term* $\Gamma \vdash u : \mathbb{N}$*, there is an external natural number* n *such that* $\Gamma \vdash u = \mathsf{suc}^n\, 0 : \mathbb{N}$.

Proof. An atomic neutral term in such a context is necessarily a variable, since none of the eliminators (except for $\sqcup$) can be stuck on a level or a type. But no variable in such a context can have type $\mathbb{N}$, by Corollary 3.12. □

4 Decidability of Equality

Following Abel et al. [2017], we show decidability of judgemental equality using a second instantiation of the fundamental lemma (Lemma 3.3) with *algorithmic equality*: an inductive specification of a bidirectional conversion checking algorithm [Lennon-Bertrand 2025]. In more detail, algorithmic equality consists of several relations, defined mutually inductively, which explain how to reduce terms and types to weak head normal forms and compare them recursively:

- $\boxed{\Gamma \vdash A \mathrel{\hat{\Longleftrightarrow}} B}$ and $\boxed{\Gamma \vdash A \Longleftrightarrow B}$ for arbitrary types and types in WHNF respectively,
- $\boxed{\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : A}$ and $\boxed{\Gamma \vdash t \Longleftrightarrow u : A}$ for arbitrary terms with arbitrary types and terms in WHNF with types in WHNF respectively, and
- $\boxed{\Gamma \vdash t \mathrel{\hat{\longleftrightarrow}} u : A}$ and $\boxed{\Gamma \vdash t \longleftrightarrow u : A}$ for atomic neutral terms and atomic neutral terms with types in WHNF respectively.

The algorithm extends straightforwardly to Lift types and multiple universes, so we focus on the main addition: the decision procedure for level equality. We take inspiration from Agda's internal design and use the following approach: two level expressions are compared by normalising them to partially canonical *views* of the form $\bigsqcup_{0 \le i < n} k_i + a_i$, where a_i is an *atomic level* ($\mathbf{0}$ or an atomic neutral), and using the fact that $\bigsqcup_i x_i \le \bigsqcup_j y_j$ if and only if $\forall i.\ \exists j.\ x_i \le y_j$. This boils down to comparing atomic neutrals for equality, so this procedure is defined mutually inductively with the rest of the algorithm. We define $\boxed{\mathsf{LevelAtom}\,\Gamma}$, $\boxed{\mathsf{Level}^+\,\Gamma}$, and $\boxed{\mathsf{Level}^{\mathsf{v}}\,\Gamma}$ as follows:

$$\frac{}{\mathbf{0} \in \mathsf{LevelAtom}\,\Gamma} \qquad \frac{\Gamma \vdash t \longleftrightarrow t : \mathsf{Level}}{t \in \mathsf{LevelAtom}\,\Gamma} \qquad \frac{n \in \mathbb{N} \qquad a \in \mathsf{LevelAtom}\,\Gamma}{(n + a) \in \mathsf{Level}^+\,\Gamma}$$

$$\frac{}{\mathbf{0} \in \mathsf{Level}^{\mathsf{v}}\,\Gamma} \qquad \frac{t \in \mathsf{Level}^+\,\Gamma \qquad u \in \mathsf{Level}^{\mathsf{v}}\,\Gamma}{(t \sqcup u) \in \mathsf{Level}^{\mathsf{v}}\,\Gamma}$$

Note that an element of $\mathsf{Level}^{\mathsf{v}}\,\Gamma$ is simply a list of $\mathsf{Level}^+\,\Gamma$.

The next step is to explain how level views are compared. We define, still mutually inductively, binary relations $\boxed{=^{\mathsf{v}}}$ and $\boxed{\le^{\mathsf{v}}}$ on $\mathsf{Level}^{\mathsf{v}}\,\Gamma$, $\boxed{\le^{+\mathsf{v}}}$ between $\mathsf{Level}^+\,\Gamma$ and $\mathsf{Level}^{\mathsf{v}}\,\Gamma$, $\boxed{\le^+}$ on $\mathsf{Level}^+\,\Gamma$, and $\boxed{\le^{\mathsf{a}}}$ on $\mathsf{LevelAtom}\,\Gamma$, as follows:

$$\frac{t \leq^{\mathsf{v}} u \qquad u \leq^{\mathsf{v}} t}{t =^{\mathsf{v}} u} \qquad \frac{}{\mathbf{0} \leq^{\mathsf{v}} v} \qquad \frac{t \leq^{+\mathsf{v}} v \qquad u \leq^{\mathsf{v}} v}{t \sqcup u \leq^{\mathsf{v}} v} \qquad \frac{t \leq^{+} u}{t \leq^{+\mathsf{v}} u \sqcup v} \qquad \frac{t \leq^{+\mathsf{v}} v}{t \leq^{+\mathsf{v}} u \sqcup v}$$

$$\frac{n \leq m \qquad t \leq^{\mathsf{a}} u}{n + t \leq^{+} m + u} \qquad \frac{}{\mathbf{0} \leq^{\mathsf{a}} t} \qquad \frac{\Gamma \vdash t \longleftrightarrow u : \mathsf{Level}}{t \leq^{\mathsf{a}} u}$$

Finally, we explain how to convert a level to a level view via the relations $\boxed{\Gamma \vdash t \mathrel{\hat{\Longrightarrow}}^{\mathsf{v}} v}$ for arbitrary levels, $\boxed{\Gamma \vdash t \Longrightarrow^{\mathsf{v}} v}$ for levels in WHNF, and $\boxed{\Gamma \vdash t \longrightarrow^{\mathsf{v}} v}$ for neutral levels, where $v \in \mathsf{Level}^{\mathsf{v}}\,\Gamma$. These relations are defined inductively as follows, where the successor operation $\boxed{v^{+}}$ on a $\mathsf{Level}^{\mathsf{v}}\,\Gamma$ is defined by adding 1 to every $\mathsf{Level}^{+}\,\Gamma$ in the list[10] and the supremum operation $\boxed{v \sqcup^{\mathsf{v}} v'}$ on $\mathsf{Level}^{\mathsf{v}}\,\Gamma$ is defined as list concatenation:

$$\frac{\Gamma \vdash t \longrightarrow^{*} \bar{t} : \mathsf{Level} \qquad \bar{t} \in \mathsf{Whnf}\, n \qquad \Gamma \vdash \bar{t} \Longrightarrow^{\mathsf{v}} v}{\Gamma \vdash t \mathrel{\hat{\Longrightarrow}}^{\mathsf{v}} v}$$

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathbf{0} \Longrightarrow^{\mathsf{v}} \mathbf{0}} \qquad \frac{\Gamma \vdash t \mathrel{\hat{\Longrightarrow}}^{\mathsf{v}} v}{\Gamma \vdash t^{+} \Longrightarrow^{\mathsf{v}} v^{+}} \qquad \frac{\Gamma \vdash t \longrightarrow^{\mathsf{v}} v}{\Gamma \vdash t \Longrightarrow^{\mathsf{v}} v}$$

$$\frac{\Gamma \vdash t \longrightarrow^{\mathsf{v}} v \qquad \Gamma \vdash u \mathrel{\hat{\Longrightarrow}}^{\mathsf{v}} v'}{\Gamma \vdash t \sqcup u \longrightarrow^{\mathsf{v}} v \sqcup^{\mathsf{v}} v'} \qquad \frac{\Gamma \vdash t \mathrel{\hat{\Longrightarrow}}^{\mathsf{v}} v \qquad \Gamma \vdash u \longrightarrow^{\mathsf{v}} v'}{\Gamma \vdash t^{+} \sqcup u \longrightarrow^{\mathsf{v}} v^{+} \sqcup^{\mathsf{v}} v'} \qquad \frac{\Gamma \vdash t \longleftrightarrow t : \mathsf{Level}}{\Gamma \vdash t \longrightarrow^{\mathsf{v}} (0 + t) \sqcup \mathbf{0}}$$

The relation $\Gamma \vdash t \longleftrightarrow u : \mathsf{Level}$ is then defined to hold if and only if $\Gamma \vdash t \Longrightarrow^{\mathsf{v}} v$ and $\Gamma \vdash u \Longrightarrow^{\mathsf{v}} v'$ with $v =^{\mathsf{v}} v'$. We have checked that the algorithmic equality extended in this way is sound in an appropriate sense, stable under weakening, decidable given that both sides are algorithmically equal to some type or term,[11] and that the normalisation of levels is deterministic with respect to a notion of syntactic equality for level views.

It only remains to show that algorithmic equality is still an instance of the generic equality relation, which includes showing certain equalities involving $\sqcup$ (associativity, idempotence, etc.). The following lemma reduces this to the task of checking those equalities for level views:

Lemma 4.1. *If* $\Gamma \vdash t \mathrel{\hat{\Longrightarrow}}^{\mathsf{v}} v_1$ *and* $\Gamma \vdash u \mathrel{\hat{\Longrightarrow}}^{\mathsf{v}} v_2$*, then there exists* $v \in \mathsf{Level}^{\mathsf{v}}\,\Gamma$ *such that* $\Gamma \vdash t \sqcup u \mathrel{\hat{\Longrightarrow}}^{\mathsf{v}} v$ *and* $v =^{\mathsf{v}} v_1 \sqcup^{\mathsf{v}} v_2$.

Proof. By recursion on the witnesses of normalisation, analogously to the proofs of Lemmas 3.1 and 3.2, and using basic properties of $=^{\mathsf{v}}$. □

Reaping the fruits of this section, we now instantiate the fundamental lemma (Lemma 3.3) with the algorithmic equality relations to obtain the following:

Theorem 4.2. *Algorithmic equality is complete with respect to judgemental equality:*

- $\Gamma \vdash A = B$ *implies* $\Gamma \vdash A \mathrel{\hat{\Longleftrightarrow}} B$.
- $\Gamma \vdash t = u : A$ *implies* $\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : A$.

Corollary 4.3. *Judgemental equality of well-formed types and terms is decidable.*

Proof. By decidability of algorithmic equality applied to witnesses of equality obtained from completeness of algorithmic equality and reflexivity. □

[10] We also prepend $1 + \mathbf{0}$ in order to avoid treating the empty list differently.

[11] For $\Gamma \vdash t \mathrel{\hat{\longleftrightarrow}} u : A$ and $\Gamma \vdash t \longleftrightarrow u : A$ it is decidable whether there is some A such that these relations hold.

$$\frac{A, B \text{ checkable-type}}{\Pi\, A\, B \text{ checkable-type}} \qquad \frac{t \text{ checkable} \quad A \text{ checkable-type}}{\mathsf{Lift}_t\, A \text{ checkable-type}} \qquad \frac{A \text{ checkable}}{A \text{ checkable-type}}$$

$$\frac{t \text{ checkable}}{\lambda\, t \text{ checkable}} \qquad \frac{t \text{ checkable}}{\mathsf{lift}\, t \text{ checkable}} \qquad \frac{t \text{ inferable}}{t \text{ checkable}}$$

$$\frac{}{x_i \text{ inferable}} \qquad \frac{A \text{ inferable} \quad B \text{ checkable}}{\Pi\, A\, B \text{ inferable}} \qquad \frac{t \text{ inferable} \quad u \text{ checkable}}{t\, u \text{ inferable}}$$

$$\frac{}{\mathsf{Level} \text{ inferable}} \qquad \frac{}{\mathbf{0} \text{ inferable}} \qquad \frac{t \text{ checkable}}{t^+ \text{ inferable}} \qquad \frac{t, u \text{ checkable}}{t \sqcup u \text{ inferable}} \qquad \frac{t \text{ checkable}}{\mathcal{U}_t \text{ inferable}}$$

$$\frac{t \text{ checkable} \quad A \text{ inferable}}{\mathsf{Lift}_t\, A \text{ inferable}} \qquad \frac{t \text{ inferable}}{\mathsf{lower}\, t \text{ inferable}} \qquad \ldots$$

Fig. 6. Selected rules for checkable types, checkable terms, and inferable terms.

Corollary 4.4. *(Deep normalisation) Well-formed types and terms have η-long normal forms.*

Proof. By recursion on witnesses of algorithmic equality obtained from completeness of algorithmic equality and reflexivity. □

Abel et al. [2023] classify certain terms as *checkable* and/or *inferable* (with every inferable term being checkable). These classes exclude, for instance, β-redexes: note that lambdas and applications are not annotated with type information. Abel et al. use a bidirectional type-checking algorithm to prove that, under a context of types that are checkable terms, type inference is decidable for an inferable term, and it is decidable whether a checkable term has a given type that is a checkable term.

With the introduction of a universe hierarchy some things become a little harder. Abel et al. define the term $\Pi\, A\, B$ to be inferable if A and B are checkable: the only way for $\Pi\, A\, B$ to be a well-typed term is if A and B both have type $\mathcal{U}$, so it suffices for A and B to be checkable (against the type $\mathcal{U}$). In our setting $\Pi\, A\, B$ is a well-typed term under Γ if there is some t such that $\Gamma \vdash t : \mathsf{Level}$, $\Gamma \vdash A : \mathcal{U}_t$ and $\Gamma, A \vdash B : \mathcal{U}_{t[\mathrm{p}]}$. Thus we define $\Pi\, A\, B$ to be inferable if A is inferable and B is checkable: in the algorithm we first try to infer a type C for A, then we check if C reduces to a WHNF $\mathcal{U}_t$ for some t, and finally we check if B has the type $\mathcal{U}_{t[\mathrm{p}]}$.

However, the question of whether the term $\Pi\, A\, B$ *is* a well-formed type is different from the question of whether it *has* a type. Note that $\Pi\, A\, B$ is a well-formed type under Γ if $\Gamma, A \vdash B$ holds (and Γ, A is well-formed if $\Gamma \vdash A$ holds). We introduce the notion of a *checkable type*, and define $\Pi\, A\, B$ to be in that class if both A and B are.

We define checkable types $\boxed{A \text{ checkable-type}}$, checkable terms $\boxed{t \text{ checkable}}$ and inferable terms $\boxed{t \text{ inferable}}$ mutually inductively. Rules for the fragment of the language that we focus on can be found in Figure 6.[12] Note that $\mathsf{lift}\, t$ is not inferable due to the absence of a level annotation.

[12]It may appear as if the classes of checkable and inferable terms are defined without reference to the class of checkable types, but the rules for several omitted eliminators include such references. See the source code for details.

The class of checkable types is strictly larger than the class of checkable terms: every checkable term is a checkable type, but the term $\Pi\,(\lambda\,x_0)\,x_0$, which is a checkable type, is not checkable. Note that this term is not a well-formed type. We have proved that every well-formed, checkable type is inferable.

With the definitions discussed above in place we can state the following theorem:

Theorem 4.5. *(Typing is decidable)*

- *If* Γ *is a context of checkable types, then it is decidable whether* $\vdash \Gamma$.
- *If* A *is a checkable type and* Γ *is a context of checkable types, then it is decidable whether* $\Gamma \vdash A$.
- *If* t *is a checkable term,* A *is a checkable type, and* Γ *is a context of checkable types, then it is decidable whether* $\Gamma \vdash t : A$.
- *If* t *is an inferable term and* Γ *is a context of checkable types, then it is decidable whether there exists an* A *such that* $\Gamma \vdash t : A$.

5 Erasing Levels Is Safe

There is no mechanism for making run-time decisions based on the value of a universe level, so it should be safe to erase them as part of the compilation process. Building on the work of Abel et al. [2023] we show that an extraction procedure that takes terms in (a variant of) our type theory to an untyped λ-calculus is safe in the sense that closed (and well-resourced) terms of type $\mathbb{N}$ that reduce to a given numeral n in the source language reduce to the same numeral in the target language. The target language of Abel et al. uses call-by-name. We additionally support call-by-value.

Abel et al. [2023] present a *graded* type theory, where the syntax is annotated with grades and there is a *usage relation* $\boxed{\gamma \triangleright^m t}$ that classifies "well-resourced" terms t under a usage context γ and mode m. Their Theorem 8.3, establishing soundness of extraction, is given for an arbitrary *modality with a well-behaved zero*, and our formalisation takes basically the same approach. Here we instead work with a fixed modality, the *erasure modality*. This modality has two grades: 0, which stands for "not used at run-time", and ω, which stands for "used an arbitrary number of times".[13] The usage contexts associate a grade with every variable, providing (approximate) information about how many times the variables are used. The modes are 0 and 1: if the mode is 0, then variables can be used freely, and if the mode is 1, then variable uses are counted.

We will not present the full details of the graded type theory here, but we can note that Π-types are annotated with a grade that states how many times a variable may be used in the "run-time parts" of the body of a lambda: zero times or an arbitrary number of times. Lambdas and applications are also annotated with this kind of grade. (Π-types are in fact annotated with a second grade as well.)

We have added levels and lifting to the type theory. In the graded type theory there are no grade annotations on the level or lifting term formers, and their typing rules are unchanged. Their usage rules are given below, along with the one for $\mathcal{U}$:

$$\frac{\gamma \triangleright^0 t}{0^{\mathrm{c}} \triangleright^m \mathcal{U}_t} \qquad \frac{}{0^{\mathrm{c}} \triangleright^m \mathsf{Level}} \qquad \frac{}{0^{\mathrm{c}} \triangleright^m \mathbf{0}} \qquad \frac{\gamma \triangleright^m t}{\gamma \triangleright^m t^+} \qquad \frac{\gamma \triangleright^m t \qquad \delta \triangleright^m u}{\gamma +^{\mathrm{c}} \delta \triangleright^m t \sqcup u}$$

$$\frac{\gamma \triangleright^0 t \qquad \delta \triangleright^m A}{\delta \triangleright^m \mathsf{Lift}_t\, A} \qquad \frac{\gamma \triangleright^m u}{\gamma \triangleright^m \mathsf{lift}\, u} \qquad \frac{\gamma \triangleright^m t}{\gamma \triangleright^m \mathsf{lower}\, t} \qquad \ldots$$

Here 0^{c} is a usage context where every entry is 0, and $\gamma +^{\mathrm{c}} \delta$ is implemented using pointwise addition: 0 is an identity element for addition, and $\omega + \omega = \omega$. Note that the premises corresponding

[13] Not to be confused with the universe level ω, which plays no role in this section.

to the level arguments of $\mathcal{U}$, Lift and lower have the mode 0: variables can be used freely in these arguments.

Let us now discuss the target language. Here is a selection of its syntactic constructions (the formalisation uses well-scoped syntax, but those details are omitted here):

$$t, u ::= \lightning \mid \mathsf{var}\, x \mid \mathsf{lam}\, t \mid \mathsf{app}^s\, t\, u \mid \mathsf{zero} \mid \mathsf{suc}\, t \mid \ldots$$

We have omitted some terms related to $\mathbb{N}$, Σ-types and unit types. The term $\lightning$ is a value that we (sometimes) use when extracting things that should be erased. Variables x are natural number literals: we use de Bruijn indices. Applications come in two forms: the *strictness* s can be strict or non-strict.

Certain terms are seen as values (pair and unit constructors have been omitted):

$$\frac{}{\mathsf{Value}\ \lightning} \qquad \frac{}{\mathsf{Value}\ (\mathsf{lam}\, t)} \qquad \frac{}{\mathsf{Value\ zero}} \qquad \frac{}{\mathsf{Value}\ (\mathsf{suc}\, t)} \qquad \ldots$$

We also use an auxiliary predicate $\mathsf{Value}^s\ t$ that holds vacuously when s is non-strict, and is equal to $\mathsf{Value}\ t$ when s is strict.

A small-step semantics is defined inductively in the following way (rules relating to $\mathbb{N}$, Σ-types and unit types are omitted):

$$\frac{t \Rightarrow t'}{\mathsf{app}^s\, t\, u \Rightarrow \mathsf{app}^s\, t'\, u} \qquad \frac{\mathsf{Value}\ t \qquad u \Rightarrow u'}{\mathsf{app}^{\mathsf{strict}}\, t\, u \Rightarrow \mathsf{app}^{\mathsf{strict}}\, t\, u'} \qquad \frac{\mathsf{Value}^s\ u}{\mathsf{app}^s\, (\mathsf{lam}\, t)\, u \Rightarrow t[u/0]} \qquad \ldots$$

The β-rule uses substitution (the definition of which is omitted). Note that strict applications only reduce if the second argument is a value: the second argument of an application can reduce prior to β-reduction only if the application is strict.

The extraction function takes a strictness argument. We use the value $\lightning$ only in the strict case, where it matters whether an unused function argument is a value or not. In the non-strict case we instead use the non-terminating term loop, which is defined in the following way (basically $(\lambda x.xx)(\lambda x.xx)$):

$$\mathsf{app}^{\mathsf{non\text{-}strict}}\ (\mathsf{lam}\ (\mathsf{app}^{\mathsf{non\text{-}strict}}\ (\mathsf{var}\ 0)\ (\mathsf{var}\ 0)))\ (\mathsf{lam}\ (\mathsf{app}^{\mathsf{non\text{-}strict}}\ (\mathsf{var}\ 0)\ (\mathsf{var}\ 0)))$$

We use loop to highlight that certain subterms produced by the extraction function are not reduced to weak head normal form: they could be replaced by anything. The notation $\lightning^s$ stands for a term that is equal to $\lightning$ when s is strict, and equal to loop when s is non-strict.

The extraction function is defined recursively in the following way (some cases related to $\mathbb{N}$, Σ-types, empty, unit and identity types have been omitted):

erase s	x_i	$= \mathsf{var}\, i$	erase s zero	$= \mathsf{zero}$
erase s	$(\mathsf{lift}\, u)$	$= \mathsf{erase}\ s\ u$	erase s $(\mathsf{suc}\, t)$	$= \mathsf{suc}^s\ (\mathsf{erase}\ s\ t)$
erase s	$(\mathsf{lower}\, t)$	$= \mathsf{erase}\ s\ t$	erase s Level	$= \lightning^s$
erase s	$(t\ ^{\omega}\ u)$	$= \mathsf{app}^s\ (\mathsf{erase}\ s\ t)\ (\mathsf{erase}\ s\ u)$	erase s $\mathbf{0}$	$= \lightning^s$
erase strict	$(t\ ^{0}\ u)$	$= \mathsf{app}^{\mathsf{strict}}\ (\mathsf{erase\ strict}\ t)\ \lightning$	erase s (t^+)	$= \lightning^s$
erase non-strict	$(t\ ^{0}\ u)$	$= \mathsf{erase\ non\text{-}strict}\ t$	erase s $(t \sqcup u)$	$= \lightning^s$
erase s	$(\lambda^{\omega}\, t)$	$= \mathsf{lam}\ (\mathsf{erase}\ s\ t)$	erase s $\mathcal{U}_t$	$= \lightning^s$
erase strict	$(\lambda^{0}\, t)$	$= \mathsf{lam}\ (\mathsf{erase\ strict}\ t)$	erase s $(\mathsf{Lift}_t\, A)$	$= \lightning^s$
erase non-strict	$(\lambda^{0}\, t)$	$= (\mathsf{erase\ non\text{-}strict}\ t)[\mathsf{loop}/0]$	erase s $(\Pi^q_p\, A\, B)$	$= \lightning^s$

Note that every application of a type constructor is erased, as is the case for every application of one of the level primitives. The constructor lift and the projection lower are removed, but their arguments are (potentially) kept. In the non-strict setting an erased function argument is removed entirely (in a departure from the work of Abel et al. [2023]), and a lambda with an erased

argument is replaced by the extraction of its body (with loop substituted for the bound variable). In the strict setting erased function arguments are instead replaced by $\lightning$: complete removal of all function arguments in some cases leads to non-termination for the extracted program, as discussed in Section 1.2. In the case for suc the notation $\mathsf{suc}^s\ t$ is used: it stands for $\mathsf{suc}\ t$ if s is non-strict, and $\mathsf{app}^{\mathsf{strict}}\ (\mathsf{lam}\ (\mathsf{suc}\ (\mathsf{var}\ 0)))\ t$ if s is strict. This construction ensures that, in the strict case, applications of the successor constructor are translated to strict applications.

The reflexive, transitive closure of $\Rightarrow$ only gives us reduction to weak head normal form, and does not include reduction under constructors. In the strict case this is addressed through the use of $\mathsf{suc}^{\mathsf{strict}}\ t$ (and a similar construction for pairs). In the non-strict case we instead use the following relation, which allows reduction under successor constructors:

$$\frac{t \Rightarrow u}{t \Rightarrow^{\mathsf{suc}} u} \qquad \frac{t \Rightarrow^{\mathsf{suc}} u}{\mathsf{suc}\ t \Rightarrow^{\mathsf{suc}} \mathsf{suc}\ u}$$

There is also a corresponding relation for the source language:

$$\frac{\Gamma \vdash t \longrightarrow u : \mathbb{N}}{\Gamma \vdash t \longrightarrow^{\mathsf{suc}} u : \mathbb{N}} \qquad \frac{\Gamma \vdash t \longrightarrow^{\mathsf{suc}} u : \mathbb{N}}{\Gamma \vdash \mathsf{suc}\ t \longrightarrow^{\mathsf{suc}} \mathsf{suc}\ u : \mathbb{N}}$$

The soundness theorem below is stated using the reflexive, transitive closure of $\Gamma \vdash - \longrightarrow^{\mathsf{suc}} - : \mathbb{N}$, denoted by $\Gamma \vdash - \longrightarrow^{\mathsf{suc}*} - : \mathbb{N}$. The theorem statement also uses $\Rightarrow_s^*$, which when s is strict stands for the reflexive, transitive closure of $\Rightarrow$, and when s is non-strict stands for the reflexive, transitive closure of $\Rightarrow^{\mathsf{suc}}$. The statement also uses the notation $\underline{n}$, which for a natural number n stands for the term suc^n zero, with n applications of suc to zero, in both the source and the target language.

Building on the proof by Abel et al. [2023][14] we have proved the following theorem ("erased matches" are explained below):[15]

Theorem 5.1. *(Soundness of extraction) If*

- *whenever erased matches are allowed for the empty type, then* Γ *is consistent (i.e. there is no term* u *such that* $\Gamma \vdash u : \bot$*),*
- *either* Γ *is empty, or erased matches are disallowed for weak* Σ *and unit types,*
- $\Gamma \vdash t : \mathbb{N}$*, and*
- $0^c \triangleright^1 t$*,*

then there is a natural number n *such that* $\Gamma \vdash t \longrightarrow^{\mathsf{suc}*} \underline{n} : \mathbb{N}$*, and, for any strictness* s*,* $\mathsf{erase}\ s\ t \Rightarrow_s^* \underline{n}$*.*

Our additions to the proof related to levels, lifting and strictness are straightforward, so we do not present them here. Note that if we restrict ourselves to empty contexts, then we get the following variant of the theorem, without any mention of erased matches or consistency:

Corollary 5.2. *If* $\cdot \vdash t : \mathbb{N}$ *and* $\cdot \triangleright^1 t$ *then there is a natural number* n *such that* $\cdot \vdash t \longrightarrow^{\mathsf{suc}*} \underline{n} : \mathbb{N}$*, and, for any strictness* s*,* $\mathsf{erase}\ s\ t \Rightarrow_s^* \underline{n}$*.*

The main formulation of the theorem allows terms with free variables, given that they all have usage count 0, and that some other conditions hold. This means that a program that uses *postulates* – for instance excluded middle – is guaranteed to terminate with a numeral, as long as the postulates are only used in "erased settings" (and the other conditions hold).

[14] And later additions, some of which are discussed in Section 2.

[15] The formalised theorem includes some conditions related to identity types, but identity types are not the focus of this paper.

So, what is an "erased match" [Abel et al. 2023]?

- For data types with a single constructor, like the (weak) Σ and unit types (without η-equality), one might want to allow matches on erased data in non-erased settings, as in the following Agda code (@0 marks an argument as erased):

  $$f : @0\ \mathbb{N} \times \mathbb{N} \to \dots$$
  $$f\ (x, y) = \dots$$

  Here $\times$ is a pair type without η-equality, and x and y are erased on the right-hand side. If one combines this feature with "erased postulates", then the theorem fails, because a program can get stuck on one of the postulates.
- For data types with *zero* constructors, i.e. empty types, one might also want to allow matches on erased data in non-erased settings, as in the following Agda code:

  $$\bot\text{-elim} : @0\ \bot \to A$$
  $$\bot\text{-elim}\ ()$$

  With this feature one can use erased information to discard impossible branches. However, if one allows both this feature and inconsistent postulates, then the theorem fails, because an "impossible" branch might not actually be impossible.

6 Discussion and Future Work

We have proved that a type theory with first-class universe levels inspired by Agda enjoys desirable metatheoretical properties, including normalisation and decidability of equality. We now discuss some possible extensions of this work.

Universes beyond ω. We expect that our results could extend to a second countable hierarchy of universes $\mathcal{U}_\omega : \mathcal{U}_{\omega+1} : \cdots$, as in Agda 2.6.2 [The Agda Team 2021], by indexing universes and the lifting operator by either level expressions or external levels.

Universes à la Tarski. We have followed Abel et al. [2017] in presenting universes *à la Russell*, i.e. silently coercing between elements of a universe $\Gamma \vdash A : \mathcal{U}_t$ and types $\Gamma \vdash A$. It seems reasonable to expect that universes *à la Tarski*, with an explicit type decoding operation $\Gamma \vdash \mathsf{El}_t(A)$, could also be supported with straightforward changes to our proofs.

Universes à la Coquand. We expect that our development could be modified to use the presentation of universes *à la Coquand* [Coquand 2012; Angiuli and Gratzer 2025], in which the typing judgement itself is stratified according to the universe hierarchy. In the presence of internal universe levels, this could mean introducing typing judgements of the form $\Gamma \vdash_t A$, where t is either a term or an external level $t \geq \omega$, such that $\Gamma \vdash A : \mathcal{U}_t$ holds if and only if $\Gamma \vdash_t A$. These typing judgements should then be precisely modelled by validity judgements of the form $\Gamma \Vdash^{\mathsf{v}}_t A$, equivalent to "$\Vdash^{\mathsf{v}} \Gamma$ and, for all contexts Δ and reducibly equal substitutions $\Delta \Vdash^{\mathsf{s}} \sigma_1 = \sigma_2 : \Gamma$, $\Delta \Vdash_{\uparrow(t[\sigma_1])} A[\sigma_1] = A[\sigma_2]$". By contrast, the current statement of our fundamental lemma involves an existential quantification over levels, and ω is sometimes used as a "catch-all" level.

Natural numbers as levels. Taking inspiration from Kovács [2022], one could hope to use the type of natural numbers itself as the type of universe levels. This would allow one to define functions on levels by recursion. A natural question is then whether the operator $\sqcup$ could be defined recursively instead of being taken as a primitive. It is unclear to us how an internal definition of $\sqcup$ would work with the approach presented here, since the level realisation function defined in Section 3.1 would apparently have to recognise the internally-defined $\sqcup$ in order to satisfy the

equality $\uparrow(t \sqcup u) = \uparrow t \sqcup \uparrow u$. With natural numbers as levels it would also be unsafe to erase all level primitives as in Section 5.

Level constraints. One of the type theories of Bezem et al. [2023] features level constraint assumptions: $i < j$ implies $\mathcal{U}_i : \mathcal{U}_j$. One may wonder if our type theory could be extended with something similar. As discussed by Chan and Weirich [2025], such constraint assumptions could be an obstacle to the goal of normalisation, since in the context $t : \mathsf{Level}$, $loop : t < t$ one has $\mathcal{U}_t : \mathcal{U}_t$, an instance of type-in-type, which allows the construction of a diverging term. Bezem et al. [2023] require that the constraint assumptions are loop-free, but subject reduction does not hold for their theory [Bezem et al. 2024].

Bounded universe polymorphism. Short of fully internalised level constraints, one might want to support *bounded* universe polymorphism as in TTBFL [Chan and Weirich 2025], which includes a type $\mathsf{Level}_{<t}$ of levels strictly less than t. While this sidesteps the issue of looping constraints, there is still a difficulty regarding normalisation: in an inconsistent context, every type is inhabited, including $\mathsf{Level}_{<t}$ for any t. Thus, in inconsistent contexts, there is a countably infinite *decreasing* chain of levels below $\mathbf{0}$, which could be problematic if the universe hierarchy is to be modelled by well-founded induction as in our approach. We speculate that one could perhaps address this by using a type theory with a non-well-founded universe hierarchy (as proven consistent by Hou (Favonia) et al. [2023]) as the metatheory.

Level annotations. An earlier version of our type theory had a level annotation on the lift constructor, e.g. $\mathsf{lift}_t\ u : \mathsf{Lift}_t\ A$. This turned out to cause a problem in the proof of deep normalisation (Corollary 4.4): because of the η-rule for Lift, the η-long normal form of any term $u : \mathsf{Lift}_t\ A$ should arguably have the form $\mathsf{lift}_{\bar{t}}\ u'$, with $\bar{t}$ the normal form of t; but the algorithmic η-equality rule for Lift does not include an assumption like "t is related to itself" that we could use in the deep normalisation proof, and it was unclear how to add such an assumption while maintaining all properties that we made use of (in particular stability under conversion and the lifting of algorithmic equality of atomic neutrals with types in WHNF to algorithmic equality of terms in WHNF). Because the lift constructor is not annotated, the type of a lift term cannot be inferred. An alternative approach might be to have level annotations on lift, but allow them to be unconstrained in the definition of η-long normal forms.

Algebraic type theory and gluing. We conclude this discussion by echoing a conclusion of Abel et al. [2017]: it remains unclear how to formally relate our extrinsically typed approach to the metatheory of type theory with more algebraic approaches [Sterling 2019; Kovács 2022; Altenkirch and Kaposi 2016, 2017]. In particular, an investigation into presheaf models [Coquand 2013] and the *gluing* construction [Kaposi et al. 2019] for type theories with dependent or first-class levels in the sense of Kovács [2022] might shed more light on the underlying mathematical structure of our proof of normalisation.

Data-Availability Statement

The formalisation discussed above is available online [Danielsson et al. 2025]. It is likely that the formalisation will be updated in the future. At the time of writing the latest version can be obtained from the following repository: https://github.com/graded-type-theory/graded-type-theory.

Acknowledgements

We would like to thank Andreas Abel, Jonathan Chan, András Kovács, Amélia Liao and Stephanie Weirich for useful conversations. We would also like to thank the anonymous reviewers.

Nils Anders Danielsson acknowledges financial support from Vetenskapsrådet (2023-04538).

References

Andreas Abel, Nils Anders Danielsson, and Oskar Eriksson. 2023. A Graded Modal Dependent Type Theory with a Universe and Erasure, Formalized. *Proceedings of the ACM on Programming Languages* 7, ICFP (2023), 35 pages. doi:10.1145/3607862

Andreas Abel, Joakim Öhman, and Andrea Vezzosi. 2017. Decidability of Conversion for Type Theory in Type Theory. *Proceedings of the ACM on Programming Languages* 2, POPL (2017), 29 pages. doi:10.1145/3158111

Arthur Adjedj, Meven Lennon-Bertrand, Kenji Maillard, Pierre-Marie Pédrot, and Loïc Pujet. 2024. Martin-Löf à la Coq. In *CPP '24, Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs*. 230–245. doi:10.1145/3636501.3636951

Guillaume Allais. 2019. Generic Level Polymorphic N-ary Functions. In *TyDe '19, Proceedings of the 4th ACM SIGPLAN International Workshop on Type-Driven Development*. 14–26. doi:10.1145/3331554.3342604

Guillaume Allais, Conor McBride, and Pierre Boutillier. 2013. New Equations for Neutral Terms: A Sound and Complete Decision Procedure, Formalized. In *DTP'13, Proceedings of the 2013 ACM SIGPLAN Workshop on Dependently-Typed Programming*. 13–24. doi:10.1145/2502409.2502411

Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. 1995. Categorical reconstruction of a reduction free normalization proof. In *Category Theory and Computer Science, 6th International Conference, CTCS '95*. 182–199. doi:10.1007/3-540-60164-3_27

Thorsten Altenkirch and Ambrus Kaposi. 2016. Type Theory in Type Theory using Quotient Inductive Types. In *POPL'16, Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. 18–29. doi:10.1145/2837614.2837638

Thorsten Altenkirch and Ambrus Kaposi. 2017. Normalisation by Evaluation for Type Theory, in Type Theory. *Logical Methods in Computer Science* 13, 4 (2017), 26 pages. doi:10.23638/LMCS-13(4:1)2017

Carlo Angiuli and Daniel Gratzer. 2025. Principles of Dependent Type Theory. Draft from 2025-07-19. https://www.danielgratzer.com/papers/type-theory-book.pdf

Marc Bezem, Thierry Coquand, Peter Dybjer, and Martín Escardó. 2023. Type Theory with Explicit Universe Polymorphism. In *28th International Conference on Types for Proofs and Programs, TYPES 2022*. 13:1–13:16. doi:10.4230/LIPIcs.TYPES.2022.13

Marc Bezem, Thierry Coquand, Peter Dybjer, and Martín Escardó. 2024. *Type Theory with Explicit Universe Polymorphism (revised and extended version)*. arXiv:2212.03284v5 [cs.LO] doi:10.48550/arXiv.2212.03284

Jonathan Chan and Stephanie Weirich. 2025. *Bounded First-Class Universe Levels in Dependent Type Theory*. arXiv:2502.20485v1 [cs.PL] doi:10.48550/arXiv.2502.20485

Thierry Coquand. 1986. An Analysis of Girard's Paradox. In *Proceedings of the Symposium on Logic in Computer Science (LICS '86)*. 227–236. https://www.cse.chalmers.se/~coquand/girard.pdf

Thierry Coquand. 2012. *Sheaf model of type theory*. http://www.cse.chalmers.se/~coquand/sheaf.pdf

Thierry Coquand. 2013. *Presheaf model of type theory*. http://www.cse.chalmers.se/~coquand/presheaf.pdf

Thierry Coquand. 2019. Canonicity and normalization for dependent type theory. *Theoretical Computer Science* 777 (2019), 184–191. doi:10.1016/j.tcs.2019.01.015

Judicaël Courant. 2002. Explicit Universes for the Calculus of Constructions. In *Theorem Proving in Higher Order Logics, 15th International Conference, TPHOLs 2002*. 115–130. doi:10.1007/3-540-45685-6_9

Nils Anders Danielsson, Naïm Camille Favier, and Ondřej Kubánek. 2025. *An Agda Formalisation of a Graded Modal Type Theory with First-Class Universe Levels and Erasure*. doi:10.5281/zenodo.17340136

N. G. de Bruijn. 1972. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae (Proceedings)* 75, 5 (1972), 381–392. doi:10.1016/1385-7258(72)90034-0

Peter Dybjer. 1996. Internal type theory. In *Types for Proofs and Programs, International Workshop, TYPES '95*. 120–134. doi:10.1007/3-540-61780-9_66

Peter Dybjer. 2000. A general formulation of simultaneous inductive-recursive definitions in type theory. *The Journal of Symbolic Logic* 65, 2 (2000), 525–549. doi:10.2307/2586554

Jean-Yves Girard. 1972. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur*. Thèse d'état. Université Paris VII. https://girard.perso.math.cnrs.fr/These.pdf

Robert Harper and Robert Pollack. 1991. Type checking with universes. *Theoretical Computer Science* 89, 1 (1991), 107–136. doi:10.1016/0304-3975(90)90108-T

Kuen-Bang Hou (Favonia), Carlo Angiuli, and Reed Mullanix. 2023. An Order-Theoretic Analysis of Universe Polymorphism. *Proceedings of the ACM on Programming Languages* 7, POPL (2023), 27 pages. doi:10.1145/3571250

Gérard Huet. 1988. Extending the Calculus of Constructions with Type:Type. Unpublished draft. https://pauillac.inria.fr/~huet/PUBLIC/typtyp.pdf

Junyoung Jang, Antoine Gaulin, Jason Z. S. Hu, and Brigitte Pientka. 2025. McTT: A Verified Kernel for a Proof Assistant. *Proceedings of the ACM on Programming Languages* 9, ICFP (2025), 32 pages. doi:10.1145/3747511

Ambrus Kaposi, Simon Huber, and Christian Sattler. 2019. Gluing for Type Theory. In *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019*. 25:1–25:19. doi:10.4230/LIPIcs.FSCD.2019.25

András Kovács. 2022. Generalized Universe Hierarchies and First-Class Universe Levels. In *30th EACSL Annual Conference on Computer Science Logic, CSL 2022*. 28:1–28:17. doi:10.4230/LIPIcs.CSL.2022.28

Meven Lennon-Bertrand. 2025. What Does It Take to Certify a Conversion Checker?. In *10th International Conference on Formal Structures for Computation and Deduction, FSCD 2025*. 27:1–27:23. doi:10.4230/LIPIcs.FSCD.2025.27

Pierre Letouzey. 2003. A New Extraction for Coq. In *Types for Proofs and Programs, International Workshop, TYPES 2002*. 200–219. doi:10.1007/3-540-39185-1_12

James McKinna. 2006. Why Dependent Types Matter. In *Conference Record of POPL® 2006: The 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages®*. 1–1. doi:10.1145/1111037.1111038

Richard Nathan Mishra-Linger. 2008. *Irrelevance, Polymorphism, and Erasure in Type Theory*. Ph. D. Dissertation. Portland State University. doi:10.15760/etd.2669

Jan M. Smith. 1988. The independence of Peano's fourth axiom from Martin-Löf's type theory without universes. *The Journal of Symbolic Logic* 53, 3 (1988), 840–845. doi:10.2307/2274575

Matthieu Sozeau, Yannick Forster, Meven Lennon-Bertrand, Jakob Nielsen, Nicolas Tabareau, and Théo Winterhalter. 2025. Correct and Complete Type Checking and Certified Erasure for Coq, in Coq. *J. ACM* 72, 1 (2025), 74 pages. doi:10.1145/3706056

Matthieu Sozeau and Nicolas Tabareau. 2014. Universe Polymorphism in Coq. In *Interactive Theorem Proving, 5th International Conference, ITP 2014*. 499–514. doi:10.1007/978-3-319-08970-6_32

Jonathan Sterling. 2019. *Algebraic Type Theory and Universe Hierarchies*. arXiv:1902.08848v1 [cs.LO] doi:10.48550/arXiv.1902.08848

W. W. Tait. 1967. Intensional interpretations of functionals of finite type I. *The Journal of Symbolic Logic* 32, 2 (1967), 198–212. doi:10.2307/2271658

The Agda Community. 2025. *The Agda standard library*. https://github.com/agda/agda-stdlib

The Agda Team. 2020. *Agda User Manual, Release 2.6.1*. https://agda.readthedocs.io/_/downloads/en/v2.6.1/pdf/

The Agda Team. 2021. *Agda User Manual, Release 2.6.2*. https://agda.readthedocs.io/_/downloads/en/v2.6.2/pdf/

The Agda Team. 2023. *Agda User Manual, Release 2.6.4*. https://agda.readthedocs.io/_/downloads/en/v2.6.4/pdf/

The Agda Team. 2025. *Agda User Manual, Release 2.8.0*. https://agda.readthedocs.io/_/downloads/en/v2.8.0/pdf/

The Lean Developers. 2025. The Lean Language Reference. https://lean-lang.org/doc/reference/4.25.0-rc2/

The Rocq Development Team. 2025. *The Rocq Prover Reference Manual, Release 9.0.0*. https://github.com/coq/coq/releases/download/V9.0.0/rocq-9.0.0-reference-manual.pdf

The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics* (first ed.). https://homotopytypetheory.org/book/

Received 2025-07-10; accepted 2025-11-06