Normalization by Evaluation for Non-cumulativity

SHENGYI JIANG, The University of Hong Kong, China
JASON Z. S. HU*, Amazon, USA
BRUNO C. D. S. OLIVEIRA, The University of Hong Kong, China

Normalization by evaluation (NbE) based on an untyped domain model is a convenient and powerful way to normalize terms to their $\beta\eta$ normal forms. It enables a concise technical setup and simplicity for mechanization. Nevertheless, to date, untyped NbE has only been studied for *cumulative* universe hierarchies, and its correctness proof critically relies on the cumulativity of the system. Therefore we are faced with the question: whether untyped NbE applies to a *non-cumulative* universe hierarchy? As such a universe hierarchy is also widely used by proof assistants like Agda and Lean, this question is also of practical significance.

Our work answers this question positively. One important property derived from non-cumulativity is *uniqueness*: every term has a unique type. In light of the uniqueness property, we work with a Martin-Löf type theory with explicit universe levels ascribed in the syntactic judgments. On the semantic side, universe levels are also explicitly managed, which leads to more complexity than the semantics with a cumulative universe hierarchy. We prove that the NbE algorithm is sound and complete, and confirm that NbE does work with non-cumulativity. Moreover, to capture common practice more faithfully, we also show that the explicit annotations of universe levels, though technically useful, are logically redundant: NbE remains applicable without these annotations. As such, we provide a mechanized foundation with NbE for non-cumulativity.

CCS Concepts: • **Theory of computation** → **Type theory**.

Additional Key Words and Phrases: type theory, dependent types, logical relations, normalization by evaluation

ACM Reference Format:
Shengyi Jiang, Jason Z. S. Hu, and Bruno C. d. S. Oliveira. 2025. Normalization by Evaluation for Non-cumulativity. *Proc. ACM Program. Lang.* 9, ICFP, Article 239 (August 2025), 35 pages. https://doi.org/10.1145/3747508

1 Introduction

Over the past decades, type theory has evolved into a mature field, providing a rigorous theoretical foundation for many popular proof assistants (e.g., Rocq [The Coq Development Team 2024], Agda [The Agda Team 2024], Lean [de Moura and Ullrich 2021; de Moura et al. 2015]). These proof assistants not only formalize cutting-edge mathematics [The Mathlib Community 2020] but also verify critical software [Leroy et al. 2016]. By the Curry–Howard correspondence, propositions are identified with types and proofs with programs. Hence, checking proofs essentially reduces to type-checking programs. Since dependent types allow types to embed arbitrarily complex computations, a practical type-checker must be coupled with a reliable evaluation procedure that always terminates. This property, known as *normalization*, guarantees that every well-typed program computes to a *normal form* and, on the meta-theoretic side, is closely related to logical consistency.

*Jason Z. S. Hu made his contribution during his Ph.D. study at McGill University, Canada.

Authors' Contact Information: Shengyi Jiang, syjiang@cs.hku.hk, The University of Hong Kong, Hong Kong, China; Jason Z. S. Hu, zhonhu@amazon.com, Amazon, Seattle, Washington, USA; Bruno C. d. S. Oliveira, bruno@cs.hku.hk, The University of Hong Kong, Hong Kong, China.

ACM 2475-1421/2025/8-ART239
https://doi.org/10.1145/3747508

A key design aspect in formulating type theories is how one organizes the hierarchy of universes [Palmgren 1998]. For instance, proof assistants such as Rocq adopt a *cumulative* universe hierarchy, whereby a type in a lower universe automatically inhabits any higher universe. Several normalization proofs in cumulative settings exist and can leverage this property. Nevertheless, proof assistants such as Agda and Lean, choose to implement *non-cumulative* universes. With non-cumulative universes, each well-formed type is assigned a unique universe level instead. An interesting property in many non-cumulativity settings is type uniqueness (up to syntactic equivalence), which simplifies certain aspects of constraint solving [Pujet and Tabareau 2023]. Nevertheless, (mechanized) normalization proofs for non-cumulative type theories are comparatively scarce and present different challenges, as several techniques employed in cumulative settings cannot be applied directly.

In this paper, we focus on a particular normalization proof style, normalization by evaluation (NbE) à la Abel [2013], which achieves $\beta\eta$-normalization and scales well to dependent type theories. NbE refers to a class of approaches to achieve normalization that generally consists of two steps. In the first step, well-formed programs are evaluated to domain values in a chosen computational domain. In the second step, normal forms are read from domain values in the computational domain back to the syntax. Properties of the computational domain are taken advantage of. The correctness of NbE is characterized by two critical theorems: the completeness theorem stating that two syntactically equivalent terms must have equal normal forms, and the soundness theorem stating that a well-formed term must be equivalent to its normal form computed by the algorithm. Thus, NbE does not need to prove confluence explicitly, resulting in a more concise technical setup and a shorter proof than a more traditional normalization proof based on a rewrite system and Tait's computability method [Tait 1967]. In addition, Abel [2013]'s NbE proof, based on an untyped domain, has one extra merit: it is easily implementable and mechanizable. Recently, Hu et al. [2023] presented a NbE proof in Agda of a modal dependent type theory in this style with significantly fewer lines of code than other similar works [Abel et al. 2018; Adjedj et al. 2024; Altenkirch and Kaposi 2016a, 2017; Pujet and Tabareau 2023, etc.]. Moreover, the NbE algorithm induces a simple equivalence checking algorithm: we first normalize two terms of the same type, and their equivalence is decided by the equality of their normal forms.

Existing proofs for untyped NbE, such as those by Abel [2013] and Hu et al. [2023], were developed in cumulative settings. Their techniques heavily depend on cumulativity. Abel [2013]; Gratzer et al. [2019]'s paper proofs include a key step of taking limits of universe levels to infinity due to cumulativity, so that their subsequent proofs are oblivious to universe levels. Hu et al. [2023] use existential quantifications to avoid limits and the exposition of universe levels in the semantics. Nonetheless, the proof still critically relies on the semantic cumulativity lemma of universes and a few lemmas to lift universes to a high enough level. Even though the NbE algorithm given by Abel and Hu et al. seems to adapt to both kinds of universe hierarchies naturally, it is not immediately clear how the dependencies of cumulativity in the proofs can be taken away.

In this paper, we develop a mechanized normalization proof for Martin-Löf Type Theory (MLTT) equipped with a full non-cumulative universe hierarchy. One immediate consequence of non-cumulativity is that all well-formed programs have unique types up to syntactic equivalence. Our work adapts the untyped NbE style of Abel [2013] and Hu et al. [2023] to a setting where precise universe levels must be tracked. Following Pujet and Tabareau [2023], we introduce explicit universe level annotations in the syntax. These additional annotations are also mirrored in the semantic domain, and allow us to design a PER model and logical relation that precisely track universe levels for the completeness and soundness proof to proceed. To bring our system closer to the common practice where no explicit universe levels are ascribed, we also show that these explicit annotations are logically redundant. This conclusion is achieved by proving an equivalence between two MLTTs with and without universe level annotations. We further show that there is an NbE algorithm that can be applied to the unascribed system directly, while maintaining its completeness and soundness. Our contributions are as follows:

- We provide a full mechanization in Agda of MLTT with a non-cumulative universe hierarchy. Our semantic models preserve precise universe level information, which is essential to proving the completeness and soundness of the NbE algorithm.
- We prove the uniqueness property and further show that universe level annotations are logically redundant by establishing an equivalence between two versions of non-cumulative MLTT—one with annotations and the other without. Moreover, we show that NbE directly applies to the system without annotations, thereby aligning the theory better with the common practice in proof assistants like Agda or Lean.
- We also include a mechanization of cumulative MLTT with a similar set of features and compare our mechanization of non-cumulative MLTT with it, highlighting how non-cumulativity complicates mechanization of meta-theoretic arguments, and discuss the trade-offs involved.

The remainder of the paper is organized as follows. Sec. 2 presents an overview of the problem setting, technical challenges and solutions. Sec. 3 formalizes the syntax of MLTT under non-cumulativity with explicit universe level annotations. In Sec. 4, we describe our NbE algorithm and, in Sec. 5, establish its theoretical properties. Sec. 6 discusses the additional complexities observed in the mechanization of the non-cumulative setting, and Sec. 7 shows that explicit universe annotations are logically redundant. We conclude with related work in Sec. 8 and final remarks in Sec. 9.

This paper includes hyperlinks to our online artifact to provide correspondence between the text and the mechanization to the readers. The mechanization, which assumes functional extensionality as its sole additional axiom, and extended version of this paper are also published at Zenodo [Jiang et al. 2025].

2 Overview

This section motivates the need for studying NbE in non-cumulative dependent type systems, and identifies the key challenges and ideas in our work.

2.1 Motivation: Cumulativity versus Non-cumulativity

In MLTT, any well-formed type must live on some universe level. There are two common structures of universe hierarchy: cumulative and non-cumulative. Cumulativity means that a type in a lower universe level is automatically a type in all the higher universes, while non-cumulativity refers to the lack of such a property. For example, the type of natural numbers N lives on level 0. With cumulativity, N also lives on all universe levels. In the works of Abel [2013] and Hu et al. [2023], cumulativity is achieved by extending the typing judgment with a rule allowing a type from a lower universe level to live on all higher levels.

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathsf{N} : \mathsf{Set}_0} \qquad\qquad \frac{\Gamma \vdash T : \mathsf{Set}_i}{\Gamma \vdash T : \mathsf{Set}_{1+i}}$$

Note that repeatedly applying the cumulativity rule lifts the universe level of N to an arbitrary level. Another (full-blown) way to support cumulativity is to introduce universe subtyping (c.f. Rocq and [Fridlender and Pagano 2013; Jang et al. 2025]). In contrast, non-cumulativity forces N to live on precisely universe level 0 by removing the cumulativity rule. Bringing types from lower universe levels to higher ones requires explicit wrapping. In Agda, this is provided by the `Lift` record. [1] For example, `Lift 2 N` lifts N from level 0 to 2, that can be abstracted using the following rules.

[1] https://github.com/agda/agda-stdlib/blob/4b3bb5419143666554f4fc8083e9353bdfbef5b9/src/Level.agda#L19

$$\frac{\Gamma \vdash T : \mathtt{Set}_i}{\Gamma \vdash \mathtt{Lift}_j\ T : \mathtt{Set}_{j+i}} \qquad \frac{\Gamma \vdash t : T}{\Gamma \vdash \mathtt{lift}_j\ t : \mathtt{Lift}_j\ T} \qquad \frac{\Gamma \vdash t : \mathtt{Lift}_j\ T}{\Gamma \vdash \mathtt{unlift}\ t : T}$$

Both styles of universe hierarchies have their own merits. Cumulativity is simply more convenient, because it saves the users from the explicit wrappings and unwrappings of `Lift`s. Cumulativity also matches many users' natural set-theoretic mindsets of universes. Non-cumulativity, on the contrary, has the uniqueness property, which says that a well-formed term must have a unique type (up to syntactic equivalence). The uniqueness property in turn relieves proof assistant implementers from the needs of universe subtyping and maintaining universe level constraint solvers, resulting in simpler implementations of proof assistants. In reality, different trade-offs are taken by different proof assistants. For example, Rocq is cumulative, while Agda and Lean adopt non-cumulativity.

Our work justifies the usage of untyped NbE in the non-cumulative universe setting. The technique is motivated by the recent work of Hu et al. [2023]. The main contribution of their work is to extend MLTT with a modality and a full cumulative universe hierarchy. Their mechanization employs NbE à la Abel [2013] and can be easily back-ported to MLTT to provide much more concise (in terms of lines of code) normalization proof than similar works [Abel et al. 2018; Adjedj et al. 2024; Altenkirch and Kaposi 2016a, 2017; Pujet and Tabareau 2023]. Nevertheless, the correctness proof of the NbE algorithm critically relies on cumulativity. Removing the dependency of cumulativity is not entirely obvious and is therefore the main challenge in our mechanization.

2.2 Proof by Keeping Track of Universe Levels

Since non-cumulativity has determined one universe level for each well-formed type, following Pujet and Tabareau [2023], we explicitly ascribe types and the typing and equivalence judgments with universe levels, so that syntactically, universe levels are constantly kept track of. For example, Π types $\Pi(x : S^{\boxed{i}}).T^{\boxed{j}}$ and the typing judgment $\Gamma \vdash t :^{\boxed{i}} T$ have universe levels annotated.

The uniqueness property has a further and more complicated implication on the semantics. With cumulativity, the semantics of a type constructor only needs to be based on smaller types on the *same* level. For example, the semantics of a Π type on level i only depends on the semantics of the input and output types both on level i. However, in a non-cumulative hierarchy, the semantics must maintain a precise account of universe levels to reflect the uniqueness property. For a Π type on level i, its semantics now must depend on the input and output types on distinct levels j and k, respectively, and $i = \max(j, k)$ must hold. This precision can be seen from the typing rules:

cumulative
$$\frac{\Gamma \vdash S : \mathtt{Set}_i \qquad \Gamma, x : S \vdash T : \mathtt{Set}_i}{\Gamma \vdash \Pi(x : S).T : \mathtt{Set}_i}$$

non-cumulative
$$\frac{\Gamma \vdash S :^{\boxed{1+j}} \mathtt{Set}_j \qquad \Gamma, x : S \vdash T :^{\boxed{1+k}} \mathtt{Set}_k}{\Gamma \vdash \Pi(x : S^{\boxed{j}}).T^{\boxed{k}} :^{\boxed{1+\max(j,k)}} \mathtt{Set}_{\max(j,k)}}$$

This precision in universe levels poses an increasing complication in the **P**artial **E**quivalence **R**elation (PER) model, which is used to establish the completeness theorem, compared to the cumulative hierarchy. The definition of the Kripke model for the soundness theorem becomes even more complex, because the Kripke model is defined on top of the PER model. Our mechanization demonstrates how exactly these models are defined in Agda to eventually prove both completeness and soundness theorems of NbE.

Once completeness and soundness of NbE are established, we can derive several consequences, including exactness of universe levels, injectivity of type constructors, consistency and canonicity. One extra syntactic consequence of the non-cumulative system is the uniqueness property. Although it is deemed obvious, its proof can only be established at this late stage. Complexity in the proofs primarily arises in the elimination cases, which require the injectivity of type constructors, making this property a consequence of the NbE proof as well.

2.3 Approaching Practical Syntax: Removing Explicit Universe Levels

Explicit annotations of universe levels are important to complete the NbE proof, but in reality, they are hardly a common practice. Empowered by the semantic models and the uniqueness property, we further show that these annotations are indeed logically redundant. In other words, all highlighted universe levels (such as the highlighted levels in the earlier typing rule for non-cumulative Π types) are dropped to form an unascribed MLTT and these two versions of MLTT's are equivalent. To our surprise, formally establishing this conclusion is not very straightforward. The direction going from the unascribed MLTT to the ascribed MLTT essentially re-annotates universe levels back to types and judgments. The proof requires us to prove not only the existence of re-annotations, but also that they these re-annotations always lead to equivalent terms. The proof depends on the uniqueness property established earlier and requires a stronger conclusion to perform induction, leading to a significantly larger proof than what we originally anticipated. When the syntactic equivalence between two systems is established, it allows us to transport properties proved in the ascribed system to the unascribed system, e.g., consistency. It further allows us to develop an NbE algorithm for the unascribed system directly and justify its soundness and completeness. Despite the complication in the proof, the unascribed NbE algorithm itself is completely irrelevant to universe-level annotations and remains simple and efficient.

3 Syntactic Definition of MLTT with a Non-Cumulative Universe Hierarchy

In this section, we define a version of MLTT with a non-cumulative universe hierarchy. The feature set is standard and is a representative subset of MLTT. Universe levels are explicitly annotated in this syntax and judgments.

3.1 Syntax

De Bruijn Indices	$n \in \mathbb{N}$	Variable Names x, y, z	Universe Levels $i, j, k \in \mathbb{N}$
Terms, Types	r, s, t, R, S, T	$::=$	$x_n \mid \lambda(x : S^{i}).t \mid t\ s \mid \mathtt{0} \mid \mathtt{suc}\ t \mid \mathtt{rec}(x.T^{i})\ r\ (x, y.s)\ t \mid$ $\mathtt{lift}_j\ t \mid \mathtt{unlift}\ t \mid t[\sigma] \mid$ $\mathtt{Set}_i \mid \Pi(x : S^{i}).T^{j} \mid \mathtt{N} \mid \mathtt{Lift}_j\ T^{i}$
Substitutions	σ, τ, γ	$::=$	$\mathtt{Id} \mid \uparrow \mid \sigma, t : T^{i}/x_0 \mid \sigma \circ \tau$
Normal Forms	v, w, V, W	$::=$	$u \mid \lambda(x : W^{i}).w \mid \mathtt{0} \mid \mathtt{suc}\ v \mid \mathtt{lift}_j\ v \mid \mathtt{Set}_i \mid$ $\Pi(x : V^{i}).W^{j} \mid \mathtt{N} \mid \mathtt{Lift}_j\ V^{i}$
Neutral Forms	u	$::=$	$x_n \mid u\ v \mid \mathtt{rec}(x.W^{i})\ v_z\ (x, y.v_s)\ u \mid \mathtt{unlift}\ u$
Contexts	Γ, Δ, Ψ	$::=$	$\cdot \mid \Gamma, x : T^{i}$

In our mechanization, we use de Bruijn indices to represent variables. In the text, for clarity, we incorporate abstract names x for readability.[2] When de Bruijn indices are significant, they are marked as subscripts of variables x_n. For example, x_0 is the topmost variable in the context, and $S \vdash 0$ (where 0 is a de Bruijn index) is now represented as $x : S \vdash x_0$. Otherwise, we often omit the subscripts to reduce noise. To avoid confusion, natural numbers in the object language are denoted with monospaced fonts ($\mathtt{N} = \mathtt{0}, \mathtt{suc}\ t$), while natural numbers $\mathbb{N}$ in the meta-language (for variable position, universe level, etc.) are denoted with normal fonts $(0, 1, i, j, n)$.

[2]We cannot fully adopt named representations due to the problem that our formulation of explicit substitution is defined for de Bruijn indices only. Interested readers can refer to our mechanization for a full de-Bruijn-index representation.

Types and terms. The syntax of types and terms is unified as often is the case for dependently typed calculi. The highlighted superscripts ^{i} in the syntax indicate exact positions where explicit universe level annotations are added. Types in our system include universe types $\texttt{Set}_i$, Π-types ($\Pi(x : S^{i}).T^{j}$), whose introduction form is λ-abstraction ($\lambda(x : S^{i}).t$) and elimination form is function application ($t\ s$). The "." in the syntax denotes a binding structure, where variables appear before it are bound in the term that appears after it. For example, in $\Pi(x : S^{i}).T^{j}$, x in bound in T. We also introduce a type of natural numbers $\texttt{N}$ whose introduction forms are ($\texttt{0}$ and $\texttt{suc}\ t$) and elimination form is the recursor $\texttt{rec}(x.T^{i})\ r\ (x, y.s)\ t$. In the recursor, we do recursion on t, which is the natural number to be recursed on. If it computes to $\texttt{0}$, then the recursor computes to the base case, represented by r. If t computes to $\texttt{suc}\ t'$ for some t', then the recursor hits the step case, represented by s. The step case s has two open variables, where x is for the predecessor, i.e., t' in this case, and y is replaced by the recursive call. Finally, the overall type of the recursion is computed by $x.T^{i}$ by replacing x with t. This type is often referred to as a *motive* [McBride 2000]. To provide a way to manually adjust the universe of types, we also have $\texttt{Lift}_j\ T^{i}$, whose introduction form is $\texttt{lift}_j\ t$ and elimination form is $\texttt{unlift}\ t$. Last but not least, our type theory is given as an explicit substitution calculus, where substitutions are delayed and explicitly recorded. Consequently, a dedicated syntax $t[\sigma]$ for applying a substitution σ to term t is needed.

Normal forms and neutral forms of this system are mutually defined and standard. Neutral forms include variables and all elimination forms where the scrutinee is a neutral form and other components are normal forms (e.g., $u\ v$). Normal forms include all introduction forms of each type and all types themselves, whose component are also normal forms (e.g., $\lambda(x : W^{i}).w$ and $\Pi(x : W^{i}).V^{j}$). Neutral forms are also normal forms.

Explicit substitutions. Explicit substitutions are a conventional formulation to study NbE on type theories [Abel 2013; Altenkirch and Kaposi 2016b; Hu and Pientka 2023; Wieczorek and Biernacki 2018], with benefits to be discussed in Sec. 4.2. They are also considered closer to real implementations, compared with substitution-as-operations. The syntax of our explicit substitutions follows that of Abadi et al. [1991] which consists of 4 cases. Identity substitutions $\texttt{Id}$ do nothing. Weakening substitutions $\uparrow$ weaken de Bruijn indices of all free variables by one. They extend the context with an variable on top. Substitution extensions $(\sigma, t : T^{i}/x_0)$ substitute the topmost variable (as indicated by x_0) by t and applies σ to the other open variables. Since we adopt Church-style functions, context extensions also include additional type annotations, similar to Abadi et al. [1991]. Substitution compositions $\sigma \circ \tau$ compose two substitutions. It first applies σ and then applies τ to a term. For brevity, we also introduce notations for two commonly used composed substitutions: $[s : S^{i}/x_0]$ is short for $[\texttt{Id}, s : S^{i}/x_0]$, which substitutes s for the topmost variable, and down-shifts de Bruijn indices of other variables by 1. The weakening of substitution $\texttt{q}(T^{i}, \sigma)$ weakens the substitution σ by extending the domain and co-domain contexts with type $(T[\sigma])^{i}$ and T^{i}. It is concretely expanded to $(\sigma \circ \uparrow), (x_0 : T^{i}/x_0)$.

3.2 Typing and Equivalence

The typing and equivalence rules are defined by six mutually defined judgments in total: context well-formedness $\vdash \Gamma$, typing (term well-formedness) $\Gamma \vdash t :^{i} T$ (Fig. 1), substitution well-formedness (substitution typing) $\Gamma \vdash \sigma : \Delta$ (Fig. 2), context equivalence $\vdash \Gamma \equiv \Delta$, term equivalence $\Gamma \vdash s \equiv t :^{i} T$ (Fig. 1) and substitution equivalence $\Gamma \vdash \sigma \equiv \tau : \Psi$ (Fig. 2). The mutual definitions of context equivalence, substitution well-formedness and substitution equivalence are due to our choice of explicit substitution, otherwise substitution well-formedness and substitution equivalence are no longer needed and context equivalence can be defined independently later. This style of definitions

$\boxed{\Gamma \vdash t :^{i} T}$ t has type T at level i under context Γ

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathsf{N} :^{1} \mathsf{Set}_0} \qquad \frac{\vdash \Gamma}{\Gamma \vdash \mathsf{Set}_i :^{2+i} \mathsf{Set}_{1+i}} \qquad \frac{\Gamma \vdash T :^{1+i} \mathsf{Set}_i}{\Gamma \vdash \mathsf{Lift}_j\, T^{i} :^{1+j+i} \mathsf{Set}_{j+i}} \qquad \frac{\vdash \Gamma \qquad x : T^{i} \in \Gamma}{\Gamma \vdash x :^{i} T}$$

$$\frac{\Gamma \vdash S :^{1+i} \mathsf{Set}_i \qquad \Gamma, x : S^{i} \vdash T :^{1+j} \mathsf{Set}_j}{\Gamma \vdash \Pi(x : S^{i}).T^{j} :^{1+\max(j,k)} \mathsf{Set}_{\max(i,j)}} \qquad \frac{\Gamma \vdash S :^{1+i} \mathsf{Set}_i \qquad \Gamma, x : S^{i} \vdash t :^{j} T}{\Gamma \vdash \lambda(x : S^{i}).t :^{\max(i,j)} \Pi(x : S^{i}).T^{j}}$$

$$\frac{\Gamma \vdash S :^{1+i} \mathsf{Set}_i \qquad \Gamma, x : S^{i} \vdash T :^{1+j} \mathsf{Set}_j \qquad \Gamma \vdash s :^{\max(i,j)} \Pi(x : S^{i}).T^{j} \qquad \Gamma \vdash t :^{i} S}{\Gamma \vdash s\, t :^{j} T[s : S^{i}/x_0]}$$

$$\frac{\Gamma \vdash t :^{i} T}{\Gamma \vdash \mathsf{lift}_j\, t :^{j+i} \mathsf{Lift}_j\, T^{i}} \qquad \frac{\Gamma \vdash T :^{1+i} \mathsf{Set}_i \qquad \Gamma \vdash t :^{j+i} \mathsf{Lift}_j\, T^{i}}{\Gamma \vdash \mathsf{unlift}\, t :^{i} T^{i}} \qquad \frac{\vdash \Gamma}{\Gamma \vdash \mathsf{0} :^{0} \mathsf{N}}$$

$$\frac{\Gamma \vdash t :^{0} \mathsf{N}}{\Gamma \vdash \mathsf{suc}\, t :^{0} \mathsf{N}} \qquad \frac{\begin{array}{c}\Gamma, z : \mathsf{N}^{0} \vdash T :^{1+i} \mathsf{Set}_i \qquad \Gamma \vdash r :^{i} T[\mathsf{0} : \mathsf{N}^{0}/z] \\ \Gamma, x : \mathsf{N}^{0}, y : (T[x_1 : \mathsf{N}^{0}/z])^{i} \vdash s :^{i} T[(\uparrow\circ\uparrow), \mathsf{suc}\, x_1 : \mathsf{N}^{0}/z] \qquad \Gamma \vdash t :^{0} \mathsf{N}\end{array}}{\Gamma \vdash \mathsf{rec}(z.T^{i})\ r\ (x, y.s)\ t :^{i} T[t : \mathsf{N}^{0}/z_0]}$$

$$\frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \vdash t :^{i} T}{\Gamma \vdash t[\sigma] :^{i} T[\sigma]} \qquad \frac{\Gamma \vdash t :^{i} S \qquad \Gamma \vdash S \equiv T :^{1+i} \mathsf{Set}_i}{\Gamma \vdash t :^{i} T}$$

$\boxed{\Gamma \vdash s \equiv t :^{i} T}$ t and s of type T are equivalent at level i under context Γ

$$\frac{\begin{array}{c}\Gamma \vdash S :^{1+i} \mathsf{Set}_i \\ \Gamma, x : S^{i} \vdash T :^{1+j} \mathsf{Set}_j \qquad \Gamma, x : S^{i} \vdash t :^{j} T \quad \Gamma \vdash s :^{i} S\end{array}}{\Gamma \vdash (\lambda(x : S^{i}).t)\ s \equiv t[s : S^{i}/x] :^{j} T[s : S^{i}/x]} \qquad \frac{\Gamma \vdash t :^{i} T}{\Gamma \vdash \mathsf{unlift}\ (\mathsf{lift}_j\ t) \equiv t :^{i} T}$$

$$\frac{\begin{array}{c}\Gamma \vdash S :^{1+i} \mathsf{Set}_i \qquad \Gamma, x : S^{i} \vdash T :^{1+j} \mathsf{Set}_j \\ \Gamma \vdash t :^{\max(i,j)} \Pi(x : S^{i}).T^{j}\end{array}}{\Gamma \vdash t \equiv \lambda(x : S^{i}).\big((t[\uparrow])\, x\big) :^{\max(i,j)} \Pi(x : S^{i}).T^{j}} \qquad \frac{\Gamma \vdash T :^{1+i} \mathsf{Set}_i \qquad \Gamma \vdash t :^{j+i} \mathsf{Lift}_j\, T^{i}}{\Gamma \vdash t \equiv \mathsf{lift}_j\ (\mathsf{unlift}\ t) :^{j+i} \mathsf{Lift}_j\, T^{i}}$$

$$\frac{\begin{array}{c}\Gamma, z : \mathsf{N}^{0} \vdash T :^{1+i} \mathsf{Set}_i \\ \Gamma \vdash r :^{i} T[\mathsf{0} : \mathsf{N}^{0}/z] \qquad \Gamma, x : \mathsf{N}^{0}, y : (T[x_1 : \mathsf{N}^{0}/z])^{i} \vdash s :^{i} T[(\uparrow \circ \uparrow), \mathsf{suc}\, x_1 : \mathsf{N}^{0}/z]\end{array}}{\Gamma \vdash \mathsf{rec}(z.T^{i})\ r\ (x, y.s)\ \mathsf{0} \equiv r :^{i} T[\mathsf{0} : \mathsf{N}^{0}/z]}$$

$$\frac{\begin{array}{c}\Gamma, z : \mathsf{N}^{0} \vdash T :^{1+i} \mathsf{Set}_i \\ \Gamma \vdash r :^{i} T[\mathsf{0} : \mathsf{N}^{0}/z] \qquad \Gamma, x : \mathsf{N}^{0}, y : (T[x_1 : \mathsf{N}^{0}/z])^{i} \vdash s :^{i} T[(\uparrow \circ \uparrow), \mathsf{suc}\, x_1 : \mathsf{N}^{0}/z] \qquad \Gamma \vdash t :^{0} \mathsf{N}\end{array}}{\Gamma \vdash \mathsf{rec}(z.T^{i})\ r\ (x, y.s)\ (\mathsf{suc}\ t) \equiv s[t : \mathsf{N}^{0}/x, (\mathsf{rec}(z.T^{i})\ r\ (x, y.s)\ t)^{i}/y] :^{i} T[\mathsf{suc}\ t : \mathsf{N}^{0}/z]}$$

Fig. 1. Typing and equivalence rules of terms (excerpt).

follows Abel [2013] closely. For conciseness, we only present the important rules in the paper, and defer the full set of the rules to Appendix A.

Well-formedness of terms and substitutions. Fig. 1 presents the typing judgments for terms and types. As explained in Sec. 2.2, we explicitly ascribe the levels of types. In other words, given $\Gamma \vdash t :^{i} T$, the universe level of T is i. Note that the typing judgment does not include the cumulativity rule in Sec. 2.1, and therefore the system is non-cumulative. In some rules, there are redundant premises. For instance, in the λ rule, the well-formedness of S is implied by the well-formedness of t. We include these redundant premises in order to prove the presupposition lemma (Theorem 3.2)

$\boxed{\Gamma \vdash \sigma : \Delta}$ σ has codomain context Δ under Γ

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathtt{Id} : \Gamma} \qquad \frac{\vdash \Gamma, T^{i}}{\Gamma, T^{i} \vdash \uparrow : \Gamma} \qquad \frac{\Gamma \vdash \sigma : \Delta \quad \Delta \vdash \tau : \Psi}{\Gamma \vdash \tau \circ \sigma : \Psi}$$

$$\frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \vdash T :^{1+i} \mathtt{Set}_i \qquad \Gamma \vdash t :^{i} T[\sigma]}{\Gamma \vdash \sigma, t : T^{i}/x_0 : (\Delta, x : T^{i})} \qquad \frac{\Gamma \vdash \sigma : \Delta \quad \vdash \Delta \equiv \Psi}{\Gamma \vdash \sigma : \Psi}$$

$\boxed{\Gamma \vdash \sigma \equiv \tau : \Delta}$ σ and τ having codomain context Δ are equivalent substitutions under Γ

$$\frac{\Gamma \vdash \sigma : \Delta \quad \Delta \vdash T :^{1+i} \mathtt{Set}_i \quad \Gamma \vdash t :^{i} T^{\sigma}}{\Gamma \vdash \uparrow \circ (\sigma, t : T^{i}/x_0) \equiv \sigma : \Delta} \qquad \frac{\Gamma \vdash \sigma : \Gamma, T^{i}}{\Gamma \vdash \sigma \equiv (\uparrow \circ \sigma), (x_0[\sigma] : T^{i}/x_0) : \Gamma, T^{i}}$$

$$\frac{\Gamma \vdash \tau : \Delta \quad \Delta \vdash \sigma : \Psi \quad \Psi \vdash T :^{1+i} \mathtt{Set}_i \quad \Delta \vdash t :^{i} T[\sigma]}{\Gamma \vdash (\sigma, t : T^{i}/x_0) \circ \tau \equiv (\sigma \circ \tau),\ (t[\tau] : T^{i}/x_0) : \Psi, T^{i}}$$

Fig. 2. Typing and equivalence rules of substitutions (excerpt).

purely syntactically, following Harper and Pfenning [2005]. The typing rules for explicit substitutions are shown at the top of Fig. 2. Due to explicit substitutions, we must include a context conversion rule at the end to swap the result context to an equivalent one.

Syntactic equivalence. The syntactic equivalence relation describes the equivalence relation between terms. The type theory regards two equivalent terms "the same" and they cannot be distinguished within the system. Syntactic equivalence rules usually consist of three kinds of rules: PER rules that include symmetry and transitivity, congruence rules that propagate equivalence deeper in the syntactic structures, and computation rules that describe how computation is performed. With explicit substitutions, there are also rules to describe how substitutions interact with terms. For brevity, the bottom of Fig. 1 highlights the β and η computational rules of syntactic equivalence. Other rules are presented in Appendix A. Note that the `Lift` type supports η expansion as well, mimicking its behavior in Agda. Finally, the bottom of Fig. 2 defines the syntactic equivalence between substitutions. They are usually properties if substitutions are defined as operations, but we need to list them as rules due to explicit substitutions.

Syntactic properties. Two important syntactic properties of this system are presupposition and equivalent context theorem. The context equivalence theorem states that every judgment still holds when we change its input context to an equivalent context. Presupposition states that each judgment implies the well-formedness of every component.

Theorem 3.1 (Context Conversion). *Given* $\vdash \Gamma \equiv \Delta$,

- *If* $\Gamma \vdash t :^{i} T$, *then* $\Delta \vdash t :^{i} T$;
- *If* $\Gamma \vdash t \equiv s :^{i} T$, *then* $\Delta \vdash t \equiv s :^{i} T$;
- *If* $\Gamma \vdash \sigma : \Psi$, *then* $\Delta \vdash \sigma : \Psi$;
- *If* $\Gamma \vdash \sigma \equiv \tau : \Psi$, *then* $\Delta \vdash \sigma \equiv \tau : \Psi$.

Theorem 3.2 (Presupposition).

- *If* $\vdash \Gamma \equiv \Delta$, *then* $\vdash \Gamma$ *and* $\vdash \Delta$;
- *If* $\Gamma \vdash t :^{i} T$, *then* $\vdash \Gamma$ *and* $\Gamma \vdash T :^{1+i} \mathtt{Set}_i$;
- *If* $\Gamma \vdash s \equiv t :^{i} T$, *then* $\vdash \Gamma$ *and* $\Gamma \vdash s :^{i} T$ *and* $\Gamma \vdash t :^{i} T$ *and* $\Gamma \vdash T :^{1+i} \mathtt{Set}_i$;
- *If* $\Gamma \vdash \sigma : \Delta$, *then* $\vdash \Gamma$ *and* $\vdash \Delta$;
- *If* $\Gamma \vdash \sigma \equiv \tau : \Delta$, *then* $\vdash \Gamma$ *and* $\Gamma \vdash \sigma : \Delta$ *and* $\Gamma \vdash \tau : \Delta$ *and* $\vdash \Delta$.

$$\begin{tikzcd}
\text{Semantics } (\beta) & & \mathrm{D} \arrow[d, "\downarrow^{B}"] & & \\
\text{Semantics } (\beta\eta) & & \mathrm{D}^{\mathrm{nf}} \arrow[d, "\mathrm{R}^{\mathrm{nf}}"] & \mathrm{D}^{\mathrm{ne}} \arrow[ul, "\uparrow^{A}"'] \arrow[d, "\mathrm{R}^{\mathrm{ne}}"'] & \mathrm{Level} \arrow[l, "\mathrm{x}"'] \\
\text{Syntax} & \mathrm{Exp} \arrow[uur, bend left, "{\llbracket _ \rrbracket}"] & \mathrm{Nf} \arrow[l, "\supseteq"'] & \mathrm{Ne} \arrow[l, "\supseteq"'] & \mathrm{Index} \arrow[l, "x"'] \arrow[u, "|\Gamma|-"']
\end{tikzcd}$$

Fig. 3. Diagram of NbE à la Abel in a locally nameless style.

These properties are proved by induction directly. With these two properties, redundant premises in the syntactic judgments can be derived from other premises. Hence it becomes easier to construct a derivation tree.

4 Normalization by Evaluation

In this section, we introduce an NbE algorithm with ascribed rules for the syntax presented in the previous section. The general definitions follow Abel et al. [2017].

Fig. 3, adapted from Abel et al. [2017], illustrates the general procedure of such an NbE algorithm. Given a well-typed term $\Gamma \vdash t : T$, the whole NbE algorithm then works in a 3-stage manner: (1) interpreting each free de Bruijn indices x_i of type T_i in Γ to reflected ($\uparrow^A$ in Fig. 3) de Bruijn levels $\mathrm{x}_{|\Gamma|-1-i}$ ($|\Gamma|-$ in Fig. 3) to form an environment ρ; (2) evaluating ($\llbracket \_ \rrbracket$ in Fig. 3) t in ρ and reifying ($\downarrow^B$ in Fig. 3) the result to a normal semantic value; (3) reading back (R^{nf} in Fig. 3) the normal semantic value into a syntactic normal form. This is more refined than the usual 2-stage description of NbE by putting a dedicated emphasis on evaluating free variables. The exact definition of evaluation, readback, and context evaluation will be developed in this section. Readers are welcome to refer back to this diagram to alternate between high-level description and exact definitions.

4.1 Semantic Values

$$\begin{array}{llcl}
\text{Environment} & \rho, \phi, \theta \in \mathbb{N} \rightarrow \mathrm{D} & & \\
\text{Semantic Values } (\mathrm{D}) & a, b, c, f, & ::= & (\!|\boldsymbol{\lambda} x.t|\!)_\rho \mid \mathbf{0} \mid \mathbf{suc}\ d \mid \mathbf{lift}_i\ a \mid \mathbf{Set}_i \mid \\
 & A, B, F & & (\!|\mathbf{\Pi}\ A^{\boldsymbol{i}}\ (x.T^{\boldsymbol{j}})|\!)_\rho \mid \mathbf{N} \mid \mathbf{Lift}_j\ A^{\boldsymbol{i}} \mid \uparrow^{\boldsymbol{i}}_A e \\
\text{Neutral Semantic Values } (\mathrm{D}^{\mathrm{ne}}) & e, E & ::= & \mathrm{x}_k \mid e\ d \mid (\!|\mathbf{rec}\ (z.T^{\boldsymbol{i}})\ a\ (x, y.s)\ e|\!)_\rho \mid \mathbf{unlift}\ e \\
\text{Normal Semantic Values } (\mathrm{D}^{\mathrm{nf}}) & d, D & ::= & \downarrow^{\boldsymbol{i}}_A a
\end{array}$$

Most semantic values correspond directly to the types and introduction form of in the syntax, including $\mathbf{0}$, $\mathbf{suc}\ d$, $\mathbf{lift}_i\ a$, $\mathbf{Set}_i$, $\mathbf{N}$ and $\mathbf{Lift}_j\ a^{\boldsymbol{i}}$. Functional values ($(\!|\boldsymbol{\lambda} x.t|\!)_\rho$, $(\!|\mathbf{\Pi}\ A^{\boldsymbol{i}}\ (x.T^{\boldsymbol{j}})|\!)_\rho$) are formulated using closures [Landin 1964]. Reflected neutral values $\uparrow^{\boldsymbol{i}}_A e$ are also values. Neutral semantic values include variables and elimination forms blocked by neutral semantic values ($e\ d$, $\mathbf{unlift}\ e$, $(\!|\mathbf{rec}\ (z.T^{\boldsymbol{i}})\ a\ (x, y.s)\ e|\!)_\rho$). $(\!|\mathbf{rec}\ (z.T^{\boldsymbol{i}})\ a\ (x, y.s)\ e|\!)_\rho$ is also equipped with a closure. Closures capture all free variables used by inner terms, such that later evaluation of the body inside the binders only depends on this exact ρ. Variables x_k are represented by de Bruijn levels k. De Bruijn levels can be viewed as an absolute name (cf. locally nameless [Charguéraud 2012]) assigned to the variable that is invariant to context extensions. Normal semantic values only include reified values $\downarrow^{\boldsymbol{i}}_A a$. Reflection ($\uparrow^{\boldsymbol{i}}_A$) and reification ($\downarrow^{\boldsymbol{i}}_A$) are markers, indicating that the actual η-expansion still has to be performed later. Our semantic values also carry the universe level information for two reasons: (1) these universe levels provide important information to develop a precise PER model and logical relation on semantic values (which will be discussed in Sec. 5); (2) our normal

$\boxed{[\![t]\!]_\rho \searrow a}$ t evaluates to a under ρ

$$\frac{}{[\![\mathsf{N}]\!]_\rho \searrow \mathbf{N}} \qquad \frac{[\![S]\!]_\rho \searrow A}{[\![\Pi(x : S^i).T^j]\!]_\rho \searrow (\!|\mathbf{\Pi}\, A^i\ (x.T^j)|\!)_\rho} \qquad \frac{[\![T]\!]_\rho \searrow A}{[\![\mathtt{Lift}_j\ T^i]\!]_\rho \searrow \mathbf{Lift}_j\ A^i}$$

$$\frac{}{[\![\mathtt{Set}_i]\!]_\rho \searrow \mathbf{Set}_i} \qquad \frac{[\![\sigma]\!]_s(\rho) \searrow \phi \qquad [\![t]\!]_\phi \searrow a}{[\![t[\sigma]]\!]_\rho \searrow a} \qquad \frac{}{[\![x_n]\!]_\rho \searrow \rho(n)} \qquad \frac{}{[\![\mathtt{0}]\!]_\rho \searrow \mathbf{0}}$$

$$\frac{[\![t]\!]_\rho \searrow a}{[\![\mathtt{suc}\ t]\!]_\rho \searrow \mathbf{suc}\ a} \qquad \frac{[\![r]\!]_\rho \searrow a \qquad [\![t]\!]_\rho \searrow b \qquad \mathrm{rec}\cdot(z.T^i, a, (x, y.s), b, \rho) \searrow c}{[\![\mathtt{rec}\ (z.T^i)\ r\ (x, y.s)\ t]\!]_\rho \searrow c}$$

$$\frac{}{[\![\lambda(x : S^i).t]\!]_\rho \searrow (\!|\boldsymbol{\lambda} x.t|\!)_\rho} \qquad \frac{[\![s]\!]_\rho \searrow f \qquad [\![t]\!]_\rho \searrow a \qquad f \cdot a \searrow b}{[\![s\ t]\!]_\rho \searrow b} \qquad \frac{[\![t]\!]_\rho \searrow a}{[\![\mathtt{lift}_i\ t]\!]_\rho \searrow \mathbf{lift}_i\ a}$$

$$\frac{[\![t]\!]_\rho \searrow a \qquad \mathrm{unlift}\cdot\ a \searrow b}{[\![\mathtt{unlift}\ t]\!]_\rho \searrow b}$$

$\boxed{[\![\sigma]\!]_s(\rho) \searrow \phi}$ σ evaluates to ϕ under ρ

$$\frac{}{[\![\mathtt{Id}]\!]_s(\rho) \searrow \rho} \qquad \frac{}{[\![\uparrow]\!]_s(\rho) \searrow \mathtt{drop}\ \rho} \qquad \frac{[\![\sigma]\!]_s(\rho) \searrow \phi \quad [\![t]\!]_\phi \searrow d}{[\![\sigma, t : T^i/x_0]\!]_s(\rho) \searrow \phi; d} \qquad \frac{[\![\tau]\!]_s(\rho) \searrow \phi \quad [\![\sigma]\!]_s(\phi) \searrow \theta}{[\![\sigma \circ \tau]\!]_s(\rho) \searrow \theta}$$

Fig. 4. Relational definitions of evaluation functions.

forms, as a subset of the expressions, need to have universe level annotations, so carrying them in values simplify the definition of readback. This semantic domain is untyped, as it does not preclude construction of non-nonsensical values like **suc** $\mathbf{Set}_i$.

Environments $\rho : \mathbb{N} \to \mathrm{D}$ are mappings from natural numbers (de Bruijn indices) to semantic values. Consequently, environment extension $(\rho; d)$ creates a new mapping where $(\rho; d)(0) = d$ and $(\rho; d)(1 + i) = \rho(i)$, and environment drop creates another new mapping $(\mathtt{drop}\ \rho)(n) = \rho(1 + n)$.

4.2 Evaluation and Readback Functions

The rules of evaluation are shown in Fig. 4 and 5 and the rules of readback are shown in Fig. 6. Evaluation and readback are all partial functions, and are presented as relations from inputs to outputs to stay close to our mechanization. It is easy to show that these relations are right-unique, and hence are partial functions. We will also sometimes use them as functions, where we implicitly assume an existential quantifier for the results.

Evaluation consists of five mutually defined relations: term evaluation $[\![t]\!]_\rho \searrow a$, substitution evaluation $[\![\sigma]\!]_s(\rho) \searrow \phi$, application $f \cdot a \searrow b$, recursion-application $\mathrm{rec}\cdot(z.T^i, a, (x, y.s), b, \rho) \searrow c$ and unlift-application $\mathrm{unlift}\cdot\ a \searrow b$. Most rules in term and substitution evaluation are straightforward. λ-abstractions and Π are evaluated to closures and variables are evaluated to the value retrieved from environments. The latter three relations are helper relations that handle elimination forms in the term evaluation. Definitions of these helper functions have a similar structure: one case when the scrutinee is of each introduction form; and one case when the scrutinee is a reflected neutral value. η-expansion happens in the last case of application and unlift-application and is interleaved with β-reduction. With explicit substitutions, the evaluation rule of $t[\sigma]$ makes the whole evaluation more aligned for terms before and after the β-reduction, which simplifies the justification of β-equality in the semantics domain [Abel 2013].

$\boxed{f \cdot a \searrow b}$ apply f to a is b

$$\frac{\llbracket t \rrbracket_{\rho;a} \searrow b}{\llparenthesis \boldsymbol{\lambda} x_0 . t \rrparenthesis_\rho \cdot a \searrow b} \qquad \frac{\llbracket T \rrbracket_{\rho,a} \searrow B}{\uparrow^{k \boxed{\coloneqq \max(i,j)}}_{\llparenthesis \mathbf{\Pi}\ A^i\ (x_0 . T^j) \rrparenthesis_\rho} e \cdot a \searrow \uparrow^j_B e\ (\downarrow^i_A a)}$$

$\boxed{\text{rec}\cdot(z.T^i, a, (x, y.s), b, \rho) \searrow c}$ rec-apply b to a and s under ρ with motive T^i is c

$$\frac{}{\text{rec}\cdot(z.T^i, a, (x, y.s), \mathbf{0}, \rho) \searrow a} \qquad \frac{\text{rec}\cdot(z.T^i, a, (x, y.s), b, \rho) \searrow c \quad \llbracket s \rrbracket_{\rho;b;c} \searrow c'}{\text{rec}\cdot(z.T^i, a, (x_0, y_1.s), \mathbf{suc}\ b, \rho) \searrow c'}$$

$$\frac{\llbracket T \rrbracket_{\rho;\uparrow^j_A e} \searrow B}{\text{rec}\cdot(z_0.T^i, a, (x, y.s), \uparrow^j_A e, \rho) \searrow \uparrow^i_B \llparenthesis \mathbf{rec}\ (z_0.T^i)\ a\ (x, y.s)\ e \rrparenthesis_\rho}$$

$\boxed{\text{unlift}\cdot\ a \searrow b}$ unlift-apply a is b

$$\frac{}{\text{unlift}\cdot\ \mathbf{lift}_i\ a \searrow a} \qquad \frac{}{\text{unlift}\cdot\ \uparrow^{k \boxed{\coloneqq j + i}}_{\mathbf{Lift}_j\ A^i} e \searrow \uparrow^i_A \mathbf{unlift}\ e}$$

Fig. 5. Relational definitions of helper evaluation functions.

Readback includes three mutually defined functions: $\mathrm{R}^{\mathrm{nf}}_n$ to readback normal values, $\mathrm{R}^{\mathrm{ne}}_n$ to readback neutral values, and ${}^i\mathrm{R}^{\mathrm{ty}}_n$ to readback normal type values at universe level i. The number n is needed to convert a de Bruijn level $\mathbf{x}_k$ into its corresponding de Bruijn index x_{n-k-1}. Readback of closures triggers evaluation of the body under extended environments. η-expansion happens when reading back a normal value ($\downarrow^i_A a$) when A is $\mathbf{\Pi}$ or $\mathbf{Lift}$, where evaluation is also triggered.

With evaluation and readback defined, our NbE algorithm follows the 3-step manner, as formally stated in the following definition. The context is first evaluated to an environment ($\uparrow^\Gamma$). Both t and T are evaluated in this environment ($\llbracket t \rrbracket_{\uparrow^\Gamma}$ and $\llbracket T \rrbracket_{\uparrow^\Gamma}$). The semantic value of t is type-value-directed reified ($\downarrow^i_{\llbracket T \rrbracket_{\uparrow^\Gamma}}$) to a normal semantic value then readback (R^{nf}) to a normal form.

Definition 4.1 (NbE Algorithm). For $\Gamma \vdash t :^i T$, $\boxed{\mathrm{NbE}^{T^i}_\Gamma(t)} \coloneqq \mathrm{R}^{\mathrm{nf}}_{|\Gamma|}\ (\downarrow^i_{\llbracket T \rrbracket_{\uparrow^\Gamma}} \llbracket t \rrbracket_{\uparrow^\Gamma})$

The last component is the context evaluation ($\uparrow^\Gamma$), which implements the first step of NbE à la Abel. This operation is inductively defined over the structure of the context. For empty context $\cdot$, it returns an empty environment, that is defined to return garbage (in our case, we return $\mathbf{0}$) for any de Bruijn index n. For context extension ($\Gamma, x : T^i$), it evaluates Γ to ρ, then evaluates T under ρ to A, and creates a semantic variable with de Bruijn level $|\Gamma|$ reflected by $\uparrow^i_A$. Notably, although context evaluation, as a procedure, occurs before the evaluation of t, its definition still depends on the evaluation function.

As reification takes a universe level as input, the NbE algorithm also requires this as input. This level (and other universe level ascriptions in t and T) is propagated and used throughout the evaluation and readback process.

During the design of evaluation and readback functions, it is tempting to add universe level checks as a guard to ensure the correctness of the algorithm. The equations with forms like $\boxed{\coloneqq \max(i,j)}$ and $\boxed{\coloneqq 0}$ in the rules in Fig. 5 and6 reflect such checks. For example, in $\mathrm{R}^{\mathrm{nf}}_n \downarrow^{i \boxed{\coloneqq 0}}_{\mathbf{N}} \mathbf{0} \searrow \mathtt{0}$, or equivalently represented as $\mathrm{R}^{\mathrm{nf}}_n \downarrow^0_{\mathbf{N}} \mathbf{0} \searrow \mathtt{0}$, means the readback only succeeds after checking the universe level of the reified value to be 0. All such checks are explicitly marked by dashed boxes. However, we later

$\boxed{\mathrm{R}^{\mathrm{nf}}_n\ d \searrow v}$ Readback from normal semantic value d to normal form v

$\boxed{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u}$ Readback from neutral semantic value e to neutral form u

$\boxed{{}^{i}\mathrm{R}^{\mathrm{ty}}_n\ A \searrow V}$ Readback from semantic value A of types to normal form V

$$\frac{{}^{i}\mathrm{R}^{\mathrm{ty}}_n\ A \searrow W}{\mathrm{R}^{\mathrm{nf}}_n \downarrow^{j = 1+i}_{\mathbf{Set}_i} A \searrow W} \qquad \frac{}{\mathrm{R}^{\mathrm{nf}}_n \downarrow^{i = 0}_{\mathrm{N}} \mathbf{0} \searrow \mathtt{0}} \qquad \frac{\mathrm{R}^{\mathrm{nf}}_n \downarrow^{i = 0}_{\mathrm{N}} a \searrow w}{\mathrm{R}^{\mathrm{nf}}_n \downarrow^{i = 0}_{\mathrm{N}} \mathbf{suc}\ a \searrow \mathtt{suc}\ w}$$

$$\frac{{}^{i}\mathrm{R}^{\mathrm{ty}}_n\ A \searrow W \qquad a \cdot \uparrow^{i}_{A} \mathbf{x}_n \searrow b \qquad [\![T]\!]_{\rho;\uparrow^{i}_{A}\mathbf{x}_n} \searrow B \qquad \mathrm{R}^{\mathrm{nf}}_{1+n} \downarrow^{j}_{B} b \searrow w}{\mathrm{R}^{\mathrm{nf}}_n \downarrow^{k = \max(i,j)}_{(\!|\mathbf{\Pi}\ A^{i}\ (x_0.T^{j})|\!)_\rho} a \searrow \lambda(x_0 : W^{i}).w}$$

$$\frac{\mathrm{unlift}\cdot\ a \searrow b \qquad \mathrm{R}^{\mathrm{nf}}_n \downarrow^{i}_{A} b \searrow w}{\mathrm{R}^{\mathrm{nf}}_n \downarrow^{k = j+i}_{(\mathbf{Lift}_j\ A^{i})} a \searrow \mathtt{lift}_j\ w} \qquad \frac{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u}{\mathrm{R}^{\mathrm{nf}}_n \downarrow^{j = 0}_{N} \uparrow^{i}_{A} e \searrow u} \qquad \frac{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u}{\mathrm{R}^{\mathrm{nf}}_n \downarrow^{k = i}_{(\uparrow^{j = 1+i}_{A} E)} \uparrow^{i}_{A'} e \searrow u}$$

$$\frac{}{\mathrm{R}^{\mathrm{ne}}_n\ \mathbf{x}_l \searrow x_{n-l-1}} \qquad \frac{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u \qquad \mathrm{R}^{\mathrm{nf}}_n\ d \searrow w}{\mathrm{R}^{\mathrm{ne}}_n\ e\ d \searrow u\ w} \qquad \frac{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u}{\mathrm{R}^{\mathrm{ne}}_n\ \mathbf{unlift}\ e \searrow \mathtt{unlift}\ u}$$

$$\frac{\begin{array}{c}[\![T]\!]_{\rho;\uparrow^{0}_{\mathrm{N}}\mathbf{x}_n} \searrow A \qquad {}^{i}\mathrm{R}^{\mathrm{ty}}_{1+n}\ A \searrow W \qquad [\![T]\!]_{\rho;\mathbf{0}} \searrow A' \qquad \mathrm{R}^{\mathrm{nf}}_n\ (\downarrow^{i}_{A'} a) \searrow w \\ [\![t]\!]_{\rho;\uparrow^{0}_{\mathrm{N}}\mathbf{x}_n;\uparrow^{i}_{A}\mathbf{x}_{1+n}} \searrow b \qquad [\![T]\!]_{\rho;\mathbf{suc}(\downarrow^{0}_{N}\mathbf{x}_n)} \searrow A'' \qquad \mathrm{R}^{\mathrm{nf}}_{2+n} \downarrow^{i}_{A''} b \searrow w' \qquad \mathrm{R}^{\mathrm{ne}}_n\ e \searrow u\end{array}}{\mathrm{R}^{\mathrm{ne}}_n\ (\!|\mathbf{rec}\ (z_0.T^{i})\ a\ (x_0, y_1.s)\ e|\!)_\rho \searrow \mathtt{rec}\ (z_0.W^{i})\ w\ (x_0.y_1.w')\ u}$$

$$\frac{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow V}{{}^{i}\mathrm{R}^{\mathrm{ty}}_n \downarrow^{j = i+1}_{A} e \searrow V} \qquad \frac{}{{}^{j = 1+i}\mathrm{R}^{\mathrm{ty}}_n\ \mathbf{Set}_i \searrow \mathtt{Set}_i} \qquad \frac{}{{}^{i = 0}\mathrm{R}^{\mathrm{ty}}_n\ \mathbf{N} \searrow \mathtt{N}}$$

$$\frac{{}^{i}\mathrm{R}^{\mathrm{ty}}_n\ A \searrow V \qquad [\![T]\!]_{\rho;\uparrow^{i}_{A}\mathbf{x}_n} \searrow B \qquad {}^{j}\mathrm{R}^{\mathrm{ty}}_{1+n}\ B \searrow W}{{}^{k = \max(i,j)}\mathrm{R}^{\mathrm{ty}}_n\ (\!|\mathbf{\Pi}\ A^{i}\ (x_0.T^{j})|\!)_\rho \searrow \Pi(x_0 : V^{i}).W^{j}} \qquad \frac{{}^{i}\mathrm{R}^{\mathrm{ty}}_n\ A \searrow W}{{}^{k = j+i}\mathrm{R}^{\mathrm{ty}}_n\ \mathbf{Lift}_j\ A^{i} \searrow \mathtt{Lift}_j\ W^{i}}$$

Fig. 6. Relational definitions of readback functions.

find these checks are unnecessary after the establishment of NbE soundness discussed in Sec. 5.3. That is, the NbE algorithm that fully removes all such checks is still sound and complete.

Such relaxations are possible because we only feed the NbE algorithm with well-typed terms. Other forms of relaxations also exist in the original NbE algorithms [Abel 2013], including relaxations of well-scopedness in converting de Bruijn indices and levels and equal-type-value check in reading back reified neutrals. The algorithm can certainly do more checks, but these checks are always satisfied by the well-typed terms. As long as the algorithm is designed properly, these properties will be maintained as an invariant. Though these checks affect the algorithm's behavior for ill-typed terms, it is not a concern both theoretically and practically. In practical type-checking of dependent-type systems, the normalization is still invoked on sub-terms that have already been type checked.

5 PER Model and Soundness and Completeness of NbE

This section develops an inductive-recursively defined PER model for (domain) values and a Kripke logical relation to show the completeness and soundness of NbE. There are several changes to adapt the definitions and proofs to the non-cumulative system: both the PER model and Kripke logical relations are refined to ensure that exact universe level information is maintained. For readability, definitions in this section are described in an informal mathematical language, while for a successful mechanization, the exact formulation in Agda is of grave importance, whose simplified signatures and more discussions are given later in Sec. 6. In this section, we no longer highlight the universe level annotations.

5.1 PER model

The PER model is used to justify extensionality rules in the semantics, as two terms that are equivalent up to η rules can be evaluated to different semantic values. We use $a \equiv a' \in \mathcal{A}$ to denote that a and a' are related by the PER $\mathcal{A}$ defined on $\mathrm{D} \times \mathrm{D}$. The notation $a \in \mathcal{A}$ means that there is some a' such that $a \equiv a' \in \mathcal{A}$ (equivalent to $a \equiv a \in \mathcal{A}$). We abuse the same notation for PERs defined on $\mathrm{D}^{\mathrm{ne}} \times \mathrm{D}^{\mathrm{ne}}$ and $\mathrm{D}^{\mathrm{nf}} \times \mathrm{D}^{\mathrm{nf}}$.

The definitions start with three extreme PERs, $\mathcal{N}e$, $\mathcal{N}f$ and $\mathcal{T}y_i$. $\mathcal{N}e$ relates two neutral semantic values if they can be readback to the same neutral form. $\mathcal{N}f$ relates two normal semantic values if they can be readback to the same normal form. $\mathcal{T}y_i$ relates two semantic type values if they can be readback to the same normal-form type on universe level i.

- $\boxed{e \equiv e' \in \mathcal{N}e}$ $:= \forall n, \exists u, \mathrm{R}^{\mathrm{ne}}_n\ e \searrow u$ and $\mathrm{R}^{\mathrm{ne}}_n\ e' \searrow u$
- $\boxed{d \equiv d' \in \mathcal{N}f}$ $:= \forall n, \exists w, \mathrm{R}^{\mathrm{nf}}_n\ d \searrow w$ and $\mathrm{R}^{\mathrm{nf}}_n\ d' \searrow w$
- $\boxed{A \equiv A' \in \mathcal{T}y_i}$ $:= \forall n, \exists V, {}^i\mathrm{R}^{\mathrm{ty}}_n\ A \searrow V$ and ${}^i\mathrm{R}^{\mathrm{ty}}_n\ A' \searrow V$

We also need a PER $\boxed{\mathcal{N}at}$ for base type N and another PER $\boxed{\mathcal{N}eu_i}$ to relate to semantic values reflected from neutral values. $\mathcal{N}at$ is defined inductively with three cases, and $\mathcal{N}eu_i$ has one.

$$\frac{}{\mathbf{0} \equiv \mathbf{0} \in \mathcal{N}at} \qquad \frac{a \equiv a' \in \mathcal{N}at}{\mathbf{suc}\ a \in \mathbf{suc}\ a' \in \mathcal{N}at} \qquad \frac{e \equiv e' \in \mathcal{N}e}{\uparrow^i_A e \equiv \uparrow^{i'}_{A'} e' \in \mathcal{N}at} \qquad \frac{e \equiv e' \in \mathcal{N}e}{\uparrow^i_A e \equiv \uparrow^i_{A'} e' \in \mathcal{N}eu_i}$$

With the base PERs defined, we then define the PERs of semantic type values via a family of inductive-recursive definitions [Dybjer 2000]. These definitions simultaneously define two PERs: (1) $\boxed{A \equiv B \in \mathcal{S}et_i}$, which inductively relates type values A and B, and (2) $\boxed{a \equiv b \in \mathcal{E}l_i(\mathcal{D})}$ given $\mathcal{D} :: A \equiv B \in \mathcal{S}et_i$, which recursively relates two values a and b whose type values are A and B. $\boxed{::}$ assigns a name to a predicate.[3] Conventionally, we pick $\mathcal{D}, \mathcal{E}, \mathcal{J}$ for predicate names. This style of definitions follows Abel [2013] and Abel et al. [2018]. The universe hierarchy of the PER model is then formed by performing a well-founded recursion on the universe level i, as we only refer to PERs that are already defined at lower universe levels for each case. $A \equiv B \in \mathcal{S}et_i$ consists of five cases, one for neutral types, and one for each semantic type values. The two bullet points – in each case explain the inductively defined case and the recursion over this case respectively.

- $\mathcal{D} :: \dfrac{E \equiv E' \in \mathcal{N}e}{\uparrow^{1+i}_A E \equiv \uparrow^{1+i}_{A'} E' \in \mathcal{S}et_i}$ and $\mathcal{E}l_i(\mathcal{D}) = \mathcal{N}eu_i$
  - Two reflected neutral type values are related on universe level i if the reflection happens on the next higher universe level.
  - Two values of reflected neutral type values are related if they are related by $\mathcal{N}eu_i$.
- $\mathcal{D} :: \dfrac{}{\mathbf{N} \equiv \mathbf{N} \in \mathcal{S}et_{i=0}}$ and $\mathcal{E}l_i(\mathcal{D}) = \mathcal{N}at$
  - Two $\mathbf{N}$ are related on universe level 0.
  - Two values of $\mathbf{N}$ are related if they are related by $\mathcal{N}at$.

[3] In mechanization, $\mathcal{D}$ represents a proof term of the judgment.

- $$\mathcal{D} :: \frac{}{\mathbf{Set}_j \equiv \mathbf{Set}_j \in \mathcal{S}et_{i=1+j}} \quad \text{and } \mathcal{E}l_i(\mathcal{D}) = \mathcal{S}et_j.$$
  - Two **Set** are related on universe level $1 + j$ if they are both $\mathbf{Set}_j$.
  - Two values of $\mathbf{Set}_j$ are related if they are related by $\mathcal{S}et_j$.
- $$\begin{array}{l} \mathcal{D}_1 :: \\ \mathcal{D}_2 :: \\ \mathcal{D} :: \end{array} \frac{\begin{array}{c} A \equiv A' \in \mathcal{S}et_j \\ \forall a \equiv a' \in \mathcal{E}l_j(\mathcal{D}_1).\llbracket T \rrbracket_{\rho;a} \equiv \llbracket T' \rrbracket_{\rho';a'} \in \mathcal{S}et_k \end{array}}{(\!|\mathbf{\Pi}\ A^j\ (x_0.T^k)|\!)_\rho \equiv (\!|\mathbf{\Pi}\ A'^j\ (x_0.T'^k)|\!)_{\rho'} \in \mathcal{S}et_{i=\max(j,k)}} \quad \text{and } \mathcal{E}l_i(\mathcal{D}) = \mathcal{P}i$$
  where $f \equiv f' \in \boxed{\mathcal{P}i} := \forall(\mathcal{E} :: a \equiv a' \in \mathcal{E}l_j(\mathcal{D}_1)), f \cdot a \equiv f' \cdot a' \in \mathcal{E}l_j(\mathcal{D}_2(\mathcal{E}))$. Note that $\mathcal{D}_2$ is a family of PERs parameterized over another PER.
  - Two $\mathbf{\Pi}$ type values are related on universe level $\max(j, k)$ if (1) their input type values are related on universe level j; (2) for all related values a, a' evaluating the output type T, T' at the extended environments $\rho; a$ and $\rho'; a'$ results in two related type values on universe level k;
  - Two values f, f' of $\mathbf{\Pi}$ type values are related if for all related values a, a' of type values A and A', the result of applying f to a and f' to a' are related. This reflects the extensionality of Π.
- $$\begin{array}{l} \mathcal{D}_1 :: \\ \mathcal{D} :: \end{array} \frac{A \equiv A' \in \mathcal{S}et_k}{\mathbf{Lift}_j\ A^k \equiv \mathbf{Lift}_j\ A'^k \in \mathcal{S}et_{i=j+k}} \quad \text{and } \mathcal{E}l_i(\mathcal{D}) = \mathcal{U}nlift$$
  where $a \equiv a' \in \boxed{\mathcal{U}nlift} := \text{unlift} \cdot a \equiv \text{unlift} \cdot a' \in \mathcal{E}l_k(\mathcal{D}_1)$
  - Two **Lift** values are related on universe level $j + k$ if (1) their inner type values are related on level k; (2) the lifted universe levels are both j;
  - Two values of **Lift** are related if the result of unlift-applying them are related. This reflects the extensionality of `Lift`.

Compared with PERs for cumulative universes, the PERs here have stricter conditions on universe levels in *all* cases. Type values must be related at a specific $\mathcal{S}et_i$ given by the equational side conditions. For example, in the non-cumulative setting, **N** and **N** must be related at universe level 0 and $\mathbf{Set}_j$ and $\mathbf{Set}_j$ must be related at universe level $1 + j$. In contrast, the cumulative setting would allow the former to be related at any universe level i, and the latter to be related at any universe level $i > j$. Meanwhile, although previous discussions suggest that the NbE algorithm might be irrelevant to the universe level information, tracking the universe levels in values makes such a precise PER definition possible. Otherwise, it is impossible, for example, to know the universe levels of domain and codomain in the Π case. More concretely, in the cumulative setting, in the Π-case values no longer carry any universe level information. Domain and co-domain types are just related at universe level i, which is the level of Π-types themselves:

$$\begin{array}{l} \mathcal{D}_1 :: \\ \mathcal{D}_2 :: \\ \mathcal{D} :: \end{array} \frac{\begin{array}{c} A \equiv A' \in \mathcal{S}et_i \\ \forall a \equiv a' \in \mathcal{E}l_i(\mathcal{D}_1).\llbracket T \rrbracket_{\rho;a} \equiv \llbracket T' \rrbracket_{\rho';a'} \in \mathcal{S}et_i \end{array}}{(\!|\mathbf{\Pi}\ A\ (x_0.T)|\!)_\rho \equiv (\!|\mathbf{\Pi}\ A'\ (x_0.T')|\!)_{\rho'} \in \mathcal{S}et_i} \quad \text{and } \mathcal{E}l_i(\mathcal{D}) = \mathcal{P}i$$

$$\text{where} f \equiv f' \in \boxed{\mathcal{P}i} := \forall(\mathcal{E} :: a \equiv a' \in \mathcal{E}l_i(\mathcal{D}_1)), f \cdot a \equiv f' \cdot a' \in \mathcal{E}l_i(\mathcal{D}_2(\mathcal{E}))$$

Several properties are expected for this PER model, including symmetry and transitivity. As we work in a type-theoretic meta-theory (i.e., Agda), we also need to handle proof-relevance explicitly, e.g., to show that although the $\mathcal{E}l$ function is defined on $\mathcal{D} :: A \equiv B \in \mathbf{Set}_i$, the returned PER is fully determined by A (or B). Statements of these properties are listed in Appendix C. The most important property is realizability, which is shown below. This property is also known as the "sandwiching" property, as it shows that our PER is sandwiched between two extreme PERs, $\mathcal{N}e$ and $\mathcal{N}f$.

Theorem 5.1 (Realizability). *Given $\mathcal{D} :: A \equiv A' \in \mathcal{S}et_i$*

- *$A \equiv A' \in \mathcal{T}y_i$;*
- *If $e \equiv e' \in \mathcal{N}e$, then $\uparrow^i_A e \equiv \uparrow^i_{A'} e' \in \mathcal{E}l_i(\mathcal{D})$;*
- *If $a \equiv a' \in \mathcal{E}l_i(\mathcal{D})$, then $\downarrow^i_A a \equiv \downarrow^i_{A'} a' \in \mathcal{N}f$.*

Finally, we can define PERs for contexts $\mathcal{D} :: \boxed{\Gamma \equiv \Delta \in Ctx}$ and environments $\boxed{\rho \equiv \phi \in \mathcal{E}l\Gamma(\mathcal{D})}$. The inductive-recursive definition is given below. The need for this induction-recursion instead of a direct elimination on the structure of Γ is again due to proof relevance of Agda [Hu et al. 2023]. Symmetry, transitivity and irrelevance theorems also hold for these two PERs.

- $$\mathcal{D} :: \frac{}{\cdot \equiv \cdot \in Ctx} \qquad \text{and } \mathcal{E}l\Gamma(\mathcal{D}) = \top$$
  where $\top$ means a trivial PER that relates everything
  - Two empty contexts are related.
  - Any two environments are related.
- $$\begin{array}{ll} \mathcal{D}_1 :: & \Gamma \equiv \Gamma' \in Ctx \\ \mathcal{D}_2 :: & \forall \rho \equiv \rho' \in \mathcal{E}l(\mathcal{D}_1).[\![T]\!]_\rho \equiv [\![T]\!]_{\rho'} \in \mathcal{S}et_i, \\ \hline \mathcal{D} :: & \Gamma, x_0 : T^i \equiv \Gamma', x_0 : T'^{i'} \in Ctx \end{array} \qquad \text{and } \mathcal{E}l\Gamma(\mathcal{D}) = Cons$$
  where $\rho \equiv \rho' \in Cons \coloneqq \mathcal{E} :: (\mathtt{drop}(\rho) \equiv \mathtt{drop}(\rho') \in \mathcal{E}l\Gamma(\mathcal{D}_1))$ and $\rho(0) \equiv \rho'(0) \in \mathcal{D}_2(\mathcal{E})$
  - $\Gamma, x_0 : T^i$ and $\Gamma', x_0 : T'^{i'}$ are related if (1) $i = i'$, (2) Γ and Γ' are recursively related, and (3) T and T' are evaluated to related values in ρ and ρ' of Γ and Γ'
  - ρ and ρ' of Γ, T^i and $\Gamma, T'^{i'}$ if (1) the dropped environments are recursively related, and (2) the 0-th value in them are related

Our context evaluation function creates an initial environment that is related (to itself) by the PER induced by Γ, that is, for any $\mathcal{D} :: \Gamma \equiv \Gamma \in Ctx$, $\uparrow^\Gamma \equiv \uparrow^\Gamma \in \mathcal{E}l\Gamma(\mathcal{D})$.

5.2 Completeness

The completeness of NbE can be decomposed into two parts: (1) syntactically equal terms are evaluated to related values; (2) related values are read back to the same normal form. (2) is already implied by the realizability of our PER. In this section, we will develop the proof of the first part.

Given the defined PERs and their properties, we can first define semantic context equivalence $\boxed{\vDash \Gamma \equiv \Delta} \coloneqq \Gamma \equiv \Delta \in Ctx$. Semantic context well-formedness is defined via semantic context equivalence, $\boxed{\vDash \Gamma} \coloneqq \vDash \Gamma \equiv \Gamma$. The semantic judgment of equivalent terms and equivalent substitutions are as follows.

- $\boxed{\Gamma \vDash s \equiv t :^i T} \coloneqq$
  - $\mathcal{D} :: \vDash \Gamma$;
  - For any related $\rho \equiv \rho' \in \mathcal{E}l\Gamma(\mathcal{D})$.
    - T is evaluated to related type values: $\mathcal{E} :: [\![T]\!]_\rho \equiv [\![T]\!]_{\rho'} \in \mathcal{S}et_i$;
    - s and t are evaluated to related values: $[\![s]\!]_\rho \equiv [\![t]\!]_{\rho'} \in \mathcal{E}l_i(\mathcal{E})$;
- $\boxed{\Gamma \vDash \sigma \equiv \tau : \Delta} \coloneqq$
  - $\mathcal{D}_1 :: \vDash \Gamma$ and $\mathcal{D}_2 :: \vDash \Delta$;
  - For any related $\rho \equiv \rho' \in \mathcal{E}l\Gamma(\mathcal{D}_1)$,
    σ and τ are evaluated to related environments: $[\![\sigma]\!]_s(\rho) \equiv [\![\tau]\!]_s(\rho') \in \mathcal{E}l\Gamma(\mathcal{D}_2)$;

Similarly, semantic typing and substitution typing are defined via semantic judgment of equivalent terms and substitutions. $\boxed{\Gamma \vDash t :^i T} \coloneqq \Gamma \vDash t \equiv t :^i T$ and $\boxed{\Gamma \vDash \sigma : \Delta} \coloneqq \Gamma \vDash \sigma \equiv \sigma : \Delta$. With all the semantic judgments defined, we can now state the fundamental theorem for completeness.

Theorem 5.2 (Fundamental Theorem for NbE Completeness).

- *If* $\vdash \Gamma$, *then* $\vDash \Gamma$;
- *If* $\Gamma \vdash t :^i T$, *then* $\Gamma \vDash t :^i T$;
- *If* $\Gamma \vdash \sigma : \Delta$, *then* $\Gamma \vDash \sigma : \Delta$;
- *If* $\vdash \Gamma \equiv \Delta$, *then* $\vDash \Gamma \equiv \Delta$;
- *If* $\Gamma \vdash s \equiv t :^i T$, *then* $\Gamma \vDash s \equiv t :^i T$;
- *If* $\Gamma \vdash \sigma \equiv \tau : \Delta$, *then* $\Gamma \vDash \sigma \equiv \tau : \Delta$.

Combining it with the realizability of PER completes the proof of the completeness of NbE, which states that it normalizes any two equivalent terms to the same normal form.

Theorem 5.3 (NbE Completeness). *If* $\Gamma \vdash s \equiv t :^i T$*, then* $\exists w$*, s.t.* $\mathrm{NbE}_\Gamma^{T^i}(s) \searrow w$ *and* $\mathrm{NbE}_\Gamma^{T^i}(t) \searrow w$.

5.3 Soundness

In this section, we establish the soundness of the NbE. As usual, the soundness proof requires a Kripke logical relation. We need two mutually defined relations, first by recursion on universe level i, then on $\mathcal{D} :: A \equiv B \in \mathcal{S}et_i$: (1) $\boxed{\Gamma \vdash T \circledR^i \mathcal{D}}$ between well-formed types T and type values A; (2) $\boxed{\Gamma \vdash t : T \circledR^i a \in \mathcal{E}l_i\ (\mathcal{D})}$ between well-formed terms t and values a. As this logical relation "glues" a term t with a semantic value a, it is also called a *gluing* model. As contexts may be weakened during the typing derivation, we need to directly encode that our logical relations are stable under weakenings. With explicit substitutions, weakenings are characterized by a restricted form of substitutions, called weakening substitutions, denoted by $\boxed{\kappa}$ in this section. Weakening substitutions are substitutions syntactically equivalent to the n-th composition of $\uparrow$ (0-th composition is $\mathtt{Id}$), whose formal definitions are shown below.

$$\frac{\Gamma \vdash \kappa \equiv \mathtt{Id} : \Delta}{\Gamma \vdash \kappa : \Delta} \qquad \frac{\Gamma \vdash \kappa' : (\Delta, x_0 : T^i) \quad \Gamma \vdash \kappa \equiv \uparrow \circ \kappa' : \Delta}{\Gamma \vdash \kappa : \Delta}$$

Before the definition for all type values in $\mathcal{S}et_i$, we first need to define the logical relation $\boxed{\Gamma \vdash t \circledR a \in \mathcal{N}at}$ for the base PER $\mathcal{N}at$,

$$\frac{\Gamma \vdash t \equiv \mathtt{0} :^0 \mathsf{N}}{\Gamma \vdash t \circledR \mathbf{0} \in \mathcal{N}at} \quad \frac{\Gamma \vdash t \equiv \mathsf{suc}\ s :^0 \mathsf{N} \quad \Gamma \vdash s \circledR a \in \mathcal{N}at}{\Gamma \vdash t \circledR \mathbf{suc}\ a \in \mathcal{N}at} \quad \frac{e \in \mathcal{N}e \quad \forall \kappa, \Delta \vdash \kappa : \Gamma \longrightarrow \Delta \vdash t[\kappa] \equiv \mathrm{R}^{\mathrm{ne}}_{|\Delta|} e :^0 \mathsf{N}}{\Gamma \vdash t \circledR \uparrow^i_A e \in \mathcal{N}at}$$

We can now define the two logical relations for each case of $\mathcal{S}et_i$ below. Similar to the definition of the PER model, the universe levels in each case are precisely tracked and propagated.

- $\mathcal{D} :: \dfrac{E \equiv E' \in \mathcal{N}e}{\uparrow^{1+i}_A E \equiv \uparrow^{1+i}_{A'} E' \in \mathcal{S}et_i}$
  - $\Gamma \vdash T \circledR^i \mathcal{D} :=$
    - $*$ T is well-typed: $\Gamma \vdash T :^{1+i} \mathtt{Set}_i$;
    - $*$ For any weakening substitution κ s.t. $\Delta \vdash \kappa : \Gamma$,
      - $\cdot$ $T[\kappa]$ is syntactically equivalent to the readback of E: $\Delta \vdash T[\kappa] \equiv \mathrm{R}^{\mathrm{ne}}_{|\Delta|} E :^{1+i} \mathtt{Set}_i$;
  - $\Gamma \vdash t : T \circledR^i (\uparrow^i_B e) \in \mathcal{E}l_i\ (\mathcal{D}) :=$
    - $*$ T is well-typed: $\Gamma \vdash T :^{1+i} \mathtt{Set}_i$;
    - $*$ t is well-typed: $\Gamma \vdash t :^i T$;
    - $*$ $e \in \mathcal{N}e$;
    - $*$ For any weakening substitution κ s.t. $\Delta \vdash \kappa : \Gamma$,
      - $\cdot$ $T[\kappa]$ is syntactically equivalent to the readback of E: $\Delta \vdash T[\kappa] \equiv \mathrm{R}^{\mathrm{ne}}_{|\Delta|} E :^{1+i} \mathtt{Set}_i$;
      - $\cdot$ $t[\kappa]$ is syntactically equivalent to the readback of e: $\Delta \vdash t[\kappa] \equiv \mathrm{R}^{\mathrm{ne}}_{|\Delta|} e :^i T[\kappa]$;
- $\mathcal{D} :: \dfrac{}{\mathsf{N} \equiv \mathsf{N} \in \mathcal{S}et_{i=0}}$
  - $\Gamma \vdash T \circledR^i \mathcal{D} :=$
    - $*$ T is syntactically equivalent to N: $\Gamma \vdash T \equiv \mathsf{N} :^1 \mathtt{Set}_0$;
  - $\Gamma \vdash t : T \circledR^i a \in \mathcal{E}l_i\ (\mathcal{D}) :=$
    - $*$ T is syntactically equivalent to N: $\Gamma \vdash T \equiv \mathsf{N} :^1 \mathtt{Set}_0$;

    - t and a are glued by the logical relation for $\mathcal{N}at$: $\Gamma \vdash t ® a \in \mathcal{N}at$;

- $$\mathcal{D} :: \frac{}{\mathbf{Set}_j \equiv \mathbf{Set}_j \in \mathcal{S}et_{i=1+j}}$$
  - $\Gamma \vdash T ®^i \mathcal{D} \coloneqq \Gamma \vdash T \equiv \mathtt{Set}_j :^{2+j} \mathtt{Set}_{1+j}$;
  - $\Gamma \vdash t : T ®^i a \in \mathcal{E}l_i\ (\mathcal{D}) \coloneqq$
    - T is syntactically equivalent to $\mathtt{Set}_j$: $\Gamma \vdash T \equiv \mathtt{Set}_j :^{2+j} \mathtt{Set}_{1+j}$;
    - t is well-typed: $\Gamma \vdash t :^i T$;
    - a is a type value at universe level j: $\mathcal{E} :: a \in \mathcal{S}et_j$;
    - t is a type and is glued with a: $\Gamma \vdash t ®^j \mathcal{E}$;
- $$\begin{array}{ll} \mathcal{D}_1 :: & A \equiv A' \in \mathcal{S}et_j \\ \mathcal{D}_2 :: & \forall a \equiv a' \in \mathcal{E}l_j(\mathcal{D}_1).\llbracket T \rrbracket_{\rho;a} \equiv \llbracket T' \rrbracket_{\rho';a'} \in \mathcal{S}et_k \\ \hline \mathcal{D} :: & (\!|\mathbf{\Pi}\ A^j\ (x_0.T^k)|\!)_\rho \equiv (\!|\mathbf{\Pi}\ A'^j\ (x_0.T'^k)|\!)_{\rho'} \in \mathcal{S}et_{i=\max(j,k)} \end{array}$$
  - $\Gamma \vdash T ®^i \mathcal{D} \coloneqq \exists$ types S, R
    - S and R are well-typed: $\Gamma \vdash S :^{1+j} \mathtt{Set}_j$ and $\Gamma, S^j \vdash R :^{1+k} \mathtt{Set}_k$;
    - T is syntactically equivalent to this Π type: $\Gamma \vdash T \equiv \Pi(x : S^j).R^k :^{1+\max(j,k)} \mathtt{Set}_{\max(j,k)}$;
    - For any weakening substitution κ s.t. $\Delta \vdash \kappa : \Gamma$,
      - $S[\kappa]$ is recursively glued with A: $\Delta \vdash S[\kappa] ®^j \mathcal{D}_1$:
      - For any term s and value b s.t. $\mathcal{E} :: b \in \mathcal{E}l_j\ (\mathcal{D}_1)$ and $\Delta \vdash s : S[\kappa] ®^j b \in \mathcal{E}l_j\ (\mathcal{D}_1)$, $\Delta \vdash R[\kappa, s : S^j/x] ®^k \mathcal{D}_2(\mathcal{E})$
  - $\Gamma \vdash t : T ®^i a \in \mathcal{E}l_i\ (\mathcal{D}) \coloneqq \exists$ types S, R
    - S and R are well-typed: $\Gamma \vdash S :^{1+j} \mathtt{Set}_j$ and $\Gamma, S^j \vdash R :^{1+k} \mathtt{Set}_k$;
    - T is syntactically equivalent to this Π type: $\Gamma \vdash T \equiv \Pi(x : S^j).R^k :^{1+i} \mathtt{Set}_i$;
    - t is well-typed: $\Gamma \vdash t :^i T$;
    - $a \in \mathcal{P}i$ ($\mathcal{P}i$ is defined in Sec. 5.1);
    - For any weakening substitution κ s.t. $\Delta \vdash \kappa : \Gamma$,
      - $\Delta \vdash S[\kappa] ®^j \mathcal{D}_1$
      - For any term s and value b s.t. $\mathcal{E} :: b \in \mathcal{E}l_j\ (\mathcal{D}_1)$ and $\Delta \vdash s : S[\kappa] ®^j b \in \mathcal{E}l_j\ (\mathcal{D}_1)$. $(t[\kappa]\ s)$ is recursively glued with $(a \cdot b)$: $\Delta \vdash (t[\kappa])\ s : R[\kappa, s : S^j/x] ®^k (a \cdot b) \in \mathcal{E}l(\mathcal{D}_2(\mathcal{E}))$
- $$\begin{array}{ll} \mathcal{D}_1 :: & A \equiv A' \in \mathcal{S}et_k \\ \hline \mathcal{D} :: & \mathbf{Lift}_j\ A^k \equiv \mathbf{Lift}_j\ A'^k \in \mathcal{S}et_{i=j+k} \end{array}$$
  - $\Gamma \vdash T ®^i \mathcal{D} \coloneqq \exists$ type S
    - S is well-typed: $\Gamma \vdash S :^{1+k} \mathtt{Set}_k$
    - T is syntactically equivalent to this `Lift` type: $\Gamma \vdash T \equiv \mathtt{Lift}_j\ S^k :^{1+i} \mathtt{Set}_i$
    - For any weakening substitution κ s.t. $\Delta \vdash \kappa : \Gamma$, $S[\kappa]$ is recursively glued with A: $\Delta \vdash S[\kappa] ®^k \mathcal{D}_1$
  - $\Gamma \vdash t : T ®^i a \in \mathcal{E}l_i\ (\mathcal{D}) \coloneqq \exists$ type S
    - S is well-typed: $\Gamma \vdash S :^{1+k} \mathtt{Set}_k$
    - T is syntactically equivalent to this `Lift` type: $\Gamma \vdash T \equiv \mathtt{Lift}_j\ S^k :^{1+i} \mathtt{Set}_i$
    - t is well-typed: $\Gamma \vdash t :^i T$;
    - $a \in \mathcal{U}nlift$ ($\mathcal{U}nlift$ is defined in Sec. 5.1);
    - For any weakening substitution κ s.t. $\Delta \vdash \kappa : \Gamma$, $\Delta \vdash (\mathtt{unlift}\ t)[\kappa] : S[\kappa] ®^k (\text{unlift}\cdot\ a) \in \mathcal{E}l_k(\mathcal{D}_1)$

Properties. This gluing model is preserved under equivalent terms, equivalent types, equivalent contexts and related type values and is monotonic with respect to weakening substitutions. Statements of these properties are listed in Appendix C. And once again, we need to handle proof-relevance. The most important property is still realizability. The realizability theorem states a similar"sandwich" property, but it now requires strengthening to hold under all extended contexts. To formalize these properties, we first introduce three definitions. Given $\mathcal{D} :: A \equiv B \in \mathcal{S}et_i$,

- $\boxed{\Gamma \vdash T \mathrel{\overline{®}^i} \mathcal{D}}$ $:= \Gamma \vdash T :^{1+i} \texttt{Set}_i, A \equiv B \in \mathcal{T}y_i$ and $\forall \kappa$ s.t. $\Delta \vdash \kappa : \Gamma$, $\Delta \vdash T[\kappa] \equiv {}^i\mathrm{R}^{\mathrm{ty}}_{|\Delta|} A :^{1+i} \texttt{Set}_i$.
- $\boxed{\Gamma \vdash t : T \mathrel{\underline{®}^i} e \in \mathcal{E}l_i(\mathcal{D})}$ $:= \Gamma \vdash t :^i T, \Gamma \vdash T ®^i \mathcal{D}$, $e \in \mathcal{N}e$, and $\forall \kappa$ s.t. $\Delta \vdash \kappa : \Gamma$, $\Delta \vdash t[\kappa] \equiv \mathrm{R}^{\mathrm{ne}}_{|\Delta|} e :^i T[\kappa]$.
- $\boxed{\Gamma \vdash t : T \mathrel{\overline{®}^i} a \in \mathcal{E}l_i(\mathcal{D})}$ $:= \Gamma \vdash t :^i T, \Gamma \vdash T ®^i \mathcal{D}$, $\downarrow^i_A a \equiv \downarrow^i_B a \in \mathcal{N}f$ and $\forall \kappa$ s.t. $\Delta \vdash \kappa : \Gamma$, $\Delta \vdash t[\kappa] \equiv \mathrm{R}^{\mathrm{nf}}_{|\Delta|} (\downarrow^i_A a) :^i T[\kappa]$.

Theorem 5.4 (Realizability). *Given $\mathcal{D} :: A \equiv B \in \mathcal{S}et_i$,*

- *If $\Gamma \vdash T ®^i \mathcal{D}$, then $\Gamma \vdash T \mathrel{\overline{®}^i} \mathcal{D}$;*
- *If $\Gamma \vdash t : T \mathrel{\underline{®}^i} e \in \mathcal{E}l_i(\mathcal{D})$, then $\Gamma \vdash t : T ®^i (\uparrow^i_A e) \in \mathcal{E}l_i\ (\mathcal{D})$;*
- *If $\Gamma \vdash t : T ®^i a \in \mathcal{E}l_i\ (\mathcal{D})$, then $\Gamma \vdash t : T \mathrel{\overline{®}^i} a \in \mathcal{E}l_i(\mathcal{D})$.*

Realizability implies that if $\Gamma \vdash t : T ®^i a \in \mathcal{E}l_i\ (\mathcal{D})$, then t is syntactically equal to the normal form obtained by reading back the reified value of a in all extended contexts from Γ. The last step is to define the logical relation that glues a substitution with an environment. Again, this is done by another inductive-recursive definition of semantic context well-formedness $\mathcal{D} :: \boxed{\Vdash \Gamma}$ and a substitution gluing model $\boxed{\Gamma \vdash \sigma ® \rho : \mathcal{E}l\mathcal{P}(\mathcal{D})}$. Here, the recursion is a bit different from previous ones as it returns a predicate on Γ, σ and ρ instead of a PER. σ is glued with ρ (under Γ) should be understood as Γ, σ, ρ satisfy the predicate returned by $\mathcal{E}l\mathcal{P}(\mathcal{D})$. This gluing model is also preserved under equivalent substitutions and contexts, and is monotonic with respect to weakening substitutions.

- $$\mathcal{D} :: \frac{}{\Vdash \cdot} \qquad \text{and } \mathcal{E}l\mathcal{P}(\mathcal{D}) = \mathcal{P}Nil$$
  where $\mathcal{P}Nil(\Delta, \tau, \phi) := \Delta \vdash \tau : \cdot$.
- $$\mathcal{D} :: \frac{\mathcal{D}_1 :: \Vdash \Gamma \quad \Gamma \vdash T :^{1+i} \texttt{Set}_i \quad \forall \Delta \vdash \sigma ® \rho : \mathcal{E}l\mathcal{P}(\mathcal{D}), \mathcal{E} :: [\![T]\!]_\rho \in \mathcal{S}et_i \text{ and } \Delta \vdash T[\sigma] ®^i \mathcal{E}}{\Vdash \Gamma, x_0 : T^i} \qquad \text{and } \mathcal{E}l\mathcal{P}(\mathcal{D}) = \mathcal{P}Cons$$
  where $\mathcal{P}Cons(\Delta, \tau, \phi) := \exists$ substitution γ and term t,
  - τ has the codomain context $(\Gamma, x_0 : T^i)$: $\Delta \vdash \tau : (\Gamma, x_0 : T^i)$;
  - $\uparrow \circ \tau$ is syntactically equivalent to γ: $\Delta \vdash \uparrow \circ \tau \equiv \gamma : \Gamma$;
  - t is syntactically equivalent to the topmost term in $(\Gamma, x_0 : T^i)$: $\Delta \vdash x_0[\tau] \equiv t :^i T[\tau]$;
  - Evaluation of T is related in $\mathcal{S}et_i$: $\mathcal{E} :: [\![T]\!]_{(\texttt{drop}\ \phi)} \in \mathcal{S}et_i$;
  - t is recursively glued with $\phi(0)$: $\Delta \vdash t : T[\gamma] ®^i \phi(0) \in \mathcal{E}l_i\ (\mathcal{E})$;
  - γ is recursively glued with $(\texttt{drop}\ \phi)$: $\Delta \vdash \gamma ® (\texttt{drop}\ \phi) : \mathcal{E}l\mathcal{P}(\mathcal{D}_1)$.

With all the setup in place, we are now ready to define the semantic well-formedness judgment of terms and substitutions and to state the fundamental theorem, which asserts that syntactic well-formedness implies semantic well-formedness.

- $\boxed{\Gamma \Vdash t :^i T}$ $:=$
  - $\mathcal{D} :: \Vdash \Gamma$;
  - For any $\Delta\ \sigma\ \rho$ s.t. $\Delta \vdash \sigma ® \rho : \mathcal{E}l\mathcal{P}(\mathcal{D})$,
    * T is evaluated to a type value at universe level i:$\mathcal{E} :: [\![T]\!]_\rho \in \mathcal{S}et_i$;
    * $t[\sigma]$ and $[\![t]\!]_\rho$ are glued: $\Delta \vdash t[\sigma] : T[\sigma] ®^i [\![t]\!]_\rho \in \mathcal{E}l_i\ (\mathcal{E})$
- $\boxed{\Gamma \Vdash \sigma : \Delta}$ $:=$
  - $\mathcal{D}_1 :: \Vdash \Gamma$ and $\mathcal{D}_2 :: \Vdash \Delta$;

– For any $\Psi\ \tau\ \rho$ s.t. $\Psi \vdash \tau \circledR \rho : \mathcal{ElP}(\mathcal{D}_1)$,
  * $\tau \circ \sigma$ and $[\![\sigma]\!]_s(\rho)$ are glued: $\Delta \vdash \tau \circ \sigma \circledR [\![\sigma]\!]_s(\rho) : \mathcal{ElP}(\mathcal{D}_2)$.

Theorem 5.5 (Fundamental Theorem for NbE Soundness).
- *If* $\vdash \Gamma$, *then* $\Vdash \Gamma$;
- *If* $\Gamma \vdash t :^i T$, *then* $\Gamma \Vdash t :^i T$;
- *If* $\Gamma \vdash \sigma : \Sigma$, *then* $\Gamma \Vdash \sigma : \Delta$.

Theorem 5.6 (NbE Soundness). *If* $\Gamma \vdash t :^i T$, *then* $\exists w$, *s.t.* $\mathrm{NbE}_{\Gamma}^{T^i}(t) \searrow w$ *and* $\Gamma \vdash t \equiv w :^i T$.

The soundness theorem is a corollary of the fundamental theorem. By applying the fundamental theorem, we know $\mathcal{D} :: \Gamma \Vdash t :^i T$ whose first conclusion is $\mathcal{E} :: \Vdash \Gamma$. Let $\sigma = \mathtt{Id}$ in the second conclusion of $\mathcal{D}$. Since $\Gamma \vdash \mathtt{Id} \circledR \uparrow^{\Gamma} : \mathcal{ElP}(\mathcal{E})$, we know $t[\mathtt{Id}]$ and $[\![t]\!]_{\uparrow^\Gamma}$ are glued in $\mathcal{E}l_i([\![T[\mathtt{Id}]]\!]_{\uparrow^\Gamma})$. The goal can then be concluded by applying the realizability theorem.

5.4 Consequences

Soundness and completeness justify the equivalence relation presented in Sec. 3, as they demonstrate that our NbE algorithm provides a decision procedure for checking this equivalence by normalizing two terms and comparing the normal forms syntactically. These two theorems also imply additional properties of the system. The following universe level exactness is a consequence of completeness.

Theorem 5.7 (Universe Level Exactness of $\mathtt{Set}_i$ and $\mathtt{N}$).
- *If* $\Gamma \vdash \mathtt{Set}_i \equiv \mathtt{Set}_{i'} :^k \mathtt{Set}_j$, *then* $i = i'$, $j = 1 + i$, *and* $k = 1 + j$;
- *If* $\Gamma \vdash \mathtt{N} \equiv \mathtt{N} :^j \mathtt{Set}_i$, *then* $i = 0$ *and* $j = 1$.

Injectivity of type constructors states that the syntactical equivalence of constructed types can be inverted to the equivalence of their components. This property cannot be established by a simple syntactic proof, mainly due to the presence of the transitivity rule. Although our proof depends on the completeness and soundness of NbE, this property may not necessarily depend on the normalizing property of the system [Lennon-Bertrand 2025].

Theorem 5.8 (Injectivity of Type Constructors Π and $\mathtt{Lift}$).
- *If* $\Gamma \vdash \Pi(x : S^i).T^j \equiv \Pi(x : S'^{i'}).T'^{j'} :^{1+k} \mathtt{Set}_k$, *then* $i = i'$, $j = j'$, $k = \max(i, j)$ *and* $\Gamma \vdash S \equiv S' :^{1+i} \mathtt{Set}_i$, *and* $\Gamma, S^i \vdash T \equiv T' :^{1+j} \mathtt{Set}_j$;
- *If* $\Gamma \vdash \mathtt{Lift}_j\ T^i \equiv \mathtt{Lift}_{j'}\ T^{i'} :^{1+k} \mathtt{Set}_k$, *then* $i = i'$, $j = j'$, $k = j + k$ *and* $\Gamma \vdash T \equiv T' :^{1+i} \mathtt{Set}_i$.

Since we add sufficient type annotations to terms (specifically, by adding argument types to λ-abstractions), and each type in this system has a unique universe level, we expect that a single term can only be typed with equivalent types at a unique universe level. With explicit substitutions, it needs to be proved together with another theorem that states each substitution produces equivalent codomain contexts. These two theorems are formally stated below.

Theorem 5.9 (Typing Uniqueness).
- *If* $\Gamma \vdash t :^i T$ *and* $\Gamma \vdash t :^{i'} T'$, *then* $i = i'$ *and* $\Gamma \vdash T \equiv T' :^{1+i} \mathtt{Set}_i$;
- *If* $\Gamma \vdash \sigma : \Delta$ *and* $\Gamma \vdash \sigma : \Delta'$, *then* $\vdash \Delta \equiv \Delta'$.

Proof by direct induction on this theorem almost works except for two elimination cases of Π and $\mathtt{Lift}$ types cases: application ($\Gamma \vdash t\ s :^i T$) and unlift ($\Gamma \vdash \mathtt{unlift}\ t :^i T$). The situation of these two cases is similar. Take the unlift case as an example. Given $\Gamma \vdash t :^{j+i} \mathtt{Lift}_j\ T^i$ and $\Gamma \vdash t :^{j'+i'} \mathtt{Lift}_j\ 'T'^{i'}$, the induction hypothesis only shows $\Gamma \vdash \mathtt{Lift}_j\ T^i \equiv \mathtt{Lift}_{j'}\ T'^{i'} :^{1+j+i} \mathtt{Set}_{j+i}$. To conclude $\Gamma \vdash T^i \equiv T'^{i'} :^{1+i} \mathtt{Set}_i$, we need Thm. 5.8. This dependency suggests that there could not be a simple syntactic proof to conclude this uniqueness theorem.

The other usual consequences of normalization are *logical consistency* and *canonicity*. Both proofs use standard techniques based on the soundness and completeness of NbE. Our consistency is phrased as there are some uninhabited types in the empty context. This phrasing does not require one explicit "false" type.[4]

THEOREM 5.10 (CONSISTENCY). $\cdot \vdash t :^{1+i} \Pi(x : \mathsf{Set}_i^{1+i}).x^i$ *is false.*

The following theorem shows that a closed term t of type N must be equivalent to a term constructed by the introduction form of N (i.e., $\mathtt{0}$ and suc) only.

THEOREM 5.11 (CANONICITY OF N). *If* $\cdot \vdash t :^0 \mathsf{N}$, *then* $\cdot \vdash t \equiv \mathsf{suc}^n\ \mathtt{0} :^0 \mathsf{N}$ *for some* $n \in \mathbb{N}$.

6 Comparison between Cumulativity and Non-cumulativity

In previous sections, we have described the models in an informal mathematical language. However, the description has taken various shortcuts for conciseness. To ensure rigor, we mechanize all our results in Agda. Our mechanization consists of two systems, an MLTT with a non-cumulative universe, the system described in this paper, and an MLTT with a cumulative universe, for comparing definitions and mechanization. The cumulative version can be viewed as a back-port of Hu et al. [2023]'s mechanization by removing modality-related features. Due to non-cumulativity, we observe significantly more complications in mechanization. Consequently, we must tame various details like proof relevance in Agda to make sure that the proofs of completeness and soundness remain manageable. As a quantitative indicator, the type-checking time increases from 2 minutes to 9 minutes in Agda comparing these two versions.

6.1 Differences in PER Models

Compared to mechanization of the cumulative universe, the uniqueness property induced by non-cumulativity has led to the need for explicit management of universe levels. The distinction between cumulativity and non-cumulativity has become evident from the type signatures of the PER models:

```
-- Cumulative
module PERDef (i : ℕ) (Univ : ∀ {j} → j < i → D → D → Set) where
  mutual
    data 𝕌 : D → D → Set
    El : ∀ {A B} → 𝕌 A B → D → D → Set

-- Non-cumulative
module PERDef where
  mutual
    data 𝕌 i (Univ : ∀ {j} → j < i → D → D → Set) : D → D → Set
    El : ∀ {A B} i (Univ : ∀ {j} → j < i → D → D → Set) →
           𝕌 i Univ A B → D → D → Set
```

The PER models are defined in their respective `PERDef` modules. With cumulativity, the module is parameterized by two arguments: `i : ℕ` fixing the universe level, and `Univ` for well-foundedness of 𝕌. In particular, `𝕌 : D → D → Set` is a binary relation between semantic types, so `Univ` allows us to have access to all previous `𝕌 j` for `j < i`. `El` relates two semantic values given a relation between two semantic types. `Univ` is eventually provided by a proof of well-foundedness for both kinds of hierarchies:

```
𝕌 : ℕ → D → D → Set
𝕌 i = PERDef.𝕌 i {- omitted proof via a well-founded induction -}
```

[4]The impredicative version of the chosen type is an encoding of the usual "false" type

The main difference in the PER models is the placement of `i` and `Univ`: with cumulativity, they are *fixed* parameters to the module, while non-cumulativity requires us to manage them in the definitions. The impact of this difference is significant in cases where multiple types with potentially different levels are involved, e.g., Π types, whose pen-and-paper definitions are given in Sec. 5.1, for both non-cumulative and cumulative settings.

```
-- Cumulative
  Π  : (iA : 𝕌 A A′) →
       (∀ {a a′} → El iA a a′ → ΠRT T (ρ ↦ a) T′ (ρ′ ↦ a′) 𝕌) →
       𝕌 (Π A T ρ) (Π A′ T′ ρ′)
```

In this definition, ΠRT is a predicate that evaluates `T` and `T′` in extended environments ρ `↦ a` (representing $\rho; a$ in Agda) and ρ' `↦ a′` respectively, and relates resulting semantic types by 𝕌. Specifically, it is always the same 𝕌 involved, because with cumulativity, semantics of types on a lower level also exist on all higher levels, i.e., `i`. On the other hand, with non-cumulativity, we need to manage not only the universe levels, but also *proofs* of `Univ`, due to proof relevance in Agda:

```
-- Non-cumulative
  Π : ∀ {j k} →
      let Univj : ∀ {l} → l < j → D → D → Set -- definition omitted
          Univk : ∀ {l} → l < k → D → D → Set -- definition omitted
      in (iA : 𝕌 j Univj A A′) →
         (∀ {a a′} → El j Univj iA a a′ → ΠRT T (ρ ↦ a) T′ (ρ′ ↦ a′) (𝕌 k Univk)) →
         𝕌 (max j k) Univ (Π j A (T ↙ k) ρ) (Π j′ A′ (T′ ↙ k′) ρ′)
```

Compared to the minimal semantics of Π with cumulativity, in the non-cumulative case, we must carry two proofs `Univj` and `Univk`, because the input types live on level `j` and the output types on `k`, which are both different from `max j k`, the level of Π.

Worse yet, this complication further propagates throughout the proofs of various properties of the PER model. Since both `i` and `Univ` are fixed as module parameters in the cumulative setting, subsequent proofs about the PER model are completely oblivious to the details about well-foundedness in `Univ` after some small technical efforts. On the other hand, the same technique does not seem to work with non-cumulativity. Difficult properties often require to directly work on `PERDef.𝕌` instead of the top-level 𝕌, causing much more effort in managing proof relevance. Proofs appear less clean than the cumulative case. Type-checking time also increases, because extra parameters drive Agda to perform non-trivial higher-order unifications in the proofs.

6.2 Challenges in Kripke Models

If the complication in the PER model is moderate, then problems in the Kripke model have reached a new next level. Since cumulativity allows us to isolate the details about well-foundedness of universe levels, we are able to give definitions of the Kripke model by a direct recursion on the top-level PER model 𝕌:

```
-- Cumulative
module Glu i (rec : ∀ {j} → j < i → ∀ {A B} → Ctx → Typ → 𝕌 j A B → Set) where
  mutual
    _⊢_®_ : Ctx → Typ → 𝕌 i A B → Set
    _⊢_∶_®_∈El_ : Ctx → Exp → Typ → D → 𝕌 i A B → Set
```

Here, `_⊢_®_` gives the Kripke relation for types and `_⊢_∶_®_∈El_` for terms. They are both defined by recursion on `𝕌 i A B`. The parameter `rec` is similar to `Univ` in the PER model to express well-foundedness of the Kripke model and is filled in by a well-foundedness proof, which we omit for brevity. With non-cumulativity, the Kripke model is defined in a similar manner to the PER model by propagating `rec` inwards. In fact, the actual definition is more complex, because to successfully recurse on the PER model, we must work with a general `Univ`:

```
-- Non-cumulative
module Glu where
  mutual
    ⟦_,_,_⟧_⊢_®_ : ∀ i (rec : ∀ {j} (j<i : j < i) (univ : ∀ {l} → l < j → D → D → Set) {A B} →
                                Ctx → Typ → PERDef.𝕌 j univ A B → Set) →
           (Univ : ∀ {j} → j < i → D → D → Set) →
           Ctx → Typ → PERDef.𝕌 i Univ A B → Set
    ⟦_,_,_⟧_⊢_:_®_∈El_ : ∀ i (rec : ∀ {j} (j<i : j < i)
                                    (univ : ∀ {l} → l < j → D → D → Set) {A B} →
                                    Ctx → Typ → PERDef.𝕌 j univ A B → Set)
            (Univ : ∀ {j} → j < i → D → D → Set) →
             Ctx → Exp → Typ → D → PERDef.𝕌 i Univ A B → Set
```

The type signatures are much more complex now. First, `i` is the given universe level. We then need `rec` for well-foundedness of the model in `i`. However, `rec` has a longer type due to the need to directly work with `PERDef.𝕌` and an extra parameter `univ` that is passed to `PERDef.𝕌`. This extra parameter is to capture the fact that the overall model is parameterized by `Univ`, the well-foundedness predicate to the input `PERDef.𝕌`. The types after `Univ` are the desired types that we would like the Kripke predicates to possess, which are what cumulativity would have brought us. The definitions thus proceed by recursion on `PERDef.𝕌 i Univ A B`. For each case, we suffer from the need of maintaining equality between proofs due to proof relevance in Agda and how the PER model is defined. The definitions are too verbose to put in the paper, so we refer interested readers to our accompanying artifact.

Challenges continue to escalate as we establish properties about the Kripke model. Not being able to isolate the well-foundedness proof entirely has been the top problem during proving. It leads to longer lemma statements, longer proofs, more difficult unification problems for Agda to solve, and hence much longer type-checking time.

Despite that the choice between cumulativity and non-cumulativity often results in a debate, we see that, at least in terms of mechanization of NbE, cumulativity provides significant simplifications, which might contribute to a critical factor of consideration at the early stage of design. We are not sure whether our mechanization of non-cumulative MLTT can be further improved, so that it becomes less resource-consuming. We leave this problem as a future work.

7 The Unascribed System

As the uniqueness theorem suggests, levels are entirely determined by typing derivations. This naturally raises the question of whether these universe level annotations can be removed. In this section, we prove that removing all universe level annotations yields a system that is sound and complete with respect to the original one, and there is a sound and complete NbE algorithm for this system as well. We refer to the original system as the *ascribed* system and the new system as the *unascribed* system. This transformation also brings the system closer to the practical syntax and rules of practical proof assistants like Agda and Lean. To save space, the concrete definitions used in the unascribed system are listed in Appendix B.

7.1 Syntactic Soundness and Completeness

The syntax and rules of the unascribed system are identical to those of the ascribed system presented in Sec. 3, except that all universe level annotations ^{i} are removed. For clarity, in this section, we use different colors to distinguish contexts, terms, types, substitutions, and judgments in the two systems. Specifically, we use Γ, t, T, σ to denote elements of the unascribed system and

Γ, t, T, σ for those in the ascribed system. Since the syntax of the two systems differs, we first need to define an appropriate way to relate contexts, expressions/types, and substitutions. As the only difference between the two systems is the presence of universe level annotations, an erasure function, denoted as $\lfloor \_ \rfloor$, which removes all these annotations in terms while preserving the remaining structure serves as a suitable relation. For example, $\Pi(x : \mathtt{Set}_i^{1+i}).x^i$ is erased to $\Pi(x : \mathtt{Set}_i).x$. This function is many to one. The same erasure function can be naturally extended to each syntactic category, and we reuse the same notion for them, e.g. $\lfloor \Gamma \rfloor$ erases universe level annotations in Γ. The definitions of these functions are straightforward and are omitted here. For brevity, we sometimes omit the erasure function and instead use the same symbol in different colors to indicate erasure, e.g., t and t in the same textual context indicate $\lfloor t \rfloor = t$.

Completeness. The completeness proof follows straightforwardly by induction on the derivations of the ascribed system. Thanks to the totality of erasure functions, they can be applied to any contexts, terms, types, substitutions without requiring knowledge of their well-formedness/well-typedness.

Theorem 7.1 (Syntactic Completeness).

- *If* $\vdash \Gamma$, *then* $\vdash \Gamma$;
- *If* $\Gamma \vdash t :^i T$, *then* $\Gamma \vdash t : T$;
- *If* $\Gamma \vdash \sigma : \Delta$, *then* $\Gamma \vdash \sigma : \Delta$;
- *If* $\vdash \Gamma \equiv \Delta$, *then* $\vdash \Gamma \equiv \Delta$;
- *If* $\Gamma \vdash s \equiv t :^i T$, *then* $\Gamma \vdash s \equiv t : T$;
- *If* $\Gamma \vdash \sigma \equiv \tau : \Delta$, *then* $\Gamma \vdash \sigma \equiv \tau : \Delta$.

Soundness. However, the soundness requires "reconstructing" such levels. That is, prove the existence of such levels. Even if we have properties of the ascribed system, the conclusion given by the existential quantifier is too weak. For example, to prove the Π case, the induction hypothesis will give us $\Gamma_1 \vdash S_1 :^{1+i} \mathtt{Set}_i$ and $\Gamma_2, S_2 \vdash T :^{1+j} \mathtt{Set}_j$. We know nothing about the relationship between Γ_1 and Γ_2 and between S_1 and S_2 except that they are both erased to the same thing. The proof is thus blocked. To prove the soundness by induction, we need stronger conclusions to relate two contexts, expressions and substitutions given by the existential quantifier. The proper relation is defined as follows.

- $\boxed{\Gamma \lfloor\equiv\rfloor \Gamma'}$:= $\forall \Gamma_1\ n$, $\lfloor \Gamma_1 \rfloor = (\mathtt{drop}\ \ n\ \ \Gamma')$, and $\vdash \Gamma_1$, then $\vdash (\mathtt{drop}\ \ n\ \ \Gamma) \equiv \Gamma_1$
- $\boxed{\Gamma \vdash t \lfloor\equiv\rfloor t'}$:=$\forall i_1\ t_1\ T_1$, $\lfloor t_1 \rfloor = t'$, and $\Gamma \vdash t_1 :^{i_1} T_1$, then $\Gamma \vdash t \equiv t_1 :^{i_1} T_1$
- $\boxed{\Gamma \vdash \sigma \lfloor\equiv\rfloor \sigma'}$:= $\forall \sigma_1\ \Delta_1$, if $\lfloor \sigma_1 \rfloor = \sigma'$, and $\Gamma \vdash \sigma_1 : \Delta_1$, then $\Gamma \vdash \sigma \equiv \sigma_1 : \Delta_1$

These relations state that all possible Γ_1, t_1, σ_1 that are well-formed and erased to Γ', t', σ' are equivalent and thus they are "interchangeable" with the specific Γ, t, σ given by the existential quantifier. There are two interesting points about these auxiliary relations. Firstly, $\Gamma \lfloor\equiv\rfloor \Gamma'$ talks about all the prefixes of Γ and Γ', achieved by the additional quantification of n and an ordinary list-`drop` operation. This generalization is essential for the formation and equivalence cases of ↑. Secondly, $\Gamma \vdash t \lfloor\equiv\rfloor t'$ does not include the type of t or t'. The quantified t_1 can be well-typed under any level i_1 and type T_1. On one hand, this is necessary for the cons case of context equivalence $\vdash \Gamma_1, T_1 \equiv \Gamma_2, T_2$. Since we do not have the information, from inverting the erasure operation, that two types T_1 and T_2 are at the same universe level, the premise must be general enough to reason about this. On the other hand, this is feasible as expressions contain enough information to recover types. By strengthening the original soundness theorem with the help of these relations, the final fundamental theorem to prove by induction is:

Theorem 7.2 ((Fundamental Theorem for) Syntactic Soundness).

- *If* $\vdash \Gamma$, *then* $\exists \Gamma$, *s.t.* $\lfloor \Gamma \rfloor = \Gamma$ *and* $\vdash \Gamma$, *and* $\Gamma \lfloor\equiv\rfloor \Gamma$;
- *If* $\Gamma \vdash t : T$, *then* $\exists i, \Gamma, t, T$, *s.t.* $\lfloor \Gamma \rfloor = \Gamma$, $\lfloor t \rfloor = t$, $\lfloor T \rfloor = T$ *and* $\Gamma \vdash t :^i T$, *and* $\Gamma \lfloor\equiv\rfloor \Gamma$ *and* $\Gamma \vdash t \lfloor\equiv\rfloor t$;

- *If* $\Gamma \vdash \sigma : \Delta$*, then* $\exists \Gamma, \sigma, \Delta$*, s.t.* $\lfloor \Gamma \rfloor = \Gamma$, $\lfloor \sigma \rfloor = \sigma$, $\lfloor \Delta \rfloor = \Delta$ *and* $\Gamma \vdash \sigma : \Delta$*, and* $\Gamma \lfloor \equiv \rfloor \Gamma$, $\Gamma \vdash \sigma \lfloor \equiv \rfloor \sigma$*, and* $\Delta \lfloor \equiv \rfloor \Delta$*;*
- *If* $\vdash \Gamma \equiv \Delta$*, then* $\exists \Gamma, \Delta$*, s.t.* $\lfloor \Gamma \rfloor = \Gamma$, $\lfloor \Delta \rfloor = \Delta$ *and* $\vdash \Gamma \equiv \Delta$*, and* $\Gamma \lfloor \equiv \rfloor \Gamma$ *and* $\Delta \lfloor \equiv \rfloor \Delta$*;*
- *If* $\Gamma \vdash s \equiv t : T$*, then* $\exists i, \Gamma, t, s, T$*, s.t.* $\lfloor \Gamma \rfloor = \Gamma$, $\lfloor s \rfloor = s$, $\lfloor t \rfloor = t$, $\lfloor T \rfloor = T$ *and* $\Gamma \vdash s \equiv t :^{i} T$*, and* $\Gamma \lfloor \equiv \rfloor \Gamma$, $\Gamma \vdash s \lfloor \equiv \rfloor s$*, and* $\Gamma \vdash t \lfloor \equiv \rfloor t$*;*
- *If* $\Gamma \vdash \sigma \equiv \tau : \Delta$*, then* $\exists \Gamma, \sigma, \tau, \Delta$*, s.t.* $\lfloor \Gamma \rfloor = \Gamma$, $\lfloor \sigma \rfloor = \sigma$, $\lfloor \tau \rfloor = \tau$, $\lfloor \Delta \rfloor = \Delta$ *and* $\Gamma \vdash \sigma \equiv \tau : \Delta$*, and* $\Gamma \lfloor \equiv \rfloor \Gamma$, $\Gamma \vdash \sigma \lfloor \equiv \rfloor \sigma$, $\Gamma \vdash \tau \lfloor \equiv \rfloor \tau$ *and* $\Delta \lfloor \equiv \rfloor \Delta$*.*

Its proof makes heavy use of the typing uniqueness theorem (Thm. 5.9). The syntactic soundness theorem is a direct corollary of this fundamental theorem by removing the extra strengthened conclusions.

With unascribed soundness and completeness, properties in the ascribed system can be transported to this unascribed system, with logical consistency being an example.

Theorem 7.3 (Consistency). $\cdot \vdash t : \Pi(x : \mathtt{Set}_i).x$ *is false.*

Proof. Assuming $\cdot \vdash t : \Pi(x : \mathtt{Set}_i).x$ is derivable and applying the unascribed soundness theorem, we get $\cdot \vdash t :^{j_1} \Pi(x : \mathtt{Set}_i^{j_2}).x^{j_3}$ for some j_1, j_2, j_3. By the presupposition lemma of the ascribed system, we can conclude $\cdot \vdash \Pi(x : \mathtt{Set}_i^{j_2}).x^{j_3} :^{1+j_1} \mathtt{Set}_{j_1}$. The only possible case for this typing to hold is $j_1 = 1 + i$, $j_2 = 1 + i$ and $j_3 = i$. However, this case is also impossible according to the consistency of the ascribed system. □

Other examples of transported theorems are context conversion (Thm. 3.1) and presupposition (Thm. 3.2). The proof strategy is straightforward by using both unascribed soundness and completeness.[5] With presupposition theorems, the additional well-formedness premises marked in Fig. 1 can also be eliminated for the unascribed system.

7.2 NbE of the Unascribed System

Despite the syntactic equivalence of these two systems, the NbE algorithm for the ascribed system cannot be applied to the unascribed system directly. The algorithm still takes ascribed terms as inputs, and performs additional checks based on the level information. Although based on previous discussions, most checks can be eliminated and the NbE algorithm should be universe-level irrelevant, it is not immediately obvious that there is an NbE algorithm that works in the unascribed system directly and enjoys desirable properties of soundness and completeness.

A promising and simple candidate of such an NbE algorithm is the algorithm that resembles the ascribed NbE algorithm, but removes all checks and processing related to universe levels. That is, the NbE algorithm takes terms and types of the unascribed syntax as input, evaluates them to unascribed domain values and read back to unascribed normal forms. The definition of the unascribed domain and evaluation /read-back functions of this NbE can be roughly understood as those presented in Sec. 4 by removing all universe-level related annotations and are omitted here for brevity. The detailed definitions can be found in Appendix B. This algorithm is no more complex or less efficient than its cumulative counterpart, as no additional universe level information is kept, not to mention any checks on them. In the remaining section, we will establish the soundness and completeness of this NbE algorithm with respect to the unascribed system.

The proof starts by proving the output completeness of the unascribed NbE to the ascribed NbE. The theorem means that if the ascribed NbE produces a result for some t of type T (of universe level i) in Γ, this unascribed NbE also produces a result for input term t of type T in Γ, and their resulting normal forms match up to erasure. We first extend the definition of erasure function to

[5]For context conversion, we need to apply the fundamental theorem of unascribed soundness directly once.

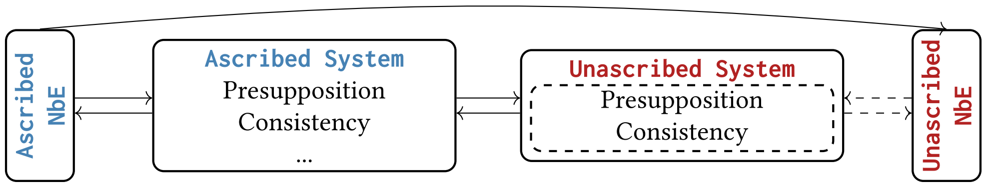

Fig. 7. Overall proof structure. Arrows indicate soundness and completeness. Properties in dashed boxes and represented by dashed arrows are proved indirectly.

(normal, neutral) semantic values $\lfloor a \rfloor$, $\lfloor d \rfloor$, $\lfloor e \rfloor$ and environments $\lfloor \rho \rfloor$, in a straightforward way. The output completeness of the unascribed NbE is a composition of the following two theorems. Note that, since we do not track the universe level information anymore, type readback function $\mathrm{R}^{\mathrm{ty}}_n$ of the unascribed NbE neither takes it as an input.

Theorem 7.4 (Output Completeness of the Unascribed Evaluation).

- *If* $[\![t]\!]_\rho \searrow a$, *then* $[\![t]\!]_\rho \searrow a$;
- *If* $[\![\sigma]\!]_s(\rho) \searrow \phi$, *then* $[\![\sigma]\!]_s(\rho) \searrow \phi$;
- *If* $f \cdot a \searrow b$, *then* $f \cdot a \searrow b$;
- *If* $\mathit{rec}\cdot(z.T^i, a, (x, y.s), b, \rho) \searrow c$, *then* $\mathit{rec}\cdot(z.T, a, (x, y.s), b, \rho) \searrow c$;
- *If* $\mathrm{unlift}\cdot\ a \searrow b$, *then* $\mathrm{unlift}\cdot\ a \searrow b$.

Theorem 7.5 (Output Completeness of the Unascribed Readback).

- *If* $\mathrm{R}^{\mathrm{nf}}_n\ d \searrow v$, *then* $\mathrm{R}^{\mathrm{nf}}_n\ d \searrow v$;
- *If* $\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u$, *then* $\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u$;
- *If* ${}^i\mathrm{R}^{\mathrm{ty}}_n\ V \searrow A$, *then* $\mathrm{R}^{\mathrm{ty}}_n\ V \searrow A$.

Their proof is done by direct induction. Combining them together, we have the completeness of the unascribed NbE to the ascribed one, whose meaning is described above. This theorem holds for all Γ, t and T, regardless of their well-formedness. Again, this is thanks to the fact that we adopt the untyped domain model for NbE.

Theorem 7.6 (Output Completeness of the Unascribed NbE). *If* $\mathrm{NbE}^{T^i}_\Gamma(t) \searrow w$, *then* $\mathrm{NbE}^{T}_\Gamma(t) \searrow w$.

With the output completeness of the unascribed NbE to the ascribed NbE (only this single direction is needed), we can immediately conclude the soundness and completeness of the unascribed NbE algorithm with respect to the unascribed system.

Theorem 7.7 (Soundness and Completeness of the Unascribed NbE).

- *If* $\Gamma \vdash t : T$, *then* $\exists w$, $\mathrm{NbE}^{T}_\Gamma(t) \searrow w$ *and* $\Gamma \vdash t \equiv w : T$;
- *If* $\Gamma \vdash s \equiv t : T$, *then* $\exists w$, $\mathrm{NbE}^{T}_\Gamma(s) \searrow w$ *and* $\mathrm{NbE}^{T}_\Gamma(t) \searrow w$.

Proof. Take soundness as an example. Give $\Gamma \vdash t : T$, the proof takes 4 steps: (1) applying the unascribed soundness theorem (Thm. 7.2) to conclude $\Gamma \vdash t :^i T$ for some i; (2) applying the soundness theorem of the ascribed NbE (Thm. 5.6) to conclude $\mathrm{NbE}^{T^i}_\Gamma(t) \searrow w$ and $\Gamma \vdash t \equiv w :^i T$ for some w; (3) applying the output completeness theorem of the unascribed NbE (Thm. 7.6) to conclude $\mathrm{NbE}^{T}_\Gamma(t) \searrow w$; (4) applying the unascribed completeness theorem (Thm. 7.1) to conclude $\Gamma \vdash t \equiv w : T$. ◻

In summary, this result largely closes some of the gap between the system and the NbE algorithm we study and the practical ones used in proof assistants like Agda. As shown in Fig. 7, all the proofs in the unascribed system are achieved by using the ascribed system as an intermediate system, proving properties in it and then transporting all the properties back to the unascribed system using syntactical soundness and completeness. Though current proof takes a large roundtrip, it is suspicious if we can skip the ascribed system and directly prove the soundness and completeness of the unascribed NbE algorithm with respect to the unascribed system. Without the precise universe level given by the ascribed system, we have to use an existential quantifier to get the level i in the semantic proof (as the case of the cumulative proof). However, for the non-cumulative case, we cannot use the trick to lift all related values to a high-enough universe level. Instead, we must further prove that two i's given by the existential quantifiers are the same. Strengthening the logical relation may be possible, but it is not obvious, and will further complicate the already complex mechanization. Note, even if such a new proof strategy is feasible, the need for the precise PER model presented in Sec. 5 (and consequently the technical challenges mentioned in Sec. 6) can hardly be exempted, as non-cumulativity forces relating type values at the the one and only correct universe level. In retrospect, this *ascribed then unascribed* proof strategy makes the whole mechanization more modular and manageable as all the uniqueness related proofs are done in the syntactic level and conceptually coincides with the common practice to prove properties in an "elaborated" system [Ferreira and Pientka 2014; Gundry 2013; Pottier 2014], although we did not foresee this from the beginning.

8 Related Work

Our work is mostly related to Hu et al. [2023], who mechanized an NbE algorithm for MLTT with modal types and a full universe hierarchy in Agda. They adjusted models used by Abel [2013]'s pen and paper proof, so that the universe levels are explicitly maintained and a limit-taking operation of universe levels is no longer needed. They also altered several definitions in order to work with the proof relevance and termination checking in Agda. Our work refines their PER model to contain precise universe level information, enabling reasoning for non-cumulativity. The remaining of this section focuses on related works about NbE and mechanized dependent type systems.

8.1 Normalization by Evaluation

We start with other works on formalized NbE for dependent type systems. Wieczorek and Biernacki [2018] mechanized a similar-style NbE algorithm for MLTT with one universe in Rocq by universally quantifying over the impredicative `Prop` to interpret dependent function types. On the other hand, intrinsically-typed NbE for dependent type systems is challenging. Danielsson [2006] first formalized an NbE algorithm for dependent type systems in AgdaLight. However, he did not cover large elimination and his formalization was later identified to use non-positive predicates [Chapman 2008]. One recent advance was done by Altenkirch and Kaposi [2016a], where they make use of quotient inductive inductive types (QIIT) [Altenkirch and Kaposi 2016c] to represent syntax of well-typed terms. Their system was still quite simple, with no support of inductive types, universe hierarchy or large elimination. Another concern about their formalization is that the meta-theory of QIIT itself remains to be established. Sterling [2022] describes a dependent type theory with a non-cumulative hierarchy and an NbE algorithm in a presheaf formulation. As opposed to our observation in Sec. 6 where cumulativity induces simpler formulation, non-cumulativity is more natural for a presheaf formulation. In fact, Coquand [2018] include extra structures to achieve cumulativity intrinsically. Therefore, which hierarchy is more complex depends on the style of the NbE algorithm. Our understanding of this situation is that an untyped domain model does not keep track of universe level information, so its models necessarily need to maintain more accurate universe levels in non-cumulativity than in cumulativity. Whereas in intrinsically typed syntax, universe levels are encoded as part of the syntax, so this information essentially induces no cost.

Achieving cumulativity contrarily requires some "relaxation" structure in the universe levels. For more sophisticated systems like Calculus of Inductive Constructions (CIC), correctness of NbE (in any style) remains open. Other than dependent type systems, NbE has been investigated on type systems including System F [Altenkirch et al. 1995] and System F^{ω} [Abel 2009].

8.2 Mechanization of Dependent Type Systems

A central focus of mechanized dependent type systems is still the normalization proof. Early work, such as that by Barras and Werner [1997], demonstrated strong normalization for the calculus of constructions in Rocq using reducibility candidates. Anand and Rahli [2014] mechanized the metatheory of a Nuprl-like type system in Rocq using a PER model to directly establish its consistency relative to Rocq's consistency. Abel et al. [2018] mechanized the decidability proof of conversion checking for the Martin-Löf Type Theory with dependent functions, natural numbers and one universe. Their algorithm first reduces terms to weak head normal forms and is sound and complete with respect to standard equivalence rules. Pujet and Tabareau [2022] extended the work of Abel et al. [2018] by mechanizing observational equality and a two-level cumulative universe hierarchy. This approach was further refined by Pujet and Tabareau [2023] when mechanizing impredicative observational equality. Adjedj et al. [2024] proved normalization of a Rocq's predicative fragment with dependent sum, dependent product, identity type, but one universe level. They proved both normalization and decidability of bidirectional type checking. Liu et al. [2025] formalized the normalization of a dependent type system with indistinguishability for dependency tracking in Rocq. Their system also supports a full cumulative universe hierarchy with several other features. Some other mechanizations proved lighter properties than normalization. Sozeau et al. [2019] formalized the type-checking and erasure of Rocq in Rocq by assuming the correctness of Rocq's metatheory. Liesnikov and Cockx [2024] formalized a subset of Agda's syntax and a sound type-checker that ensures type safety (but not logical soundness).

9 Conclusion

This work explores normalization by evaluation using an untyped domain model in MLTT with a non-cumulative universe hierarchy. Previous works put strong emphasis on applying untyped NbE to cumulative universe hierarchies, whereas practical proof assistants like Agda and Lean are non-cumulative by default. In this work, we prove that untyped NbE works for non-cumulativity. We establish this conclusion in two steps. First, we work with a system with explicit universe level annotations. These annotations help keep track of the universe levels both in syntax and in semantics such that our domain construction correctly reflects the uniqueness of universe levels. After establishing the completeness and soundness of NbE with these annotations, we take another step to show that they are logically redundant, yielding a system and an NbE algorithm that are closer to real practice.

Our work closes the theoretical gap of applying untyped NbE in non-cumulative settings by focusing on isolating the key trade-offs between cumulative and non-cumulative universe hierarchies. We expect that it shall not be too difficult to extend our work to support dependent sums, a false type and a unit type. We hope that future work can build on top of our work to bridge the gap in terms of features to practical proof assistants like Agda or Lean that also adopt a non-cumulative universe hierarchy. Practical proof assistants incorporate much richer type-theoretic features and more implementation optimization, including but not limited to custom inductive types [Sozeau et al. 2019], universe polymorphism [Bezem et al. 2022; Sozeau and Tabareau 2014], termination-checking [Abel 2024] instead of recursor based recursion, and efficient conversion checking (often via weak head normalization [Abel et al. 2018]). Studying and mechanizing these features and their interactions are all interesting but challenging, both theoretically and technically.

Acknowledgments

The authors thank Xuejing Huang and Xu Xue for their suggestions on paper writing and proofreading and all the anonymous reviewers for their suggestions on improving the paper. Jason Z. S. Hu was funded partly by Postgraduate Scholarship - Doctoral from the Natural Sciences and Engineering Research Council of Canada and partly by Doctoral (B2X) Research Scholarship from Fonds de recherche du Québec - Nature et technologies during his Ph.D. study.

References

Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. 1991. Explicit Substitutions. *J. Funct. Program.* 1, 4 (1991), 375–416. doi:10.1017/S0956796800000186

Andreas Abel. 2009. Typed Applicative Structures and Normalization by Evaluation for System F^{omega}. In *Computer Science Logic, 23rd international Workshop, CSL 2009, 18th Annual Conference of the EACSL, Coimbra, Portugal, September 7-11, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5771)*, Erich Grädel and Reinhard Kahle (Eds.). Springer, 40–54. doi:10.1007/978-3-642-04027-6_6

Andreas Abel. 2013. *Normalization by Evaluation: Dependent Types and Impredicativity.* Habilitation Thesis. Ludwig-Maximilians-Universität München, Munich, Germany. https://www.cse.chalmers.se/~abela/habil.pdf

Andreas Abel. 2024. foetus - Termination Checker for Simple Functional Programs. *CoRR* abs/2407.06924 (2024). doi:10.48550/ARXIV.2407.06924 arXiv:2407.06924

Andreas Abel, Joakim Öhman, and Andrea Vezzosi. 2018. Decidability of Conversion for Type Theory in Type Theory. *Proc. ACM Program. Lang.* 2, POPL (2018), 23:1–23:29. doi:10.1145/3158111

Andreas Abel, Andrea Vezzosi, and Théo Winterhalter. 2017. Normalization by Evaluation for Sized Dependent Types. *Proc. ACM Program. Lang.* 1, ICFP (2017), 33:1–33:30. doi:10.1145/3110277

Arthur Adjedj, Meven Lennon-Bertrand, Kenji Maillard, Pierre-Marie Pédrot, and Loïc Pujet. 2024. Martin-Löf à la Coq. In *Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2024, London, UK, January 15-16, 2024*, Amin Timany, Dmitriy Traytel, Brigitte Pientka, and Sandrine Blazy (Eds.). ACM, 230–245. doi:10.1145/3636501.3636951

Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. 1995. Categorical Reconstruction of a Reduction Free Normalization Proof. In *Proceedings of the 6th International Conference on Category Theory and Computer Science, CTCS 1995, Cambridge, UK, August 7-11, 1995 (Lecture Notes in Computer Science, Vol. 953)*, David H. Pitt, David E. Rydeheard, and Peter T. Johnstone (Eds.). Springer, 182–199. doi:10.1007/3-540-60164-3_27

Thorsten Altenkirch and Ambrus Kaposi. 2016a. Normalisation by Evaluation for Dependent Types. In *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, Porto, Portugal, June 22-26, 2016 (LIPIcs, Vol. 52)*, Delia Kesner and Brigitte Pientka (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:16. doi:10.4230/LIPICS.FSCD.2016.6

Thorsten Altenkirch and Ambrus Kaposi. 2016b. Normalisation by Evaluation for Dependent Types. In *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, Porto, Portugal, June 22-26, 2016 (LIPIcs, Vol. 52)*, Delia Kesner and Brigitte Pientka (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:16. https://doi.org/10.4230/LIPIcs.FSCD.2016.6

Thorsten Altenkirch and Ambrus Kaposi. 2016c. Type Theory in Type Theory Using Quotient Inductive Types. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, Florida, USA, January 20-22, 2016*, Rastislav Bodík and Rupak Majumdar (Eds.). ACM, 18–29. doi:10.1145/2837614.2837638

Thorsten Altenkirch and Ambrus Kaposi. 2017. Normalisation by Evaluation for Type Theory, in Type Theory. *Log. Methods Comput. Sci.* 13, 4 (2017). doi:10.23638/LMCS-13(4:1)2017

Abhishek Anand and Vincent Rahli. 2014. Towards a Formally Verified Proof Assistant. In *Interactive Theorem Proving - 5th International Conference, ITP 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8558)*, Gerwin Klein and Ruben Gamboa (Eds.). Springer, 27–44. doi:10.1007/978-3-319-08970-6_3

Bruno Barras and Benjamin Werner. 1997. Coq in Coq. https://www.lix.polytechnique.fr/Labo/Bruno.Barras/publi/coqincoq.pdf Unpublished manuscript.

Marc Bezem, Thierry Coquand, Peter Dybjer, and Martín Escardó. 2022. Type Theory with Explicit Universe Polymorphism. In *28th International Conference on Types for Proofs and Programs, TYPES 2022, LS2N, University of Nantes, France, June 20-25, 2022 (LIPIcs, Vol. 269)*, Delia Kesner and Pierre-Marie Pédrot (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 13:1–13:16. doi:10.4230/LIPICS.TYPES.2022.13

James Chapman. 2008. Type Theory Should Eat Itself. In *Proceedings of the International Workshop on Logical Frameworks and Metalanguages: Theory and Practice, LFMTP@LICS 2008, Pittsburgh, Pennsylvania, USA, June 23, 2008 (Electronic Notes* *in Theoretical Computer Science, Vol. 228)*, Andreas Abel and Christian Urban (Eds.). Elsevier, 21–36. doi:10.1016/J.ENTCS.2008.12.114

Arthur Charguéraud. 2012. The Locally Nameless Representation. *Journal of Automated Reasoning* 49, 3 (01 Oct 2012), 363–408. doi:10.1007/s10817-011-9225-2

Thierry Coquand. 2018. Canonicity and normalisation for Dependent Type Theory. *CoRR* abs/1810.09367 (2018). arXiv:1810.09367 http://arxiv.org/abs/1810.09367

Nils Anders Danielsson. 2006. A Formalisation of a Dependently Typed Language as An Inductive-Recursive Family. In *International Workshop on Types for Proofs and Programs, TYPES 2006, Nottingham, UK, April 18-21, 2006, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 4502)*, Thorsten Altenkirch and Conor McBride (Eds.). Springer, 93–109. doi:10.1007/978-3-540-74464-1_7

Leonardo de Moura and Sebastian Ullrich. 2021. The Lean 4 Theorem Prover and Programming Language. In *Proceedings of the 28th International Conference on Automated Deduction, CADE 2021, Virtual Event, July 12-15, 2021 (Lecture Notes in Computer Science, Vol. 12699)*, André Platzer and Geoff Sutcliffe (Eds.). Springer, 625–635. doi:10.1007/978-3-030-79876-5_37

Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The Lean Theorem Prover (System Description). In *Proceedings of the 25th International Conference on Automated Deduction, CADE 2015, Berlin, Germany, August 1-7, 2015 (Lecture Notes in Computer Science, Vol. 9195)*, Amy P. Felty and Aart Middeldorp (Eds.). Springer, 378–388. doi:10.1007/978-3-319-21401-6_26

Peter Dybjer. 2000. A General Formulation of Simultaneous Inductive-Recursive Definitions in Type Theory. *J. Symb. Log.* 65, 2 (2000), 525–549. doi:10.2307/2586554

Francisco Ferreira and Brigitte Pientka. 2014. Bidirectional Elaboration of Dependently Typed Programs. In *Proceedings of the 16th International Symposium on Principles and Practice of Declarative Programming, Kent, Canterbury, United Kingdom, September 8-10, 2014*, Olaf Chitil, Andy King, and Olivier Danvy (Eds.). ACM, 161–174. doi:10.1145/2643135.2643153

Daniel Fridlender and Miguel Pagano. 2013. A Type-Checking Algorithm for Martin-Löf Type Theory with Subtyping Based on Normalisation by Evaluation. In *Typed Lambda Calculi and Applications, 11th International Conference, TLCA 2013, Eindhoven, The Netherlands, June 26-28, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 7941)*, Masahito Hasegawa (Ed.). Springer, 140–155. doi:10.1007/978-3-642-38946-7_12

Daniel Gratzer, Jonathan Sterling, and Lars Birkedal. 2019. Implementing A Modal Dependent Type Theory. *Proc. ACM Program. Lang.* 3, ICFP (2019), 107:1–107:29. doi:10.1145/3341711

Adam Michael Gundry. 2013. *Type inference, Haskell and dependent types*. Ph. D. Dissertation. University of Strathclyde, Glasgow, UK. http://oleg.lib.strath.ac.uk/R/?func=dbin-jump-full&object_id=22728

Robert Harper and Frank Pfenning. 2005. On equivalence and canonical forms in the LF type theory. *ACM Trans. Comput. Log.* 6, 1 (2005), 61–101. doi:10.1145/1042038.1042041

Jason Z. S. Hu, Junyoung Jang, and Brigitte Pientka. 2023. Normalization by Evaluation for Modal Dependent Type Theory. *J. Funct. Program.* 33 (2023). doi:10.1017/S0956796823000060

Jason Z. S. Hu and Brigitte Pientka. 2023. Layered Modal Type Theories. *CoRR* abs/2305.06548 (2023). arXiv:2305.06548 https://doi.org/10.48550/arXiv.2305.06548

Junyoung Jang, Jason Z. S. Hu, Antoine Gaulin, and Brigitte Pientka. 2025. McTT: Building A Correct-By-Construction Proof Checker For Martin-Löf Type Theory. (2025). In the fourth Workshop on the Implementation of Type Systems (WITS 2025).

Shengyi Jiang, Jason Z. S. Hu, and Bruno C. d. S. Oliveira. 2025. *Normalization by Evaluation for Non-cumulativity (Artifact)*. doi:10.5281/zenodo.15686111

P. J. Landin. 1964. The Mechanical Evaluation of Expressions. *Comput. J.* 6, 4 (1964), 308–320. doi:10.1093/COMJNL/6.4.308

Meven Lennon-Bertrand. 2025. What does it take to certify conversion? arXiv:2502.15500 [cs.PL] https://arxiv.org/abs/2502.15500

Xavier Leroy, Sandrine Blazy, Daniel Kästner, Bernhard Schommer, Markus Pister, and Christian Ferdinand. 2016. CompCert - A Formally Verified Optimizing Compiler. In *8th European Congress on Embedded Real Time Software and Systems, ERTS 2016, Toulouse, France, January, 2016.* https://inria.hal.science/hal-01238879

Bohdan Liesnikov and Jesper Cockx. 2024. Building a Correct-by-Construction Type Checker for a Dependently Typed Core Language. In *Programming Languages and Systems - 22nd Asian Symposium, APLAS 2024, Kyoto, Japan, October 22-24, 2024, Proceedings (Lecture Notes in Computer Science, Vol. 15194)*, Oleg Kiselyov (Ed.). Springer, 63–83. doi:10.1007/978-981-97-8943-6_4

Yiyun Liu, Jonathan Chan, and Stephanie Weirich. 2025. Consistency of a Dependent Calculus of Indistinguishability. *Proc. ACM Program. Lang.* 9, POPL (2025), 183–209. doi:10.1145/3704843

Conor McBride. 2000. Elimination with a Motive. In *Types for Proofs and Programs, International Workshop, TYPES 2000, Durham, UK, December 8-12, 2000, Selected Papers (Lecture Notes in Computer Science, Vol. 2277)*, Paul Callaghan, Zhaohui Luo, James McKinna, and Robert Pollack (Eds.). Springer, 197–216. doi:10.1007/3-540-45842-5_13

Erik Palmgren. 1998. On Universes in Type Theory. In *Twenty Five Years of Constructive Type Theory*. Oxford University Press. doi:10.1093/oso/9780198501275.003.0012

François Pottier. 2014. Hindley-milner elaboration in applicative style: functional pearl. In *Proceedings of the 19th ACM SIGPLAN international conference on Functional programming, Gothenburg, Sweden, September 1-3, 2014*, Johan Jeuring and Manuel M. T. Chakravarty (Eds.). ACM, 203–212. doi:10.1145/2628136.2628145

Loïc Pujet and Nicolas Tabareau. 2022. Observational Equality: Now for Good. *Proc. ACM Program. Lang.* 6, POPL (2022), 1–27. doi:10.1145/3498693

Loïc Pujet and Nicolas Tabareau. 2023. Impredicative Observational Equality. *Proc. ACM Program. Lang.* 7, POPL (2023), 2171–2196. doi:10.1145/3571739

Matthieu Sozeau, Simon Boulier, Yannick Forster, Nicolas Tabareau, and Théo Winterhalter. 2019. Coq Coq correct! verification of type checking and erasure for Coq, in Coq. *Proc. ACM Program. Lang.* 4, POPL, Article 8 (Dec. 2019), 28 pages. doi:10.1145/3371076

Matthieu Sozeau and Nicolas Tabareau. 2014. Universe Polymorphism in Coq. In *Interactive Theorem Proving - 5th International Conference, ITP 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8558)*, Gerwin Klein and Ruben Gamboa (Eds.). Springer, 499–514. doi:10.1007/978-3-319-08970-6_32

Jonathan Sterling. 2022. *First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory*. PhD Thesis. Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. https://doi.org/10.1184/r1/19632681.v1

William W. Tait. 1967. Intensional Interpretations of Functionals of Finite Type I. *J. Symb. Log.* 32, 2 (1967), 198–212. doi:10.2307/2271658

The Agda Team. 2024. Agda 2.6.4.3. https://wiki.portal.chalmers.se/agda/pmwiki.php

The Coq Development Team. 2024. *The Coq Proof Assistant*. doi:10.5281/zenodo.14542673

The Mathlib Community. 2020. The Lean Mathematical Library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, Louisiana, USA, January 20-21, 2020*, Jasmin Blanchette and Catalin Hritcu (Eds.). ACM, 367–381. doi:10.1145/3372885.3373824

Pawel Wieczorek and Dariusz Biernacki. 2018. A Coq Formalization of Normalization by Evaluation for Martin-Löf Type Theory. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, California, USA, January 8-9, 2018*, June Andronick and Amy P. Felty (Eds.). ACM, 266–279. doi:10.1145/3167091

A Complete Rules

Fig. 8 and 9 show the complete rules of term and type equivalence. Fig. 10 shows the complete rules of substitution equivalence. Fig. 11 shows the rules of context equivalence.

$\boxed{\Gamma \vdash t \equiv s :^{i} T}$ — t and s of type T are equivalent at level i under context Γ

$$\frac{\Gamma \vdash S :^{1+i} \mathsf{Set}_i \quad \Gamma, x : S^{i} \vdash T :^{1+j} \mathsf{Set}_j \quad \Gamma, x : S^{i} \vdash t :^{j} T \quad \Gamma \vdash s :^{i} S}{\Gamma \vdash (\lambda(x : S^{i}).t)\ s \equiv t[s : S^{i}] :^{j} T[s : S^{i}/x]} \qquad \frac{\Gamma \vdash t :^{i} T}{\Gamma \vdash \mathtt{unlift}(\mathtt{lift}_j\ t) \equiv t :^{i} T}$$

$$\frac{\begin{array}{c}\Gamma, z : \mathsf{N}^{0} \vdash T :^{1+i} \mathsf{Set}_i \\ \Gamma \vdash r :^{i} T[\mathtt{0} : \mathsf{N}^{0}/z] \qquad \Gamma, x : \mathsf{N}^{0}, y : (T[x_1 : \mathsf{N}^{0}/z])^{i} \vdash s :^{i} T[(\uparrow \circ \uparrow), \mathsf{suc}\ x_1 : \mathsf{N}^{0}/z]\end{array}}{\Gamma \vdash \mathsf{rec}(z.T^{i})\ r\ (x, y.s)\ \mathtt{0} \equiv s :^{i} T[\mathtt{0} : \mathsf{N}^{0}/z]}$$

$$\frac{\begin{array}{c}\Gamma, z : \mathsf{N}^{0} \vdash T :^{1+i} \mathsf{Set}_i \qquad \Gamma \vdash r :^{i} T[\mathtt{0} : \mathsf{N}^{0}/z] \\ \Gamma, x : \mathsf{N}^{0}, y : (T[x_1 : \mathsf{N}^{0}/z])^{i} \vdash s :^{i} T[(\uparrow \circ \uparrow), \mathsf{suc}\ x_1 : \mathsf{N}^{0}/z] \qquad \Gamma \vdash t :^{0} \mathsf{N}\end{array}}{\Gamma \vdash \mathsf{rec}(z.T^{i})\ r\ (x, y.s)\ (\mathsf{suc}\ t) \equiv s[t : \mathsf{N}^{0}/x, (\mathsf{rec}(z.T^{i})\ r\ (x, y.s)\ t)^{i}/y] :^{i} T[\mathsf{suc}\ t : \mathsf{N}^{0}/z]}$$

$$\frac{\Gamma \vdash S :^{1+i} \mathsf{Set}_i \quad \Gamma, x : S^{i} \vdash T :^{1+j} \mathsf{Set}_j \quad \Gamma \vdash t :^{\max(i,j)} \Pi(x : S^{i}).T^{j}}{\Gamma \vdash t \equiv \lambda(x : S^{i}).(t[\uparrow])\ x :^{\max(i,j)} \Pi(x : S^{i}).T^{j}}$$

$$\frac{\Gamma \vdash T :^{1+i} \mathsf{Set}_i \quad \Gamma \vdash t :^{j+i} \mathtt{Lift}_j\ T^{i}}{\Gamma \vdash t \equiv \mathtt{lift}_j\ (\mathtt{unlift}\ t) :^{j+i} \mathtt{Lift}_j\ T^{i}}$$

$$\frac{\Gamma \vdash S :^{1+i} \mathsf{Set}_i \quad \Gamma \vdash S \equiv S' :^{1+i} \mathsf{Set}_i \quad \Gamma, S^{i} \vdash T \equiv T' :^{1+j} \mathsf{Set}_j}{\Gamma \vdash \Pi(x : S^{i}).T^{j} \equiv \Pi(x : S'^{i}).T'^{j} :^{1+\max(i,j)} \mathsf{Set}_{\max(i,j)}}$$

$$\frac{\Gamma \vdash T \equiv T' :^{1+i} \mathsf{Set}_i}{\Gamma \vdash \mathtt{Lift}_j\ T^{i} \equiv \mathtt{Lift}_j\ T'^{i} :^{1+j+i} \mathsf{Set}_{j+i}} \qquad \frac{\vdash \Gamma \quad x : T^{i} \in \Gamma}{\Gamma \vdash x \equiv x :^{i} T}$$

$$\frac{\Gamma \vdash A :^{1+i} \mathsf{Set}_i \quad \Gamma, S^{i} \vdash T :^{1+j} \mathsf{Set}_j \quad \Gamma \vdash s \equiv s' :^{\max(i,j)} \Pi(S :^{i}).T^{j} \quad \Gamma \vdash t \equiv t' :^{i} S}{\Gamma \vdash s\ t \equiv s'\ t' :^{j} T\ [s : S^{i}]}$$

$$\frac{\Gamma \vdash S :^{1+i} \mathsf{Set}_i \quad \Gamma \vdash S \equiv S' :^{1+j} \mathsf{Set}_j \quad \Gamma, x : S^{i} \vdash t \equiv t' :^{j} T}{\Gamma \vdash \lambda(x : S^{i}).t \equiv \lambda(x : S'^{i}).t' :^{\max(i,j)} \Pi(S :^{i}).T^{j}} \qquad \frac{\Gamma \vdash \sigma \equiv \sigma' : \Delta \quad \Delta \vdash t \equiv t' :^{i} T}{\Gamma \vdash t[\sigma] \equiv t[\sigma'] :^{i} T[\sigma]}$$

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathtt{0} \equiv \mathtt{0} :^{0} \mathsf{N}} \qquad \frac{\Gamma \vdash t \equiv t' :^{0} \mathsf{N}}{\Gamma \vdash \mathsf{suc}\ t \equiv \mathsf{suc}\ t' :^{0} \mathsf{N}}$$

$$\frac{\begin{array}{c}\Gamma, z : \mathsf{N}^{0} \vdash T :^{1+i} \mathsf{Set}_i \quad \Gamma, z : \mathsf{N}^{0} \vdash T \equiv T' :^{1+i} \mathsf{Set}_i \quad \Gamma \vdash r \equiv r' :^{i} T[\mathtt{0} : \mathsf{N}^{0}/z] \\ \Gamma, x : \mathsf{N}^{0}, y : T^{i} \vdash s \equiv s' :^{i} T[(\uparrow \circ \uparrow), \mathsf{suc}\ x_1 : \mathsf{N}^{0}] \quad \Gamma \vdash t \equiv t' :^{i} \mathsf{N}\end{array}}{\Gamma \vdash \mathsf{rec}\ (z.T^{i})\ r\ (x, y.s)\ t \equiv \mathsf{rec}\ (z.T'^{i})\ r'\ (x, y.s')\ t' :^{i} T[t : \mathsf{N}^{0}]}$$

$$\frac{\Gamma \vdash t \equiv t' :^{i} T}{\Gamma \vdash \mathtt{lift}_j\ t \equiv \mathtt{lift}_j\ t' :^{j+i} \mathtt{Lift}_j\ T^{i}} \qquad \frac{\Gamma \vdash T :^{1+i} \mathsf{Set}_i \quad \Gamma \vdash t \equiv t' :^{j+i} \mathtt{Lift}_j\ T^{i}}{\Gamma \vdash \mathtt{unlift}\ t \equiv \mathtt{unlift}\ t' :^{i} T}$$

$$\frac{\Gamma \vdash \sigma : \Delta}{\Gamma \vdash \mathsf{N}[\sigma] \equiv \mathsf{N} :^{1} \mathsf{Set}_0} \qquad \frac{\Gamma \vdash \sigma : \Delta}{\Gamma \vdash \mathsf{Set}_i[\sigma] \equiv \mathsf{Set}_i :^{2+i} \mathsf{Set}_{1+i}} \qquad \frac{\Gamma \vdash \sigma : \Delta \quad \Delta \vdash t :^{0} \mathsf{N}}{\Gamma \vdash (\mathsf{suc}\ t)[\sigma] \equiv \mathsf{suc}\ (t[\sigma]) :^{0} \mathsf{N}}$$

$$\frac{\Gamma \vdash \sigma : \Delta}{\Gamma \vdash \mathtt{0}[\sigma] \equiv \mathtt{0} :^{0} \mathsf{N}} \qquad \frac{\Gamma \vdash \sigma : \Delta \quad \Delta \vdash A :^{1+i} \mathsf{Set}_i}{\Gamma \vdash (\mathtt{Lift}_j\ T^{i})[\sigma] \equiv \mathtt{Lift}_j\ (T[\sigma])^{i} :^{1+j+i} \mathsf{Set}_{j+i}}$$

Fig. 8. Term equivalence rules (part 1 of 2).

$$\frac{\Gamma \vdash \sigma : \Delta \quad \Delta \vdash S :^{1+i} \mathtt{Set}_i \quad \Delta, S^i \vdash T :^{1+j} \mathtt{Set}_j}{\Gamma \vdash (\Pi(x : S^i).T^j)[\sigma] \equiv \Pi(x : (S[\sigma])^i).(T[\mathsf{q}\, S^i \sigma])^j :^{1+\max(i,j)} \mathtt{Set}_{\max(i,j)}}$$

$$\frac{\begin{array}{c}\Gamma \vdash \sigma : \Delta \quad \Delta, z : \mathsf{N}^0 \vdash T :^{1+i} \mathtt{Set}_i \quad \Delta \vdash s :^i T[\mathtt{0} : \mathsf{N}^0/z] \\ \Delta, x : \mathsf{N}^0, y : (T[x_1 : \mathsf{N}^0/z])^i \vdash r :^i T[(\uparrow \circ \uparrow), \mathsf{suc}\ x_1 : \mathsf{N}^0] \quad \Delta \vdash t :^0 \mathsf{N}\end{array}}{\Gamma \vdash (\mathtt{rec}\ (z.T^i)\ r\ (x, y.s)\ t)[\sigma] \equiv \mathtt{rec}\ (z.T[\mathsf{q}\, \mathsf{N}^0 \sigma])^i\ (r[\sigma])\ (x, y.s[\mathsf{q}\, T^i (\mathsf{q}\, \mathsf{N}^0 \sigma)])\ (t[\sigma]) :^i T[\sigma,\ t[\sigma] : \mathsf{N}^0/z]}$$

$$\frac{\Gamma \vdash t :^i T}{\Gamma \vdash t \equiv t[\mathtt{Id}] :^i T} \qquad \frac{x_n : T^i \in \Gamma \quad \Gamma \vdash S :^{1+j} \mathtt{Set}_j}{\Gamma, S^j \vdash x_n[\uparrow] \equiv x_{1+n} :^i T\ [\uparrow]} \qquad \frac{\Gamma \vdash \tau : \Delta \quad \Delta \vdash \sigma : \Psi \quad \Psi \vdash t :^i T}{\Gamma \vdash t[\sigma][\tau] \equiv t[\sigma \circ \tau] :^i T[\sigma \circ \tau]}$$

$$\frac{\Gamma \vdash \sigma : \Delta \quad \Delta \vdash T :^{1+i} \mathtt{Set}_i \quad \Gamma \vdash t :^i T[\sigma]}{\Gamma \vdash x_0[\sigma, t : T^i] \equiv t :^i T[\sigma]} \qquad \frac{\Gamma \vdash \sigma : \Delta \quad \Delta \vdash T :^{1+i} \mathtt{Set}_i \quad \Gamma \vdash t :^i T[\sigma] \quad x_n : S^j \in \Delta}{\Gamma \vdash x_{1+n}[\sigma, t : T^i] \equiv x_n[\sigma] :^j S[\sigma]}$$

$$\frac{\Gamma \vdash t :^i T}{\Gamma \vdash t \equiv t :^i T} \qquad \frac{\Gamma \vdash t \equiv s :^i T}{\Gamma \vdash s \equiv t :^i T} \qquad \frac{\Gamma \vdash t \equiv s :^i T \quad \Gamma \vdash s \equiv r :^i T}{\Gamma \vdash t \equiv r :^i T}$$

Fig. 9. Term equivalence rules (part 2 of 2).

$\boxed{\Gamma \vdash \sigma \equiv \tau : \Delta}$ σ and τ having codomain context Δ are equivalent substitutions under Γ

$$\frac{\Gamma \vdash \sigma : \Delta \quad \Delta \vdash T :^{1+i} \mathtt{Set}_i \quad \Gamma \vdash t :^i T^\sigma}{\Gamma \vdash \uparrow \circ (\sigma, t : T^i/x_0) \equiv \sigma : \Delta} \qquad \frac{\Gamma \vdash \sigma : \Gamma, T^i}{\Gamma \vdash \sigma \equiv (\uparrow \circ \sigma, x_0[\sigma] : T^i/x_0) : \Gamma, T^i}$$

$$\frac{\Gamma \vdash \tau : \Delta \quad \Delta \vdash \sigma : \Psi \quad \Psi \vdash T :^{1+i} \mathtt{Set}_i \quad \Delta \vdash t :^i T[\sigma]}{\Gamma \vdash (\sigma, t : T^i/x_0) \circ \tau \equiv \sigma \circ \tau,\ (t[\tau] : T^i/x_0) : \Psi, T^i} \qquad \frac{\Gamma \vdash \sigma : \Delta}{\Gamma \vdash \mathtt{Id} \circ \sigma \equiv \sigma : \Delta}$$

$$\frac{\Gamma \vdash \sigma : \Delta}{\Gamma \vdash \sigma \circ \mathtt{Id} \equiv \sigma : \Delta} \qquad \frac{\vdash \Gamma}{\Gamma \vdash \mathtt{Id} \equiv \mathtt{Id} : \Gamma} \qquad \frac{\vdash \Gamma, T^i}{\Gamma, T^i \vdash \uparrow \equiv \uparrow : \Gamma} \qquad \frac{\Gamma \vdash \sigma \equiv \sigma' : \Delta \quad \Delta \vdash \tau \equiv \tau' : \Psi}{\Gamma \vdash \tau \circ \sigma \equiv \tau' \circ \sigma' : \Psi}$$

$$\frac{\Gamma \vdash \sigma \equiv \sigma' : \Delta \quad \Delta \vdash T :^{1+i} \mathtt{Set}_i \quad \Delta \vdash T \equiv T' :^{1+i} \mathtt{Set}_i \quad \Gamma \vdash \tau \equiv \tau' : \Delta}{\Gamma \vdash \sigma, \tau : T^i/x_0 \equiv \sigma', \tau' : T'^i/x_0 : (\Delta, T^i)}$$

$$\frac{\Gamma \vdash \gamma : \Delta \quad \Delta \vdash \tau : \Psi \quad \Psi \vdash \sigma : \Psi'}{\Gamma \vdash (\sigma \circ \tau) \circ \gamma \equiv \sigma \circ (\tau \circ \gamma) : \Psi'}$$

$$\frac{\Gamma \vdash \sigma : \Delta}{\Gamma \vdash \sigma \equiv \sigma : \Delta} \qquad \frac{\Gamma \vdash \sigma \equiv \tau : \Delta}{\Gamma \vdash \tau \equiv \sigma : \Delta} \qquad \frac{\Gamma \vdash \sigma \equiv \tau : \Delta \quad \Gamma \vdash \tau \equiv \gamma : \Delta}{\Gamma \vdash \sigma \equiv \gamma : \Delta}$$

Fig. 10. Substitution equivalence rules.

$\boxed{\vdash \Gamma \equiv \Delta}$ Γ and Δ are equivalent

$$\frac{}{\vdash \cdot \equiv \cdot} \qquad \frac{\vdash \Gamma \equiv \Delta \quad \Gamma \vdash T :^{1+i} \mathtt{Set}_i \quad \Delta \vdash T' :^{1+i} \mathtt{Set}_i \quad \Gamma \vdash T \equiv T' :^{1+i} \mathtt{Set}_i \quad \Delta \vdash T \equiv T' :^{1+i} \mathtt{Set}_i}{\vdash \Gamma, x : T^i \equiv \Delta, x : T'^i}$$

Fig. 11. Context equivalence rules.

$$
\begin{array}{llll}
\text{De Bruijn Indices} & n \in \mathbb{N} \quad \text{Variable Names} \quad x, y, z & \text{Universe Levels} \quad i, j, k \in \mathbb{N} & \\
\text{Terms, Types} & r, s, t, R, S, T & ::= & x_n \mid \lambda(x : S).t \mid t\ s \mid \mathtt{0} \mid \mathtt{suc}\ t \mid \mathtt{rec}(x.T)\ r\ (x, y.s)\ t \mid \\
& & & \mathtt{lift}_j\ t \mid \mathtt{unlift}\ t \mid t[\sigma] \mid \\
& & & \mathtt{Set}_i \mid \Pi(x : S).T \mid \mathtt{N} \mid \mathtt{Lift}_j\ T \\
\text{Substitutions} & \sigma, \tau, \gamma & ::= & \mathtt{Id} \mid \uparrow \mid \sigma, t : T/x_0 \mid \sigma \circ \tau \\
\text{Normal Forms} & v, w, V, W & ::= & u \mid \lambda(x : W).w \mid \mathtt{0} \mid \mathtt{suc}\ v \mid \mathtt{lift}_j\ v \mid \mathtt{Set}_i \mid \\
& & & \Pi(x : V).W \mid \mathtt{N} \mid \mathtt{Lift}_j\ V \\
\text{Neutral Forms} & u & ::= & x_n \mid u\ v \mid \mathtt{rec}(x.W)\ v_z\ (x, y.v_s)\ u \mid \mathtt{unlift}\ u \\
\text{Contexts} & \Gamma, \Delta, \Psi & ::= & \cdot \mid \Gamma, x : T
\end{array}
$$

Fig. 12. Syntax of the unascribed system.

$$
\begin{array}{llll}
\text{Environment} & \rho, \phi, \theta \in \mathbb{N} \to \mathrm{D} & & \\
\text{Semantic Values (D)} & a, b, c, f, & ::= & (\!|\boldsymbol{\lambda} x.t|\!)_\rho \mid \mathbf{0} \mid \mathbf{suc}\ d \mid \mathbf{lift}_i\ a \mid \mathbf{Set}_i \mid \\
& A, B, F & & (\!|\mathbf{\Pi}\ A\ (x.T)|\!)_\rho \mid \mathbf{N} \mid \mathbf{Lift}_j\ A \mid \uparrow_A e \\
\text{Neutral Semantic Values } (\mathrm{D}^{\mathrm{ne}}) & e, E & ::= & \mathbf{x}_k \mid e\ d \mid (\!|\mathbf{rec}\ (z.T)\ a\ (x, y.s)\ e|\!)_\rho \mid \mathbf{unlift}\ e \\
\text{Normal Semantic Values } (\mathrm{D}^{\mathrm{nf}}) & d, D & ::= & \downarrow_A a
\end{array}
$$

Fig. 13. Domain values of the unascribed system.

$\boxed{[\![t]\!]_\rho \searrow a}$ t evaluates to a under ρ

$$
\frac{}{[\![\mathtt{N}]\!]_\rho \searrow \mathbf{N}}
\qquad
\frac{[\![S]\!]_\rho \searrow A}{[\![\Pi(x : S).T]\!]_\rho \searrow (\!|\mathbf{\Pi}\ A\ T|\!)_\rho}
\qquad
\frac{[\![T]\!]_\rho \searrow V}{[\![\mathtt{Lift}_j\ T]\!]_\rho \searrow \mathbf{Lift}_j\ a}
\qquad
\frac{}{[\![\mathtt{Set}_i]\!]_\rho \searrow \mathbf{Set}_i}
$$

$$
\frac{}{[\![x_n]\!]_\rho \searrow \rho(n)}
\qquad
\frac{}{[\![\mathtt{0}]\!]_\rho \searrow \mathbf{0}}
\qquad
\frac{[\![t]\!]_\rho \searrow a}{[\![\mathtt{suc}\ t]\!]_\rho \searrow \mathbf{suc}\ a}
$$

$$
\frac{[\![s]\!]_\rho \searrow a \qquad [\![t]\!]_\rho \searrow b \qquad \mathrm{rec}\cdot(z.T, a, x, y.r, b, \rho) \searrow c}{[\![\mathtt{rec}\ z.T\ s\ x, y.r\ t]\!]_\rho \searrow c}
\qquad
\frac{}{[\![\lambda(x : S).t]\!]_\rho \searrow (\!|\boldsymbol{\lambda} t|\!)_\rho}
$$

$$
\frac{[\![s]\!]_\rho \searrow f \qquad [\![t]\!]_\rho \searrow a \qquad f \cdot a \searrow b}{[\![s\ t]\!]_\rho \searrow b}
\qquad
\frac{[\![t]\!]_\rho \searrow a}{[\![\mathtt{lift}_i\ t]\!]_\rho \searrow \mathbf{lift}_i\ a}
\qquad
\frac{[\![t]\!]_\rho \searrow a \qquad \mathrm{unlift}\cdot a \searrow b}{[\![\mathtt{unlift}\ t]\!]_\rho \searrow b}
$$

$$
\frac{[\![\sigma]\!]_s(\rho) \searrow \phi \qquad [\![t]\!]_\phi \searrow a}{[\![t[\sigma]]\!]_\rho \searrow a}
$$

$\boxed{[\![\sigma]\!]_s(\rho) \searrow \phi}$ ρ is updated to ϕ

$$
\frac{}{[\![\mathtt{Id}]\!]_s(\rho) \searrow \rho}
\qquad
\frac{}{[\![\uparrow]\!]_s(\rho) \searrow \mathsf{drop}\ \rho}
\qquad
\frac{[\![\sigma]\!]_s(\rho) \searrow \phi \qquad [\![t]\!]_\phi \searrow d}{[\![\sigma, t : T]\!]_s(\rho) \searrow \phi; d}
\qquad
\frac{[\![\tau]\!]_s(\rho) \searrow \phi \qquad [\![\sigma]\!]_s(\phi) \searrow \theta}{[\![\sigma \circ \tau]\!]_s(\rho) \searrow \theta}
$$

Fig. 14. Relational definitions of the evaluation functions of the unascribed system.

B Definitions of the Unascribed System

We present the definitions of the syntax (Fig. 12), domain values (Fig. 13), and NbE algorithm of the unascribed system, as discussed in Sec. 7. In contrast to the ascribed system, the major difference is the absence of universe level annotations. The other difference is the type read-back function is now defined as $\mathrm{R}^{\mathrm{ty}}_n A$ instead of ${}^i\mathrm{R}^{\mathrm{ty}}_n A$.

$\boxed{f \cdot a \searrow b}$ apply f to a is b

$$\frac{\llbracket t \rrbracket_{\rho;a} \searrow b}{\llparenthesis \boldsymbol{\lambda} t \rrparenthesis_\rho \cdot a \searrow b} \qquad \frac{\llbracket T \rrbracket_{\rho,a} \searrow B}{\uparrow_{\llparenthesis \mathbf{\Pi}\ A\ T \rrparenthesis_\rho} e \cdot a \searrow \uparrow_B e\ (\downarrow_A a)}$$

$\boxed{\mathrm{rec}{\cdot}(T, a, s, b, \rho) \searrow c}$ rec-apply b to a and r under ρ is c

$$\frac{}{\mathrm{rec}{\cdot}(z.T, a, x, y.r, \mathbf{0}, \rho) \searrow a} \qquad \frac{\mathrm{rec}{\cdot}(z.T, a, x, y.s, b, \rho) \searrow c \quad \llbracket s \rrbracket_{\rho;b;c} \searrow c'}{\mathrm{rec}{\cdot}(z.T, a, x, y.s, \mathbf{suc}\ b, \rho) \searrow c'}$$

$$\frac{\llbracket T \rrbracket_{\rho;\uparrow_A e} \searrow B}{\mathrm{rec}{\cdot}(z.T, a, x, y.s, \uparrow_A e, \rho) \searrow \uparrow_B \llparenthesis \mathbf{rec}\ z.T\ \ a\ (x, y.s)\ e \rrparenthesis_\rho}$$

$\boxed{\mathrm{unlift}{\cdot}\ a \searrow b}$ unlift-apply a is b

$$\frac{}{\mathrm{unlift}{\cdot}\ \mathbf{lift}_j\ a \searrow a} \qquad \frac{}{\mathrm{unlift}{\cdot}\ \uparrow_{\mathbf{Lift}_j\ A} e \searrow \uparrow_A \mathbf{unlift}\ e}$$

Fig. 15. Relational definitions of the helper evaluation functions of the unascribed system.

$\boxed{\mathrm{R}^{\mathrm{nf}}_n\ d \searrow v}$ Readback from normal semantic value d to normal form v

$\boxed{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u}$ Readback from neutral semantic value e to neutral form u

$\boxed{\mathrm{R}^{\mathrm{ty}}_n\ A \searrow V}$ Readback from semantic value A of types to normal form V

$$\frac{\mathrm{R}^{\mathrm{ty}}_n\ A \searrow W}{\mathrm{R}^{\mathrm{nf}}_n\ \downarrow_{\mathbf{Set}_i} A \searrow W} \qquad \frac{}{\mathrm{R}^{\mathrm{nf}}_n\ \downarrow_N \mathbf{0} \searrow \texttt{0}} \qquad \frac{\mathrm{R}^{\mathrm{nf}}_n\ \downarrow_N a \searrow w}{\mathrm{R}^{\mathrm{nf}}_n\ \downarrow_N \mathbf{suc}\ a \searrow \texttt{suc}\ w}$$

$$\frac{\mathrm{R}^{\mathrm{ty}}_n\ A \searrow W \qquad a \cdot \uparrow_A \mathtt{x}_n \searrow b \qquad \llbracket T \rrbracket_{\rho;\uparrow_A \mathtt{x}_n} \searrow B \qquad \mathrm{R}^{\mathrm{nf}}_{1+n}\ \downarrow_B b \searrow w}{\mathrm{R}^{\mathrm{nf}}_n\ \downarrow_{\llparenthesis \mathbf{\Pi}\ A\ T \rrparenthesis_\rho} a \searrow \lambda(x_0 : W).w} \qquad \frac{\mathrm{unlift}{\cdot}\ a \searrow b \qquad \mathrm{R}^{\mathrm{nf}}_n\ \downarrow_A b \searrow w}{\mathrm{R}^{\mathrm{nf}}_n\ \downarrow_{(\mathbf{Lift}_j\ A)} a \searrow \texttt{Lift}_j\ w}$$

$$\frac{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u}{\mathrm{R}^{\mathrm{nf}}_n\ \downarrow_N \uparrow_A e \searrow u} \qquad \frac{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u}{\mathrm{R}^{\mathrm{nf}}_n\ \downarrow_{(\uparrow_A E)} \uparrow_{A'} e \searrow u}$$

$$\frac{}{\mathrm{R}^{\mathrm{ne}}_n\ x_l \searrow \mathtt{x}_{n-l-1}} \qquad \frac{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u \quad \mathrm{R}^{\mathrm{nf}}_n\ d \searrow w}{\mathrm{R}^{\mathrm{ne}}_n\ e\ d \searrow u\ w} \qquad \frac{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow u}{\mathrm{R}^{\mathrm{ne}}_n\ \mathbf{unlift}\ e \searrow \texttt{unlift}\ u}$$

$$\frac{\begin{array}{c}\llbracket T \rrbracket_{\rho;\uparrow_{\mathrm{N}} \mathtt{x}_n} \searrow A \qquad \mathrm{R}^{\mathrm{ty}}_{1+n}\ A \searrow W \qquad \llbracket T \rrbracket_{\rho;\mathbf{0}} \searrow A' \\ \mathrm{R}^{\mathrm{nf}}_n\ (\downarrow_{A'} a) \searrow w \qquad \llbracket t \rrbracket_{\rho;\uparrow_{\mathrm{N}} \mathtt{x}_n;\uparrow_A \mathtt{x}_{1+n}} \searrow b \qquad \llbracket T \rrbracket_{\rho;\mathbf{suc}(\downarrow_N \mathtt{x}_n)} \searrow A'' \qquad \mathrm{R}^{\mathrm{nf}}_{2+n}\ \downarrow_{A''} b \searrow w' \qquad \mathrm{R}^{\mathrm{ne}}_n\ e \searrow u\end{array}}{\mathrm{R}^{\mathrm{ne}}_n\ \llparenthesis \mathbf{rec}\ z.T\ \ a\ (x, y.s)\ e \rrparenthesis_\rho \searrow \texttt{rec}\ (z_0.W)\ w\ (x_0.y_1.w')\ u}$$

$$\frac{\mathrm{R}^{\mathrm{ne}}_n\ e \searrow V}{\mathrm{R}^{\mathrm{ty}}_n\ \downarrow_A e \searrow V} \qquad \frac{}{\mathrm{R}^{\mathrm{ty}}_n\ \mathbf{Set}\ i \searrow \texttt{Set}_i} \qquad \frac{}{\mathrm{R}^{\mathrm{ty}}_n\ \mathbf{N} \searrow \texttt{N}} \qquad \frac{\mathrm{R}^{\mathrm{ty}}_n\ A \searrow V \qquad \llbracket T \rrbracket_{\rho;\uparrow_A \mathtt{x}_n} \searrow B \qquad \mathrm{R}^{\mathrm{ty}}_{1+n}\ B \searrow W}{\mathrm{R}^{\mathrm{ty}}_n\ \llparenthesis \mathbf{\Pi}\ A\ T \rrparenthesis_\rho \searrow \Pi(x_0 : V).W}$$

$$\frac{\mathrm{R}^{\mathrm{ty}}_n\ A \searrow W}{\mathrm{R}^{\mathrm{ty}}_n\ \mathbf{Lift}_j\ A \searrow \texttt{Lift}_j\ W}$$

Fig. 16. Relational definitions of the readback functions of the unascribed system.

C Additional Properties

The properties are stated in their simple forms. Some of the actual statements to be proved by induction require generalization and additional conditions. Compared with the cumulative system, the proof in the non-cumulative system is more technically challenging, due to the extra handling of proof-relevance caused by universe-level adjustment, as discussed in Sec. 6. ○ denotes the simple form and ● denotes the full form. Blue denotes the non-cumulative system and yellow denotes the cumulative system.

The following are properties of the PER model of type values and values.

Theorem C.1 (Symmetry ○ ● ○ ●). *Given* $\mathcal{D}_1 :: A \equiv B \in \mathcal{S}et_i$,

- $\mathcal{D}_2 :: B \equiv A \in \mathcal{S}et_i$;
- *if* $a \equiv b \in \mathcal{E}l_i(\mathcal{D}_1)$, *then* $b \equiv a \in \mathcal{E}l_i(\mathcal{D}_2)$.

The statements between the non-cumulative and cumulative system are slightly different, where in the cumulative statements, universe levels can be adjusted.

Theorem C.2 (Transitivity ○ ●). *Given* $\mathcal{D}_1 :: A \equiv B \in \mathcal{S}et_i$ *and* $\mathcal{D}_2 :: B \equiv C \in \mathcal{S}et_i$,

- $\mathcal{D}_3 :: A \equiv C \in \mathcal{S}et_i$;
- *if* $a \equiv b \in \mathcal{E}l_i(\mathcal{D}_1)$, *and* $b \equiv c \in \mathcal{E}l_i(\mathcal{D}_2)$, *then* $a \equiv c \in \mathcal{E}l_i(\mathcal{D}_3)$.

Theorem C.3 (Transitivity ○ ●). *Given* $\mathcal{D}_1 :: A \equiv B \in \mathcal{S}et_i$ *and* $\mathcal{D}_2 :: B \equiv C \in \mathcal{S}et_j$,

- $\mathcal{D}_3 :: A \equiv C \in \mathcal{S}et_i$;
- *if* $a \equiv b \in \mathcal{E}l_i(\mathcal{D}_1)$, *and* $b \equiv c \in \mathcal{E}l_j(\mathcal{D}_2)$, *then* $a \equiv c \in \mathcal{E}l_i(\mathcal{D}_3)$.

Theorem C.4 (Irrelevance to the Right Type ○ ●). *Given* $\mathcal{D}_1 :: A \equiv B \in \mathcal{S}et_i$ *and* $\mathcal{D}_2 :: A \equiv B' \in \mathcal{S}et_i$, *if* $a \equiv b \in \mathcal{E}l_i(\mathcal{D}_1)$, *then* $a \equiv b \in \mathcal{E}l_i(\mathcal{D}_2)$.

Theorem C.5 (Irrelevance to the Right Type ○). *Given* $\mathcal{D}_1 :: A \equiv B \in \mathcal{S}et_i$ *and* $\mathcal{D}_2 :: A \equiv B' \in \mathcal{S}et_j$, *if* $a \equiv b \in \mathcal{E}l_i(\mathcal{D}_1)$, *then* $a \equiv b \in \mathcal{E}l_j(\mathcal{D}_2)$.

The cumulative system needs to additionally justify cumulativity of the PER model. Other necessary lemmas to adjust universe levels are not listed here. The precision of universe levels in the non-cumulative system saves this step.

Theorem C.6 (Semantic Cumulativity ○ ●). *Given* $i \leq j$ *and* $\mathcal{D}_1 :: A \equiv B \in \mathcal{S}et_i$,

- $\mathcal{D}_2 :: A \equiv B \in \mathcal{S}et_j$;
- *if* $a \equiv b \in \mathcal{E}l_i(\mathcal{D}_1)$, *then* $a \equiv b \in \mathcal{E}l_j(\mathcal{D}_2)$.

The following are properties of the gluing model of types and terms.

Theorem C.7 (Preservation under Equivalent Types ○ ○ and Contexts ○ ○). *Given* $\mathcal{D}{::}A \equiv B \in \mathcal{S}et_i$, $\vdash \Gamma \equiv \Gamma'$, *and* $\Gamma \vdash T \approx T' :^{1+i} \texttt{Set}_i$, *if* $\Gamma \vdash T \circledR^i \mathcal{D}$, *then* $\Gamma' \vdash T' \circledR^i \mathcal{D}$.

Theorem C.8 (Preservation under Equivalent Terms ○ ○, Types ○ ○ and Contexts ○ ○). *Given* $\mathcal{D}{::}A \equiv B \in \mathcal{S}et_i$, $\vdash \Gamma \equiv \Gamma'$, $\Gamma \vdash t \approx t' :^i T$ *and* $\Gamma \vdash T \approx T' :^{1+i} \texttt{Set}_i$, *if* $\Gamma \vdash t : T \circledR^i a \in \mathcal{E}l_i\ (\mathcal{D})$, *then* $\Gamma' \vdash t' : T' \circledR^i a \in \mathcal{E}l_i\ (\mathcal{D})$.

Theorem C.9 (Irrelevance to Symmetry ○ ○). *Given* $\mathcal{D}_1 :: A \equiv B \in \mathcal{S}et_i$, $\mathcal{D}_2 :: B \equiv A \in \mathcal{S}et_i$,

- *if* $\Gamma \vdash T \circledR^i \mathcal{D}_1$, *then* $\Gamma \vdash T \circledR^i \mathcal{D}_2$;
- *if* $\Gamma \vdash t : T \circledR^i a \in \mathcal{E}l_i\ (\mathcal{D}_1)$, *then* $\Gamma \vdash t : T \circledR^i a \in \mathcal{E}l_i\ (\mathcal{D}_2)$.

Theorem C.10 (Irrelevance to the Right Type ○ ○). *Given* $\mathcal{D}_1 :: A \equiv B \in \mathcal{S}et_i$, $\mathcal{D}_2 :: A \equiv B' \in \mathcal{S}et_i$,

- *if* $\Gamma \vdash T \circledR^i \mathcal{D}_1$, *then* $\Gamma \vdash T \circledR^i \mathcal{D}_2$;
- *if* $\Gamma \vdash t : T \circledR^i a \in \mathcal{E}l_i\ (\mathcal{D}_1)$, *then* $\Gamma \vdash t : T \circledR^i a \in \mathcal{E}l_i\ (\mathcal{D}_2)$.

Theorem C.11 (Monotonicity of the Gluing Model ○ ● ○ ●). *Given* $\mathcal{D} :: A \equiv B \in \mathcal{S}et_i$ *and* $\Delta \vdash \kappa : \Gamma$,

- *if* $\Gamma \vdash T \circledR^i \mathcal{D}$, *then* $\Delta \vdash T[\kappa] \circledR^i \mathcal{D}$;
- *if* $\Gamma \vdash t : T \circledR^i a \in \mathcal{E}l_i\ (\mathcal{D})$, *then* $\Delta \vdash t[\kappa] : T[\kappa] \circledR^i a \in \mathcal{E}l_i\ (\mathcal{D})$.

Though we only show the properties of the PER model and gluing model of terms and types, these models of contexts, environments and substitutions share a similar set of properties.