Normalization by Evaluation for Sized Dependent Types

ANDREAS ABEL, Gothenburg University
ANDREAS VEZZOSI, Chalmers
THEO WINTERHALTER, École Normale Supérieure de Cachan

Sized types have been developed to make termination checking more perspicuous, more powerful, and more modular by integrating termination into type checking. In dependently-typed proof assistants where proofs by induction are just recursive functional programs, the termination checker is an integral component of the trusted core, as validity of proofs depend on termination. However, a rigorous integration of full-fledged sized types into dependent type theory is lacking so far. Such an integration is non-trivial, as explicit sizes in proof terms might get in the way of equality checking, making terms appear distinct that should have the same semantics.

In this article, we integrate dependent types and sized types with higher-rank size polymorphism, which is essential for generic programming and abstraction. We introduce a size quantifier ∀ which lets us ignore sizes in terms for equality checking, alongside with a second quantifier Π for abstracting over sizes that do affect the semantics of types and terms. Judgmental equality is decided by an adaptation of normalization-by-evaluation for our new type theory, which features *type shape*-directed reflection and reification. It follows that subtyping and type checking of normal forms are decidable as well, the latter by a bidirectional algorithm.

CCS Concepts: • **Theory of computation → Type theory**; **Type structures**; *Program verification*; Operational semantics;

Additional Key Words and Phrases: dependent types, eta-equality, normalization-by-evaluation, proof irrelevance, sized types, subtyping, universes.

ACM Reference format:
Andreas Abel, Andreas Vezzosi, and Theo Winterhalter. 2017. Normalization by Evaluation for Sized Dependent Types. *Proc. ACM Program. Lang.* 1, 1, Article 1 (January 2017), 36 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Dependently-typed programming languages and proof assistants, such as Agda [2017] and Coq [INRIA 2016], require programs to be total, for two reasons. First, for consistency: since propositions are just types and proofs of a proposition just programs which inhabit the corresponding type, some types need to be empty; otherwise, each proposition would be true. However, in a partial language with general recursion, each type is inhabited by the looping program $f = f$. Secondly, totality is needed for decidability of type checking. Since types can be the result of a computation, we need computation to terminate during type checking, even for *open terms*, i. e., terms with free variables.

Consequently, the aforementioned languages based on Type Theory come with a termination checker, which needs to reject all non-terminating programs, and should accept sufficiently many terminating programs to allow the user to express her algorithms. In current termination checkers, programs are required to terminate by structural descent [Giménez 1995]; the structural order may be extended to a lexicographic [Abel and Altenkirch 2002] or size-change termination criterion

2017. 2475-1421/2017/1-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[Lee et al. 2001; Wahlstedt 2007]. This is not a fundamental limitation, since Type Theory allows many functions to be expressed in a structurally recursive manner, if needed by the help of a well-founded relation [Nordström 1988], inductive domain predicates [Bove and Capretta 2005], or inductive descriptions of the function graph [Bove 2009]. However, the syntactic termination check is very sensitive to reformulations of the program and hostile to abstraction [Abel 2012].

Sized types [Hughes et al. 1996] delegate the checking for structural descent to the type system by annotating data types with a size parameter. The type checker can then ensure that in recursive calls the size goes down, certifying termination. In the simplest setting [Abel 2008; Barthe et al. 2004], the size is just an upper bound on the tree height of the data structure; however, more sophisticated size annotations have also been considered [Blanqui 2004; Xi 2002]. Most sized type systems are non-dependent [Abel and Pientka 2016; Amadio and Coupet-Grimal 1998; Barthe et al. 2008a,b; Blanqui and Riba 2006; Lago and Grellois 2017], yet the combination of sized and dependent types has been studied as well [Barthe et al. 2006; Blanqui 2005; Grégoire and Sacchini 2010; Sacchini 2013, 2014]. However, to the best of our knowledge, no study combines *higher-rank size polymorphism* with full-fledged dependent types.[1]

Higher-rank size quantification takes termination checking to the next level; it is necessary for abstraction and generic programming. For instance, it allows us to write a generic tree traversal which applies a user-given preprocessor on subtrees before recursively descending into these trees, and a postprocessor after surfacing from the descent. The condition is that preprocessing does not increase the size of the subtree; otherwise, termination could not be guaranteed. Concretely, assume a type $\mathsf{T}\,i$ of trees of size $< i$ with a constructor $\mathsf{node} : \forall i.\, \mathsf{List}\,(\mathsf{T}\,i) \to \mathsf{T}\,(i+1)$ which takes a finite list of subtrees to form a new tree. In the following definition of trav, the preprocessing $\mathit{pre} : \forall i.\, \mathsf{T}\,i \to \mathsf{T}\,i$ can be safely applied to input tree t because the type of *pre* bounds the size of $\mathit{pre}\,t$ by the size of t. In case $\mathit{pre}\,t = \mathsf{node}\,\mathit{ts}$, the trees in the list *ts* are still guaranteed to be of strictly smaller size than t, thus, the recursive call to trav, communicated via the map function for lists, is safe.

$$\begin{array}{l}\mathsf{trav} : (\mathit{pre} : \forall i.\, \mathsf{T}i \to \mathsf{T}i)\ (\mathit{post} : \mathsf{T}\infty \to \mathsf{T}\infty) \to \forall i.\, \mathsf{T}\,i \to \mathsf{T}\,\infty \\ \mathsf{trav}\ \mathit{pre}\ \mathit{post}\ t = \mathit{post}\ (\mathsf{case}\ \mathit{pre}\,t\ \mathsf{of}\ \{\, \mathsf{node}\ \mathit{ts} \to \mathsf{node}\,(\mathsf{map}\ (\mathsf{trav}\,\mathit{pre}\,\mathit{post})\ \mathit{ts})\,\})\end{array}$$

The display shows the *Curry*-style program as provided by the user, but state-of-the-art type checkers elaborate the program from surface syntax into an internal *Church*-style syntax with explicit type abstractions and type applications.[2] With implicit type and size applications elaborated, trav would look as follows:

$$\mathsf{trav}\ \mathit{pre}\ \mathit{post}\ i\ t = \mathit{post}\ (\mathsf{case}\ \mathit{pre}\,i\,t\ \mathsf{of}\ \{\, \mathsf{node}\,j\,\mathit{ts} \to \mathsf{node}\,\infty\,(\mathsf{map}\ (\mathsf{T}\,j)\ (\mathsf{T}\,\infty)\ (\mathsf{trav}\,\mathit{pre}\,\mathit{post}\,j)\ \mathit{ts})\})$$

Church-style syntax is the basis for all program analyses and transformations to follow and should be considered as the *true* syntax. However, from a dependent-type perspective, explicit size applications in terms can be problematic when the type checker compares terms for equality—which is necessary as types can depend on values. Inferred sizes may not be unique, as we have subtyping $\mathsf{T}\,i \leq \mathsf{T}\,j$ for $i \leq j$: we can always weaken an upper bound. For instance, given $\mathit{ts} : \mathsf{List}\,(\mathsf{T}\,i)$, any of the terms $\mathsf{node}\,i\,\mathit{ts}$, $\mathsf{node}\,(i+1)\,\mathit{ts}, \ldots, \mathsf{node}\,\infty\,\mathit{ts}$ has type $\mathsf{T}\,\infty$. Yet semantically, all these trees are equal, thus, the syntactic equality check should ignore the size argument to node. Similarly, in the application $\mathit{pre}\,i\,t$ the size argument i should be ignored by the equality check. Yet $\mathit{pre}\,i : \mathsf{T}\,i \to \mathsf{T}\,i$ and $\mathit{pre}\,j : \mathsf{T}\,j \to \mathsf{T}\,j$ have different types for $i \neq j$, and moreover these function types are not in the subtyping relation due to the mixed-variant occurrence of the size parameter. It seems that during equality checking we have to consider terms of different types, at least for a while. Once we apply

[1] Xi [2002] has first-class size polymorphism, but only indexed types, no universes or large eliminations.
[2] Agda, Coq, Idris [Brady 2013], and Haskell [Sulzmann et al. 2007] all have Church-style internal languages.

pre i and *pre* j to the same tree $t : \mathsf{T}\,k$, which determines $i = j = k$, we are back to safety. However, allowing types to differ during an equality check needs special consideration, especially when the equality-check is type directed.

Consider the analogous situation for the polymorphic lambda calculus System F, be it the predicative variant or not, extended by a unit type $\mathbf{1}$. For Church-style, we can give a type-directed $\beta\eta$-equality test which equates all terms at the unit type. The most interesting rules are the η-rules for unit and function type and the congruence rule for type application:

$$\frac{}{\Gamma \vdash t = t' : \mathbf{1}} \qquad \frac{\Gamma, x{:}A \vdash t\,x = t'\,x : B}{\Gamma \vdash t = t' : A \to B} \qquad \frac{\Gamma \vdash t = t' : \forall X.B}{\Gamma \vdash t\,A = t'\,A : B[A/X]}$$

The Curry-style version replaces the last conclusion by $\Gamma \vdash t = t' : B[A/X]$ where the type A to instantiate X has to be guessed. However, in Curry-style more terms are equated than in Church-style, as for instance the Church-style terms $t\,A\,(\lambda x : A.\,x)$ and $t\,B\,(\lambda x : B.\,x)$ map to the same Curry-style term $t\,(\lambda x.\,x)$. How would we adapt the algorithm for Church-style such that it equates all terms that are equal in Curry-style? The conclusion of the last rule could be changed to $\Gamma \vdash t\,A = t'\,A' : B[A/X]$, but then the second term $t'\,A'$ does not have the ascribed type $B[A/X]$, and η-laws applied to this term would be unsound. For instance, the algorithm would yield $t\,\mathbf{1}\,x = t\,(A \to A)\,y$ even for $x \neq y$. We could also consider a heterogeneous check $(\Gamma \vdash t : A) = (\Gamma' \vdash t' : A')$ where each term is paired with its own type and context, but this leaves us with the dilemma of explaining the meaning of this judgement when A and A' are incompatible.

Does the literature offer a solution to this problem? In fact, a Church-style calculus with Curry-style equality has been studied before, it is ICC* [Barras and Bernardo 2008; Mishra-Linger and Sheard 2008] based on Miquel's Implicit Calculus of Constructions [2001]. In ICC*, equality is checked by erasing all type abstractions and applications, and comparing the remaining untyped terms for $\beta\eta$-equality. While this works for η-laws that can be formulated on untyped terms, such as η-contraction of functions $\lambda x.\,t\,x \longrightarrow_\eta t$ (when x not free in t), it does not extend to type-directed η-laws such as extensionality for the unit type. Further, ICC* is not a type theory formulated with a typed equality judgement, which makes it hard to define its models [Miquel 2000]—we wish not to go there, but stay within the framework of Martin-Löf Type Theory [1975].

Now, if the types of compared Curry-style terms are not equal, can they be sufficiently related to give a proper meaning to the algorithmic equality judgement? It has already been observed that for a type-directed equality check the precise type is not necessary, a *shape* or *skeleton* is sufficient. The skeleton informs the algorithm whether the terms under comparison are functions, inhabitants of the unit type, or something else, to possibly apply the appropriate η-law. For the Logical Framework (LF), the simplest dependent lambda-calculus, the skeletons are simple types that can be obtained from the original dependent types by erasing the dependencies: dependent function types map to non-dependent ones and indexed data types to simple data types. Harper and Pfenning [2005] present such an equality check for LF which is directed by simple types, and their technique should scale to other type theories that admit dependency erasure.[3]

By *large eliminations* [Werner 1992] we refer to types computed by case distinction over values; they occur in type theories that feature both universes and data types. In the presence of large eliminations, dependency erasure fails, and it is not clear what the skeleton of a dependent type should be. For instance consider the type $(n{:}\mathbb{N}) \to \underbrace{A \to \cdots \to A}_{n} \to A$ of n-ary functions; its shape

[3]For instance, the types of the Calculus of Constructions erase to F^{ω}-types [Geuvers 1994], and the latter could be used to direct the equality check. Lovas and Pfenning [2010] consider also *refinement types for logical frameworks* which can be erased to simple types.

is dependent on the value of n, thus cannot be determined statically. Thus, the "skeleton" idea is also not directly applicable.

Going beyond the standard syntax-directed equality check, there is a technique that can deal with dynamic η-expansion. It is a type-directed normalization function inspired by normalization-by-evaluation (NbE) that computes η-long normal forms [Berger and Schwichtenberg 1991; Danvy 1999]. We can check the computed normal forms for identity and, thus, decide definitional equality. NbE has proven to be a robust method to decide equality in powerful type theories with non-trivial η-laws. It scales to universes and large eliminations [Abel et al. 2007], topped with singleton types or proof irrelevance [Abel et al. 2011], and even impredicativity [Abel 2010]. At its heart there are reflection $\uparrow^T$ and reification $\downarrow^T$ functions directed by type T and orchestrating just-in-time η-expansion. Reflection $\uparrow^T x$ maps variables x into the realm of values of type T and lets us compute with open terms. Reification $\downarrow^T a$ takes a value a of type T and computes its long normal form. For instance, the normal form of a closed function $f : U \to T$ would be $\lambda x. \downarrow^T (f\ (\uparrow^U x))$, and for its dependently-typed variant $f : (x{:}U) \to T[x]$ it would be $\lambda x. \downarrow^{T[\uparrow^U x]} (f\ (\uparrow^U x))$.

The central technical observation is that reflection and reification do not need the precise type T, they work the same for any *shape* S of T. We managed, while not introducing a new syntax for shapes, to define a relation $T \lessapprox S$ on type values stating that type S qualifies as shape for type T. Shapes unfold dynamically during reflection and reification. For example, when reflecting a variable x into the polymorphic function type $\forall i. F\,i$ where $F\,i = \mathsf{Nat}\ i \to \mathsf{Nat}\ i$, we obtain $(\uparrow^{\forall i. F\,i} x)\ i = \uparrow^{F\,i} (x\ i)$ for size i and $(\uparrow^{\forall i. F\,i} x)\ j = \uparrow^{F\,j} (x\ j)$ for size j. The new types $F\,i$ and $F\,j$ we reflect at are no longer equal (and they are not subtypes of each other), but they still have the same shape, $\mathsf{Nat}\ \_ \to \mathsf{Nat}\ \_$. This means they will still move in lock-step in respect to η-expansion, which is sufficient to prove NbE correct for judgmental equality. We call the enabling property of F *shape irrelevance*, meaning that for any pair i, j of legal arguments, $F\,i$ and $F\,j$ have the same shape. Whenever we form a irrelevant function type $\forall x{:}U. T[x]$, we require $T[x]$ to be shape-irrelevant in x. This is the middle ground between ICC*, where no restriction is placed on T but η for unit types is out of reach (at least for the moment), and Pfenning's [2001] irrelevance modality, adapted to full dependent types by Abel and Scherer [2012], which requires T to be irrelevant in x and, thus, has type equality $T[i] = T[j]$.

For the time being, we do not (and cannot) develop a general theory of shape irrelevance. We confine ourselves to size-irrelevant function types $\forall i. T[i]$. This relieves us from defining a special shape-irrelevance modality, since all size-indexed types $T[i]$ *are* shape irrelevant in i, simply because there is no case distinction on size, and sizes appear relevantly only under a sized type constructor such as Nat. Our technique would not extend to the polymorphic types $\forall X. B[X]$ of System F. Even though there is no case distinction on types, shape irrelevance of $B[X]$ fails in general, as X could appear as a type on the top-level, e.g. in $B[X] = X \to X$, and then $B[\mathbf{1}]$ and $B[A \to A]$ would have distinct shapes.

To summarize, this article makes the following novel contributions:

(1) We present the first integration of a dependent type theory with higher-rank size polymorphism. Concretely, we consider a type theory à la Martin-Löf with dependent function types, cumulative universes, subtyping, a judgmental equality with η-laws, a sized type of natural numbers and two size quantifiers: an irrelevant one ($\forall$) for binding of sizes in irrelevant positions, and a relevant one (Π) for binding of sizes in shape-irrelevant positions (Section 3). Judgmental equality features a "Curry-style" rule for irrelevant size application which ignores the size arguments, and consequently, the corresponding typing rule will also ignore the size

argument. (In the following rules, a, a', and b stand for arbitrary size expressions.)

$$\frac{\Gamma \vdash t = t' : \forall i.\, T}{\Gamma \vdash t\, a = t'\, a' : T[b/i]} \qquad \frac{\Gamma \vdash t : \forall i.\, T}{\Gamma \vdash t\, a : T[b/i]} \qquad \frac{\Gamma \vdash t = t' : \Pi i.\, T}{\Gamma \vdash t\, a = t'\, a : T[a/i]} \qquad \frac{\Gamma \vdash t : \Pi i.\, T}{\Gamma \vdash t\, a : T[a/i]}$$

Our substitution theorem distinguishes term- from type-side substitutions.

(2) We adapt normalization-by-evaluation (NbE) to sized types and size quantification and show that it decides judgmental equality (sections 4 and 5). The novel technical tool is a relation $T \sqsubseteq S$ which relates a type T to its possible shapes S. This approximation relation allows reflection and reification at size-polymorphic types $\forall i.\, T$. As usual for the meta-theory of Type Theory with large eliminations, the machinery is involved, but we just require the usual two logical relations: First, a PER model to define the semantics of types and prove the completeness of NbE (Section 4). Secondly, a relation between syntax and semantics to prove soundness of NbE (Section 5).

(3) We present an bidirectional type checking algorithm [Coquand 1996] which takes the irrelevant size argument as reliable hint for the type checker (sections 6 and 7). It is complete for normal forms which can be typed with the restricted rule for ∀-elimination:

$$\frac{\Gamma \vdash t : \forall i.\, T}{\Gamma \vdash t\, a : T[a/i]}$$

The algorithm employs the usual lazy reduction for types, i. e., just-in-time weak-head evaluation, in type and sub-type checker [Huet 1989]. In this, it improves on Fridlender and Pagano [2013] which instruments full normalization (NbE) at every step.

This article is accompanied by a prototypical type checker `Sit` which implements the type system and type checking algorithm as described in the remainder of the paper. But before going into the technical details, we will motivate our type system from a practical perspective: reasoning about programs involving sized types in Agda.

2 SIZE IRRELEVANCE IN PRACTICE

In this section, we show how the lack of size irrelevance prevents us from reasoning naturally about programs involving sized types in Type Theory. We focus on Agda, at the time of writing the only mature implementation of Type Theory with an experimental integration of sized types.

The problem of the current implementation of sized types in Agda can be demonstrated by a short example. Consider the type of sized natural numbers.

```
data Nat : Size → Set where
  zero : ∀ i → Nat (i + 1)
  suc  : ∀ i → Nat i → Nat (i + 1)
```

The predecessor function is *size preserving*, i. e., the output can be assigned the same upper bound i as the input. In the code to follow, the dot on the left hand side, preceding $(i + 1)$, marks an *inaccessible* pattern. Its value is determined by the subsequent match on the natural number argument, no actual matching has to be carried out on this argument.

```
pred : ∀ i → Nat i → Nat i
pred .(i + 1) (zero i)  = zero i
pred .(i + 1) (suc i x) = x
```

Note that in the second clause, we have applied subtyping to cast x from Nat i to Nat $(i + 1)$.

We now define subtraction $x \dot{-} y$ on natural numbers, sometimes called the monus function, which computes $\max(0, x - y)$. It is defined by induction on the size j of the second argument y, while the output is bounded by size i of the first argument x. The input-output relation of monus is needed for a natural implementation of Euclidean divison.

There are several ways to implement monus, we have chosen a tail-recursive variant which treats the first argument as accumulator. It computes the result by applying the predecessor function y times to x.

```
monus : ∀ i → Nat i → ∀ j → Nat j → Nat i
monus i x .(j + 1) (zero j)  = x
monus i x .(j + 1) (suc j y) = monus i (pred i x) j y
```

To document subgoals in proof terms, we introduce a mixfix version of the identity function with a visible type argument:

```
prove_by_ : (A : Set) → A → A
prove A by x = x
```

We now wish to prove that subtracting x from itself yields 0, by induction on x. The case $x = 0$ should be trivial, as $x \dot{-} 0 = x$ by definition, hence, $0 \dot{-} 0 = 0$. As simple proof by reflexivity should suffice. In case $x + 1$, the goal $0 \equiv (x + 1) \dot{-} (x + 1)$ should reduce to $0 \equiv x \dot{-} x$, thus, an application of the induction hypothesis should suffice. The following display shows that partial proofs, leaving holes {! … !} already filled with the desired proof terms.

```
monus-diag : ∀ i → (x : Nat i) → zero ∞ ≡ monus i x i x
monus-diag .(i + 1) (zero i)  = prove zero ∞ ≡ zero i                by {! refl !}
monus-diag .(i + 1) (suc i x) = prove zero ∞ ≡ monus (i + 1) x i x by {! monus-diag i x !}
```

Unfortunately, in Agda our proof is not accepted, as sizes get in the way. In the first goal, there is a mismatch between size ∞ and size i, the latter coming from the computation of monus $(i + 1)$ (zero i) $(i+1)$ (zero i). In the second goal, there is a mismatch between size $i+1$ in term monus $(i+1)\,x\,i\,x$ of the reduced goal and size i of the respective term monus $i\,x\,i\,x$ from the induction hypothesis we wish to apply.

The proof would go through if Agda ignored sizes where they act as *type argument*, i. e., in constructors and term-level function applications, but not in types where they act as regular argument, e. g., in Nat i.

The solution we present in this article already works in current Agda,[4] but the implementation is not perfect. Thus, it is hidden under a scarcely documented flag:

```
{-# OPTIONS --experimental-irrelevance #-}
```

We mark the size argument of Nat as *shape irrelevant* by preceding the binder with two dots. In a future implementation, we could treat all data type parameters as shape irrelevant by default. In the types of the constructors, we mark argument i as *irrelevant* by prefixing the binder with a single dot. This is sound because i occurs in subsequent parts of the type only in shape-irrelevant positions.

```
data Nat : ..(i : Size) → Set where
  zero : ∀ .i → Nat (i + 1)
  suc  : ∀ .i → Nat i → Nat (i + 1)
```

Similarly, "type" argument i to pred is irrelevant. Agda checks that it only occurs shape-irrelevantly in the type and irrelevantly in the term. The latter is the case since i is also an irrelevant argument to the constructors zero and suc; otherwise, we would get a type error.

[4] https://github.com/agda/agda, development version of 2017-02-27.

$\mathsf{pred} : \forall\,.i \to \mathsf{Nat}\ i \to \mathsf{Nat}\ i$
$\mathsf{pred}\ .(i+1)\ (\mathsf{zero}\ i) \ = \mathsf{zero}\ i$
$\mathsf{pred}\ .(i+1)\ (\mathsf{suc}\ i\ x) = x$

The two size arguments i and j to monus are also irrelevant. In this case, type checking succeeds since the size argument to pred has been declared irrelevant.

$\mathsf{monus} : \forall\,.i \to \mathsf{Nat}\ i \to \forall\,.j \to \mathsf{Nat}\ j \to \mathsf{Nat}\ i$
$\mathsf{monus}\ i\ x\ .(j+1)\ (\mathsf{zero}\ j) \ = x$
$\mathsf{monus}\ i\ x\ .(j+1)\ (\mathsf{suc}\ j\ y) = \mathsf{monus}\ i\ (\mathsf{pred}\ i\ x)\ j\ y$

Now, with sizes being ignored in the involved terms, we can complete the proof of our lemma:

$\mathsf{monus\text{-}diag} : \forall\,.i \to (x : \mathsf{Nat}\ i) \to \mathsf{zero}\ \infty \equiv \mathsf{monus}\ i\ x\ i\ x$
$\mathsf{monus\text{-}diag}\ .(i+1)\ (\mathsf{zero}\ i) \ = \mathsf{prove}\ \mathsf{zero}\ \infty \equiv \mathsf{zero}\ i \qquad \mathsf{by}\ \mathsf{refl}$
$\mathsf{monus\text{-}diag}\ .(i+1)\ (\mathsf{suc}\ i\ x) = \mathsf{prove}\ \mathsf{zero}\ \infty \equiv \mathsf{monus}\ (i+1)\ x\ i\ x\ \mathsf{by}\ \mathsf{monus\text{-}diag}\ i\ x$

3 A TYPE SYSTEM WITH IRRELEVANT SIZE APPLICATION

In this section, we give the syntax and the *declarative* typing, equality, and subtyping judgements. The typing relation $\Gamma \vdash t : T$ will *not* be decidable; instead, we present algorithmic typing $\Gamma \vdash t \leftleftarrows T$ in Section 7. However, equality and subtyping will be decidable for well-formed input, see sections 4–6.

We present our type theory as (domain-free) *pure type system* [Barendregt 1991] with extra structure. The *sorts* s are drawn from an infinite predicative hierarchy of universes Set_ℓ for $\ell \in \mathbb{N}$. Universes provide us with polymorphism and the capability to define types by recursion on values. Whether we have just two universes Set_0 and Set_1 or infinitely many, does not matter for the technical difficulty of the meta theory. The present setup have the advantage that every sort has again a sort since $\mathsf{Set}_\ell : \mathsf{Set}_{\ell+1}$, thus, we do not have to introduce a separate judgement $\Gamma \vdash T$ for well-formedness of types, we can define it as $\exists s.\ \Gamma \vdash T : s$.

Sort	$\ni s$	$::= \mathsf{Set}_\ell\ (\ell \in \mathbb{N})$	sort (universe)
Ann	$\ni \star$	$::= \div \mid :$	annotation (irrelevant, relevant)
Exp	$\ni t, u, T, U$	$::= w \mid t\ e$	expressions
Whnf	$\ni w, W$	$::= n \mid s \mid \mathsf{Size} \mid \Pi^\star U\ T \mid \lambda t \mid \mathsf{Nat}\ a \mid c$	weak head normal forms
Data	$\ni c$	$::= \mathsf{zero}\langle a\rangle \mid \mathsf{suc}\langle a\rangle t$	constructed data
NeExp	$\ni n$	$::= \mathsf{v}_i \mid n\ e$	neutral expressions
Elim	$\ni e$	$::= t \mid a \mid \langle a\rangle \mid \mathsf{case}_\ell\ T\ t_z\ t_s \mid \mathsf{fix}_\ell\ T\ t$	eliminations
SizeExp	$\ni a, b$	$::= \infty \mid o \mid \mathsf{v}_i + o\ (o \in \mathbb{N})$	size expressions
Cxt	$\ni \Gamma, \Delta$	$::= () \mid \Gamma.^{:}T \mid \Gamma.^{\div}\mathsf{Size}$	contexts
Subst	$\ni \eta, \rho, \sigma, \tau, \xi$	$::= () \mid (\sigma, t)$	substitutions

Fig. 1. Syntax.

For the expression syntax (see Fig. 1), we use de Bruijn [1972] indices v_i to represent variables. The index $i \in \mathbb{N}$ points to the ith enclosing binder of variable v_i. Binders are lambda abstraction λt and dependent function types $\Pi^\star U\ T$, which bind the 0th index in t and T, resp. For instance, the term $\lambda x.\ x\ (\lambda z.\ z)\ (\lambda y.\ x\ y)$ with named variables x, y, z has de Bruijn representation $\lambda.\ \mathsf{v}_0\ (\lambda.\ \mathsf{v}_0)\ (\lambda.\ \mathsf{v}_1\ \mathsf{v}_0)$.

The notation $\Pi^\star U\ T$ is an umbrella for three kinds of function types, where $\star \in \{\div, :\}$ is a relevance annotation borrowed from Pfenning [2001]. $\Pi^{:} U\ T$ is the ordinary dependent function type,

$\Pi^{:}\mathsf{Size}\,T$ is relevant size quantification, and $\Pi^{\div}\mathsf{Size}\,T$ is irrelevant size quantification. We omit the ":"-markers from Π by default (and also in contexts Γ) and write $\forall\,T$ for $\Pi^{\div}\mathsf{Size}\,T$. Examples for relevant size quantification $\Pi\,\mathsf{Size}\,T$ are $\Pi\,\mathsf{Size}\,\mathsf{Set}_0$ and $\Pi\,\mathsf{Size}\,\Pi\,(\mathsf{Nat}\,\mathsf{v}_0)\,\mathsf{Set}_0$. In a syntax with named variables and non-dependent function type they could be written as $\mathsf{Size} \to \mathsf{Set}_0$ and $(z\,{:}\,\mathsf{Size}) \to \mathsf{Nat}\,z \to \mathsf{Set}_0$, resp. An instance of irrelevant quantification $\forall T$ would be $\forall.\,\Pi\,(\mathsf{Nat}\,\mathsf{v}_0)\,(\mathsf{Nat}\,\mathsf{v}_1)$ which is $\forall z.\,\mathsf{Nat}\,z \to \mathsf{Nat}\,z$ in a named syntax. Herein, $\mathsf{Nat}\,z$ denotes the type of natural numbers below z. The expression Size is a possible instance of U in $\Pi^{\star}U\,T$, or a possible type of a variable in a typing context Γ, but not a first-class type, i. e., we cannot construct our own functions on sizes.

Canonical natural numbers c are constructed by $\mathsf{zero}\langle a\rangle$ and $\mathsf{suc}\langle a\rangle t$. A size expression a is either a constant $o \in \mathbb{N}$, a variable $\mathsf{v}_i + o$ possibly with increment o, or the limit ordinal ∞ which stands for ω. The size argument a in the constructors zero and suc is a suggestion for the type checker but bears no semantic significance. For example, in the declarative typing presented here, we can have $\vdash \mathsf{zero}\langle 5\rangle : \mathsf{Nat}\,1$. In the algorithmic typing however, $\vdash \mathsf{zero}\langle 5\rangle \leftleftarrows \mathsf{Nat}\,1$ will be an error. Note, however, that $\vdash \mathsf{zero}\langle a\rangle : \mathsf{Nat}\,0$ is impossible for any a, as zero is not strictly below 0 (when both term and size are interpreted as natural numbers).

Regular application $t\,u$, relevant size application $t\,a$, and irrelevant size application $t\,\langle a\rangle$ eliminate functions t and are subsumed under the form $t\,e$ with $e ::= u \mid a \mid \langle a\rangle$. We have two further eliminations, which make sense when t stands for a natural number. These are case distinction $e = \mathsf{case}_\ell\,T\,t_z\,t_s$ and recursive function application $e' = \mathsf{fix}_{\ell'}\,T'\,t'$. Application of case distinction $\mathsf{zero}\langle a\rangle\,e$ will reduce to the zero-branch t_z, and application $(\mathsf{suc}\langle a\rangle t)\,e$ to the instantiation $t_s\,t$ of the successor branch. The type annotation T in case allows us to infer the type of the whole case statement $t\,e$ as $T\,t$. The function call $c\,e'$ for a canonical number c and elimination $e' = \mathsf{fix}_{\ell'}\,T'\,t'$ reduces to $t'\,(\lambda x.\,x\,e')\,c$ where we allowed ourselves the use of a named abstraction in the presentation to the reader. The unfolding of fixed-points is thus restricted to application to canonical numbers; this is the usual reduction strategy which converges for terminating functions [Barthe et al. 2004].

For ordinary β-reduction we employ *substitutions* σ. These are simply lists of terms that provide one term as replacement for each free de Bruijn index in a term t. We write $\boxed{t\sigma}$ for the application of substitution σ to term t which is defined as usual. Let *lifting* $\boxed{\uparrow^k_m}$ be the substitution $(\mathsf{v}_{k+m-1}, \dots, \mathsf{v}_{k+1}, \mathsf{v}_k)$ which accepts a term with m free indices and increases each of them by k. We write $\uparrow_m$ for the lifting $\uparrow^1_m$ and $\boxed{\mathsf{id}_m}$ for the identity substitution $\uparrow^0_m$. In general, we refer to liftings by letter ξ. The substitution $\boxed{[u]_m} = (\mathsf{id}_m, u)$ replaces free index v_0 by term u and decrements the other m free indices by 1. We drop subscript m from liftings and substitutions when clear from the context. Substitution composition $\boxed{\sigma\tau}$ is the pointwise application of substitution τ to the list of terms σ. In the proofs to follow, we freely use the following identities:

$$
\begin{array}{ccccc}
t\,\mathsf{id} \equiv t & (t\sigma)\tau \equiv t(\sigma\tau) & \sigma\,\mathsf{id} \equiv \sigma & \mathsf{id}\,\tau \equiv \tau & (\rho\sigma)\tau \equiv \rho(\sigma\tau) \\
\mathsf{v}_0(\sigma, t) \equiv t & \uparrow(\sigma, t) \equiv \sigma & [t]\sigma \equiv (\sigma, t\sigma) & \uparrow[t] \equiv \mathsf{id} &
\end{array}
$$

As already done in some examples, we may use a named dependent function type notation as syntactic sugar for the corresponding de Bruijn representation. For instance, $(z\,{:}\,\mathsf{Size}) \to \mathsf{Nat}\,z \to \mathsf{Set}_\ell$ is sugar for $\Pi\,\mathsf{Size}\,\Pi\,(\mathsf{Nat}\ \mathsf{v}_0)\,\mathsf{Set}_\ell$. We abbreviate this type by $\boxed{\mathsf{FixK}\,\ell}$, and let $\boxed{\mathsf{FixT}\,T}$ stand for $\forall z.\,((x : \mathsf{Nat}\,z) \to T\,z\,x) \to (x : \mathsf{Nat}\,(z+1)) \to T\,(z+1)\,x$. Similarly to for Π, we use named lambda abstraction as sugar for de Bruijn abstraction. Named abstraction takes care of proper lifting of de Bruijn indices, for instance, $\lambda x.\,tx = \lambda.(t\uparrow)\,\mathsf{v}_0$ if t is outside the scope of x. We may also use names when we construct concrete contexts, for instance, if T is well-formed in context Γ, we may write $T\,z\,x$ in context $\Gamma.z\,{:}\,\mathsf{Size}.x\,{:}\,\mathsf{Nat}\,z$ to mean $T\uparrow^2 \mathsf{v}_1\,\mathsf{v}_0$ in context $\Gamma.\mathsf{Size}.\mathsf{Nat}\,\mathsf{v}_0$.

Inductively defined judgements (mutual).

$\vdash \Gamma$	context Γ is well-formed
$\Gamma(i) = {}^\star T$	in context Γ, de Bruijn index i has type T and annotation $\star$
$\Gamma \vdash a : \mathsf{Size}$	in context Γ, size expression a is well-formed
$\Gamma \vdash t : T$	in context Γ, term t has type T
$\Gamma \vdash t = t' : T$	in context Γ, terms t and t' are equal of type T
$\Gamma \vdash T \leq T'$	in context Γ, type T is a subtype of T'
$\Gamma \vdash \sigma : \Delta$	σ is a valid substitution for Δ
$\Gamma \vdash \sigma = \sigma' \mathrel{\dot{=}} \tau : \Delta$	$\sigma/\sigma'/\tau$ are a equal term/term/type-level substitutions for Δ

Derived judgements.

$\Gamma \vdash T$	$:\Longleftrightarrow$	$\Gamma \vdash T : s$ for some s
$\Gamma \vdash T = T'$	$:\Longleftrightarrow$	$\Gamma \vdash T = T' : s$ for some s
$\Gamma \vdash a = b : \mathsf{Size}$	$:\Longleftrightarrow$	$\Gamma \vdash a : \mathsf{Size}$ and $a = b$
$\Gamma \vdash a \leq b : \mathsf{Size}$	$:\Longleftrightarrow$	$\Gamma \vdash a : \mathsf{Size}$ and $\Gamma \vdash b : \mathsf{Size}$ and $a \leq b$
$\Gamma \vdash T : \mathsf{Adm}\,\ell$	$:\Longleftrightarrow$	$\Gamma \vdash T : \mathsf{FixK}\,\ell$ and $\Gamma.z{:}\mathsf{Size}.x{:}\mathsf{Nat}\,z \vdash T\,z\,x \leq T\,\infty\,x$
$\xi : \Gamma \leq \Delta$	$:\Longleftrightarrow$	$\Gamma \vdash \xi : \Delta$ and $\xi = \uparrow^k_m$ with $m = \lvert\Delta\rvert$ and $k = \lvert\Gamma\rvert - m$

Fig. 2. Judgements.

In typing contexts Γ, we distinguish relevant (:) and irrelevant ($\div$) bindings. When type checking a variable, it needs to be bound in the context relevantly. However, when entering an irrelevant position, for instance when checking size a in term $\mathsf{suc}\langle a\rangle t$ we declare previously irrelevant variables as relevant. This operation on the context has been coined *resurrection* by Pfenning [2001]; formally $\boxed{\Gamma^\oplus}$ removes the "$\div$"-markers from all bindings in Γ, i. e., replaces them by ":"-markers. Note that, trivially, resurrection is idempotent: $\Gamma^{\oplus\oplus} = \Gamma^\oplus$.

Size increment $\boxed{a + o'}$ for $o' \in \mathbb{N}$ extends addition by $\infty + o' = \infty$ and $(\mathsf{v}_i + o) + o' = \mathsf{v}_i + (o + o')$. Sizes are partially ordered; size comparison $\boxed{a \leq b}$ holds as expected if either $b = \infty$ or $o \leq o'$ where either $a = o$ and $b = o'$ or $a \in \{o, \mathsf{v}_i + o\}$ and $b = \mathsf{v}_i + o'$. Different size variables are incomparable.

Fig. 2 lists the inductive and derived judgements of our type theory and figures 3 and 4 the inference rules. We have $\boxed{\text{boxed}}$ the rules dealing with irrelevant size application. Fig. 5 adds the typing and equality rules for case distinction and recursion on natural numbers. Judgement $\Gamma \vdash T : \mathsf{Adm}\,\ell$ characterizes the valid type annotations T in recursion $\mathsf{fix}_\ell\,T\,t$. The type constructor T has to be monotone in the size argument; this is a technical condition for type-based termination [Barthe et al. 2004]. We will make use of it in Section 4.7. We write $\boxed{\mathcal{D} :: J}$ to express that $\mathcal{D}$ is a derivation of judgement J.

In the typing judgement $\Gamma \vdash t : T$, the term t is in scope of Γ, i. e., may not mention irrelevant variables in relevant positions. However, the type T is in scope of the resurrected context $\Gamma^\oplus$, hence, can mention all variables declared in Γ. The other judgements are organized similarly. To understand this distinction, consider judgement $z \div \mathsf{Size} \vdash \mathsf{Nat}\,z$. This would mean that z is irrelevant in $\mathsf{Nat}\,z$ and thus, $\Gamma \vdash \mathsf{Nat}\,a = \mathsf{Nat}\,a'$ for all sizes $\Gamma^\oplus \vdash a, a' : \mathsf{Size}$. But this is exactly wrong! However, judgement $z \div \mathsf{Size} \vdash \mathsf{zero}\langle z\rangle : \mathsf{Nat}\,(z+1)$ is fine, it implies $\Gamma \vdash \mathsf{zero}\langle a\rangle = \mathsf{zero}\langle a'\rangle : \mathsf{Nat}\,(b+1)$ for all $\Gamma^\oplus \vdash a, a', b : \mathsf{Size}$.

$\boxed{\vdash \Gamma}$ (implies $\vdash \Gamma^{\oplus}$) and $\boxed{\Gamma(i) = {}^{\star}T}$ (implies $\Gamma^{\oplus} \vdash T$ if $\vdash \Gamma$).

$$\frac{}{\vdash ()} \qquad \frac{\vdash \Gamma \qquad \Gamma^{\oplus} \vdash T}{\vdash \Gamma.T} \qquad \frac{\vdash \Gamma}{\vdash \Gamma.{}^{\star}\mathsf{Size}} \qquad \frac{}{(\Gamma.{}^{\star}T)(0) = {}^{\star}T\uparrow} \qquad \frac{\Gamma(i) = {}^{\star}T}{(\Gamma.\_)(i+1) = {}^{\star}T\uparrow}$$

$\boxed{\Gamma \vdash a : \mathsf{Size}}$ (implies $\vdash \Gamma$ and $\Gamma^{\oplus} \vdash a : \mathsf{Size}$).

$$\frac{\vdash \Gamma}{\Gamma \vdash \infty : \mathsf{Size}} \qquad \frac{\vdash \Gamma}{\Gamma \vdash o : \mathsf{Size}}\; o \in \mathbb{N} \qquad \frac{\vdash \Gamma \qquad \Gamma(i) = {}^{:}\mathsf{Size}}{\Gamma \vdash \mathsf{v}_i + o : \mathsf{Size}}\; o \in \mathbb{N}$$

$\boxed{\Gamma \vdash t : T}$ (implies $\vdash \Gamma$ and $\Gamma^{\oplus} \vdash T$ [and $\Gamma^{\oplus} \vdash t : T$]. Note: no rule for $\Gamma \vdash \mathsf{Size} : s$.)

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathsf{Set}_{\ell} : \mathsf{Set}_{\ell'}}\; \ell < \ell' \qquad \frac{\Gamma \vdash U : s \qquad \Gamma.U \vdash T : s}{\Gamma \vdash \Pi\, U\, T : s} \qquad \frac{\Gamma.\mathsf{Size} \vdash T : s}{\Gamma \vdash \Pi^{\star} \mathsf{Size}\, T : s} \qquad \frac{\Gamma \vdash a : \mathsf{Size}}{\Gamma \vdash \mathsf{Nat}\, a : \mathsf{Set}_0}$$

$$\frac{\vdash \Gamma \qquad \Gamma(i) = {}^{:}T}{\Gamma \vdash \mathsf{v}_i : T}\; T \neq \mathsf{Size} \qquad \frac{\Gamma.{}^{\star}U \vdash t : T}{\Gamma \vdash \lambda t : \Pi^{\star} U\, T} \qquad \frac{\Gamma \vdash t : \Pi\, U\, T \qquad \Gamma \vdash u : U}{\Gamma \vdash t\, u : T[u]}$$

$$\frac{\Gamma \vdash t : \Pi\, \mathsf{Size}\, T \qquad \Gamma \vdash a : \mathsf{Size}}{\Gamma \vdash t\, a : T[a]} \qquad \boxed{\frac{\Gamma \vdash t : \forall T \qquad \Gamma^{\oplus} \vdash a, b : \mathsf{Size}}{\Gamma \vdash t\langle a \rangle : T[b]}}$$

$$\frac{\Gamma^{\oplus} \vdash a, b : \mathsf{Size}}{\Gamma \vdash \mathsf{zero}\langle a \rangle : \mathsf{Nat}\,(b+1)} \qquad \frac{\Gamma^{\oplus} \vdash a : \mathsf{Size} \qquad \Gamma \vdash t : \mathsf{Nat}\, b}{\Gamma \vdash \mathsf{suc}\langle a \rangle t : \mathsf{Nat}\,(b+1)} \qquad \frac{\Gamma \vdash t : T \qquad \Gamma^{\oplus} \vdash T \leq T'}{\Gamma \vdash t : T'}$$

$\boxed{\Gamma \vdash T \leq T'}$ (implies $\Gamma \vdash T, T'$)

$$\frac{\vdash \Gamma \qquad \ell \leq \ell'}{\Gamma \vdash \mathsf{Set}_{\ell} \leq \mathsf{Set}_{\ell'}} \qquad \frac{\Gamma \vdash a \leq b : \mathsf{Size}}{\Gamma \vdash \mathsf{Nat}\, a \leq \mathsf{Nat}\, b} \qquad \frac{\Gamma \vdash T = T'}{\Gamma \vdash T \leq T'}$$

$$\frac{\Gamma \vdash U' \leq U \qquad \Gamma.U' \vdash T \leq T'}{\Gamma \vdash \Pi\, U\, T \leq \Pi\, U'\, T'} \qquad \frac{\Gamma.\mathsf{Size} \vdash T \leq T'}{\Gamma \vdash \Pi^{\star} \mathsf{Size}\, T \leq \Pi^{\star} \mathsf{Size}\, T'} \qquad \frac{\Gamma \vdash T_1 \leq T_2 \qquad \Gamma \vdash T_2 \leq T_3}{\Gamma \vdash T_1 \leq T_3}$$

$\boxed{\Gamma \vdash \tau : \Delta}$ (implies $\vdash \Gamma$ and $\vdash \Delta$ [and $\Gamma^{\oplus} \vdash \tau : \Delta$] and $\Gamma^{\oplus} \vdash \tau : \Delta^{\oplus}$).

$$\frac{\vdash \Gamma}{\Gamma \vdash () : ()} \qquad \frac{\Gamma \vdash \tau : \Delta \qquad \Delta^{\oplus} \vdash T \qquad \Gamma \vdash t : T\tau}{\Gamma \vdash (\tau, t) : \Delta.T}$$

$$\frac{\Gamma \vdash \tau : \Delta \qquad \Gamma \vdash a : \mathsf{Size}}{\Gamma \vdash (\tau, a) : \Delta.\mathsf{Size}} \qquad \boxed{\frac{\Gamma \vdash \tau : \Delta \qquad \Gamma^{\oplus} \vdash a : \mathsf{Size}}{\Gamma \vdash (\tau, a) : \Delta.{}^{\div}\mathsf{Size}}}$$

$\boxed{\Gamma \vdash \sigma = \sigma' \risingdotseq \tau : \Delta}$ (implies $\vdash \Gamma$ and $\vdash \Delta$ and $\Gamma \vdash \tau : \Delta$ and $\Gamma \vdash \sigma' = \sigma \risingdotseq \tau : \Delta$).

$$\frac{\vdash \Gamma}{\Gamma \vdash () = () \risingdotseq () : ()} \qquad \frac{\Gamma \vdash \sigma = \sigma' \risingdotseq \tau : \Delta \qquad \Delta^{\oplus} \vdash T \qquad \Gamma \vdash u = u' = t : T\tau}{\Gamma \vdash (\sigma, u) = (\sigma', u') \risingdotseq (\tau, t) : \Delta.T}$$

$$\frac{\Gamma \vdash \sigma = \sigma' \risingdotseq \tau : \Delta \qquad \Gamma \vdash a = a' = b : T\tau}{\Gamma \vdash (\sigma, a) = (\sigma', a') \risingdotseq (\tau, b) : \Delta.\mathsf{Size}} \qquad \boxed{\frac{\Gamma \vdash \sigma = \sigma' \risingdotseq \tau : \Delta \qquad \Gamma^{\oplus} \vdash a, a', b : T\tau}{\Gamma \vdash (\sigma, a) = (\sigma', a') \risingdotseq (\tau, b) : \Delta.{}^{\div}\mathsf{Size}}}$$

Fig. 3. Typing, subtyping, and substitution judgements.

Computation rules.

$$\frac{\Gamma.U \vdash t : T \qquad \Gamma \vdash u : U}{\Gamma \vdash (\lambda t)\, u = t[u] : T[u]} \qquad \frac{\Gamma.\mathsf{Size} \vdash t : T \qquad \Gamma \vdash a : \mathsf{Size}}{\Gamma \vdash (\lambda t)\, a = t[a] : T[a]} \qquad \boxed{\frac{\Gamma.^{\div}\mathsf{Size} \vdash t : T \qquad \Gamma^{\oplus} \vdash a, b : \mathsf{Size}}{\Gamma \vdash (\lambda t)\, \langle a \rangle = t[a] : T[b]}}$$

Extensionality rules.

$$\frac{\Gamma \vdash t : \Pi\, U\, T}{\Gamma \vdash t = \lambda x.t\, x : \Pi\, U\, T} \qquad \boxed{\frac{\Gamma \vdash t : \forall T \qquad \Gamma^{\oplus}.\mathsf{Size} \vdash a : \mathsf{Size}}{\Gamma \vdash t = \lambda x.t\langle a \rangle : \forall T}}$$

Congruence rules.

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathsf{Set}_\ell = \mathsf{Set}_\ell : \mathsf{Set}_{\ell'}}\, \ell < \ell' \qquad \frac{\Gamma \vdash a : \mathsf{Size}}{\Gamma \vdash \mathsf{Nat}\, a = \mathsf{Nat}\, a : \mathsf{Set}_0}$$

$$\frac{\Gamma \vdash U = U' : s \qquad \Gamma.U \vdash T = T' : s}{\Gamma \vdash \Pi\, U\, T = \Pi\, U'\, T' : s} \qquad \frac{\Gamma.\mathsf{Size} \vdash T = T' : s}{\Gamma \vdash \Pi^{\star}\mathsf{Size}\, T = \Pi^{\star}\mathsf{Size}\, T' : s}$$

$$\frac{\vdash \Gamma \qquad \Gamma(i) = {}^{:}T}{\Gamma \vdash \mathsf{v}_i = \mathsf{v}_i : T}\, T \neq \mathsf{Size} \qquad \frac{\Gamma.^{\star}U \vdash t = t' : T}{\Gamma \vdash \lambda t = \lambda t' : \Pi^{\star}U\, T} \qquad \frac{\Gamma \vdash t = t' : \Pi\, U\, T \qquad \Gamma \vdash u = u' : U}{\Gamma \vdash t\, u = t'\, u' : T[u]}$$

$$\frac{\Gamma \vdash t = t' : \Pi\, \mathsf{Size}\, T \qquad \Gamma \vdash a : \mathsf{Size}}{\Gamma \vdash t\, a = t'\, a : T[a]} \qquad \boxed{\frac{\Gamma \vdash t = t' : \forall T \qquad \Gamma^{\oplus} \vdash a, a', b : \mathsf{Size}}{\Gamma \vdash t\langle a \rangle = t'\langle a' \rangle : T[b]}}$$

$$\frac{\Gamma^{\oplus} \vdash a, a', b : \mathsf{Size}}{\Gamma \vdash \mathsf{zero}\langle a \rangle = \mathsf{zero}\langle a' \rangle : \mathsf{Nat}\,(b+1)} \qquad \frac{\Gamma^{\oplus} \vdash a, a' : \mathsf{Size} \qquad \Gamma \vdash t = t' : \mathsf{Nat}\, b}{\Gamma \vdash \mathsf{suc}\langle a \rangle t = \mathsf{suc}\langle a' \rangle t' : \mathsf{Nat}\,(b+1)}$$

$$\frac{\Gamma \vdash t = t' : T \qquad \Gamma^{\oplus} \vdash T \leq T'}{\Gamma \vdash t = t' : T'}$$

Equivalence rules.

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash t = t : T} \qquad \frac{\Gamma \vdash t = t' : T}{\Gamma \vdash t' = t : T} \qquad \frac{\Gamma \vdash t_1 = t_2 : T \qquad \Gamma \vdash t_2 = t_3 : T}{\Gamma \vdash t_1 = t_3 : T}$$

Fig. 4. Definitional equality $\boxed{\Gamma \vdash t = t' : T}$ (implies $\Gamma^{\oplus} \vdash T$ and $\Gamma \vdash t, t' : T$ [and $\Gamma^{\oplus} \vdash t = t' : T$]).

Our substitution theorem needs to reflect the distinct scope of things left of the colon vs. things right of the colon. In the last example we have applied the substitution triple $\Gamma \vdash [a] = [a'] \fallingdotseq [b] : (z \div \mathsf{Size})$ to judgement $z \div \mathsf{Size} \vdash \mathsf{zero}\langle z \rangle : \mathsf{Nat}\,(z+1)$. The first two substitutions apply to the term side while the third substitution applies to the type side. The fact that we replace an irrelevant variable z allows a, a', b to refer to irrelevant variables from Γ, thus, they are in scope of $\Gamma^{\oplus}$.

Typing requires from annotations $\langle a \rangle$ in a term only that they are well-scoped size expressions, i. e., just mention relevant size variables. Let $\boxed{t^{\infty}}$ denote the *erasure* of term t, meaning that we replace all annotations $\langle a \rangle$ in t by $\langle \infty \rangle$. Let $\boxed{t \approx u}$ relate terms that only differ in their annotations, i. e., $t \approx u :\Longleftrightarrow t^{\infty} = u^{\infty}$. Erasure does not change the term modulo judgmental equality:

Lemma 3.1 (Erasure and similarity).

(1) *If* $\Gamma \vdash t : T$ *then* $\Gamma \vdash t = t^{\infty} : T$.
(2) *If* $\Gamma \vdash t, u : T$ *and* $t \approx u$ *then* $\Gamma \vdash t = u : T$.

Case distinction.

$$\frac{\Gamma \vdash u : \mathsf{Nat}\,(a{+}1) \qquad \Gamma \vdash t_z : T\,(\mathsf{zero}\langle a\rangle) \qquad \Gamma \vdash t_s : (x : \mathsf{Nat}\,a) \to T\,(\mathsf{suc}\langle a\rangle x) \qquad \Gamma^{\oplus} \vdash T : \mathsf{Nat}\,(a{+}1) \to \mathsf{Set}_\ell}{\Gamma \vdash u\,\mathsf{case}_\ell\,T\,t_z\,t_s : T\,u}$$

$$\frac{\Gamma^{\oplus} \vdash T = T' : \mathsf{Nat}\,(a+1) \to \mathsf{Set}_\ell \qquad \Gamma \vdash u = u' : \mathsf{Nat}\,(a+1) \qquad \Gamma \vdash t_z = t_z' : T\,(\mathsf{zero}\langle a\rangle) \qquad \Gamma \vdash t_s = t_s' : (x : \mathsf{Nat}\,a) \to T\,(\mathsf{suc}\langle a\rangle x)}{\Gamma \vdash u\,\mathsf{case}_\ell\,T\,t_z\,t_s = u'\,\mathsf{case}_\ell\,T'\,t_z'\,t_s' : T\,u}$$

$$\frac{\Gamma^{\oplus} \vdash a, b : \mathsf{Size} \qquad \Gamma^{\oplus} \vdash T : \mathsf{Nat}\,(b+1) \to \mathsf{Set}_\ell \qquad \Gamma \vdash t_z : T\,(\mathsf{zero}\langle b\rangle) \qquad \Gamma \vdash t_s : (x : \mathsf{Nat}\,b) \to T\,(\mathsf{suc}\langle b\rangle x)}{\Gamma \vdash (\mathsf{zero}\langle a\rangle)\,\mathsf{case}_\ell\,T\,t_z\,t_s = t_z : T\,\mathsf{zero}\langle b\rangle}$$

$$\frac{\Gamma^{\oplus} \vdash a : \mathsf{Size} \quad \Gamma \vdash t : \mathsf{Nat}\,b \quad \Gamma^{\oplus} \vdash T : \mathsf{Nat}\,(b+1) \to \mathsf{Set}_\ell \quad \Gamma \vdash t_z : T\,(\mathsf{zero}\langle b\rangle) \quad \Gamma \vdash t_s : (x : \mathsf{Nat}\,b) \to T\,(\mathsf{suc}\langle b\rangle x)}{\Gamma \vdash (\mathsf{suc}\langle a\rangle t)\,\mathsf{case}_\ell\,T\,t_z\,t_s = t_s\,t : T\,(\mathsf{suc}\langle b\rangle t)}$$

Recursion.

$$\frac{\Gamma \vdash u : \mathsf{Nat}\,a \qquad \Gamma^{\oplus} \vdash T : \mathsf{Adm}\ \ell \qquad \Gamma \vdash t : \mathsf{FixT}\,T}{\Gamma \vdash u\,\mathsf{fix}_\ell\,T\,t : T\,a\,u}$$

$$\frac{\Gamma \vdash u = u' : \mathsf{Nat}\,a \qquad \Gamma^{\oplus} \vdash T = T' : \mathsf{Adm}\ \ell \qquad \Gamma \vdash t = t' : \mathsf{FixT}\,T}{\Gamma \vdash u\,\mathsf{fix}_\ell\,T\,t = u'\,\mathsf{fix}_\ell\,T'\,t' : T\,a\,u}$$

$$\frac{\Gamma \vdash c : \mathsf{Nat}\,b \qquad \Gamma^{\oplus} \vdash a : \mathsf{Size} \qquad \Gamma^{\oplus} \vdash T : \mathsf{Adm}\ \ell \qquad \Gamma \vdash t : \mathsf{FixT}\,T}{\Gamma \vdash c\,\mathsf{fix}_\ell\,T\,t = t\langle a\rangle(\lambda x.x\,\mathsf{fix}_\ell\,T\,t)\,c : T\,b\,c}$$

Fig. 5. Rules for case distinction and recursion.

We should remark here that we have *neither type unicity nor principal types* due to the irrelevant size application rule. In the following, we list syntactic properties of our judgements. To this end, let J match a part of a judgement.

Lemma 3.2 (Context well-formedness).

(1) *If* $\vdash \Gamma.\Delta$ *then* $\vdash \Gamma$
(2) *If* $\Gamma \vdash J$ *then* $\vdash \Gamma$.

All types in a context are considered in the resurrected context, which justifies the first statement of the following lemma. A resurrected context is more permissive, as it brings more variable into scope. As such, it is comparable to an extended context or a context where types have been replaced by subtypes. This intuition accounts for the remaining statements but (4). The latter is a defining property of substitutions: only replacement for irrelevant sizes may refer to irrelevant size variables.

Lemma 3.3 (Resurrection).

(1) $\vdash \Gamma$ *iff* $\vdash \Gamma^{\oplus}$. *Then* $\Gamma^{\oplus} \vdash \mathsf{id} : \Gamma$, *which can be written* $\mathsf{id} : \Gamma^{\oplus} \leq \Gamma$.
(2) *If* $\Gamma \vdash J$ *then* $\Gamma^{\oplus} \vdash J$.
(3) *If* $\Gamma \vdash \sigma : \Delta^{\oplus}$ *then* $\Gamma \vdash \sigma : \Delta$.

(4) *If* $\Gamma \vdash \sigma : \Delta$ *then* $\Gamma^{\oplus} \vdash \sigma : \Delta^{\oplus}$.

Lemma 3.4 (Substitution).

(1) *If* $\Gamma \vdash \sigma : \Delta$ *and* $\Delta \vdash J$ *then* $\Gamma \vdash J\sigma$.
(2) *If* $\Gamma \vdash \sigma = \sigma' \risingdotseq \tau : \Delta$ *and* $\Delta \vdash t : T$ *then* $\Gamma \vdash t\sigma : T\tau$ *and* $\Gamma \vdash t\sigma' : T\tau$.

Lemma 3.5 (Specific substitutions).

(1) *If* $\vdash \Gamma.\Delta$ *then* $\Gamma.\Delta \vdash \uparrow^{|\Delta|}_{|\Gamma|} : \Gamma$. *If* $\vdash \Gamma.T$ *then* $\Gamma.T \vdash \uparrow : \Gamma$.
(2) *If* $\vdash \Gamma$ *then* $\Gamma \vdash \mathsf{id} : \Gamma$.
(3) *If* $\Gamma \vdash u : U$ *then* $\Gamma \vdash [u] : \Gamma.U$.

The relation $\Gamma \vdash \sigma = \sigma' \risingdotseq \tau : \Delta$ is a partial equivalence relation (PER) on term-side substitutions σ, σ'. Note that usually we cannot resurrect this judgement to $\Gamma^{\oplus} \vdash \sigma = \sigma' \risingdotseq \tau : \Delta^{\oplus}$. For instance, $z_1 \div \mathsf{Size}.\, z_2 \div \mathsf{Size} \vdash [z_1] = [z_2] \risingdotseq [\infty] : z \div \mathsf{Size}$ holds but $z_1{:}\mathsf{Size}.\, z_2{:}\mathsf{Size} \vdash [z_1] = [z_2] \risingdotseq [\infty] : z{:}\mathsf{Size}$ clearly not.

Lemma 3.6 (Substitution equality).

(1) *Conversion: If* $\Gamma \vdash \sigma = \sigma' \risingdotseq \tau_1 : \Delta$ *and* $\Gamma^{\oplus} \vdash \tau_1 = \tau_2 \risingdotseq \tau : \Delta^{\oplus}$ *then* $\Gamma \vdash \sigma = \sigma' \risingdotseq \tau_2 : \Delta$.
(2) *Reflexivity: If* $\Gamma \vdash \sigma : \Delta$ *then* $\Gamma \vdash \sigma = \sigma \risingdotseq \sigma : \Delta$.
(3) *Symmetry: If* $\Gamma \vdash \sigma = \sigma' \risingdotseq \tau : \Delta$ *then* $\Gamma \vdash \sigma' = \sigma \risingdotseq \tau : \Delta$.
(4) *Transitivity: If* $\Gamma \vdash \sigma_1 = \sigma_2 \risingdotseq \tau : \Delta$ *and* $\Gamma \vdash \sigma_2 = \sigma_3 \risingdotseq \tau : \Delta$ *then* $\Gamma \vdash \sigma_1 = \sigma_3 \risingdotseq \tau : \Delta$.
(5) *Functionality: Let* $\Gamma \vdash \sigma = \sigma' \risingdotseq \tau : \Delta$.
  (a) *If* $\Delta \vdash t : T$ *then* $\Gamma \vdash t\sigma = t\sigma' : T\tau$.
  (b) *If* $\Delta \vdash t = t' : T$ *then* $\Gamma \vdash t\sigma = t'\sigma' : T\tau$.
  (c) *Corollary: If* $\Delta \vdash T \leq T'$ *then* $\Gamma \vdash T\sigma \leq T\sigma'$.

Corollary 3.7 (Partial resurrection for substitution equality). *If* $\Gamma \vdash \sigma = \sigma' \risingdotseq \tau : \Delta$ *then* $\Gamma^{\oplus} \vdash \tau = \tau \risingdotseq \tau : \Delta^{\oplus}$.

Lemma 3.8 (Inversion of typing).

(1) *If* $\Gamma \vdash \mathsf{Nat}\, a : T'$ *then* $\Gamma \vdash a : \mathsf{Size}$ *and* $\Gamma^{\oplus} \vdash \mathsf{Set}_0 \leq T'$.
(2) *If* $\Gamma \vdash \mathsf{Set}_\ell : T'$ *then* $\Gamma^{\oplus} \vdash \mathsf{Set}_{\ell+1} \leq T'$.
(3) *If* $\Gamma \vdash \Pi\, U\, T : T'$ *then* $\Gamma \vdash U : s$ *and* $\Gamma.U \vdash T : s$ *and* $\Gamma^{\oplus} \vdash s \leq T'$ *for some* s.
(4) *If* $\Gamma \vdash \Pi^{\star}\mathsf{Size}\, T : T'$ *then* $\Gamma.\mathsf{Size} \vdash T : s$ *and* $\Gamma^{\oplus} \vdash s \leq T'$.
(5) *If* $\Gamma \vdash \mathsf{v}_i : T'$ *then* $\Gamma(i) = {}^{:}T$ *and* $\Gamma^{\oplus} \vdash T \leq T'$ *for some* T.
(6) *If* $\Gamma \vdash \lambda t : T'$ *then either* $\Gamma.U \vdash t : T$ *and* $\Gamma^{\oplus} \vdash \Pi\, U\, T \leq T'$ *for some* U, T *or* $\Gamma.^{\star}\mathsf{Size} \vdash t : T$ *and* $\Gamma^{\oplus} \vdash \Pi^{\star}\mathsf{Size}\, T \leq T'$ *for some* T.
(7) *If* $\Gamma \vdash t\, u : T'$ *then* $\Gamma \vdash t : \Pi\, U\, T$ *and* $\Gamma \vdash u : U$ *and* $\Gamma^{\oplus} \vdash T[u] \leq T'$ *for some* U, T.
(8) *If* $\Gamma \vdash t\, a : T'$ *then* $\Gamma \vdash t : \Pi\,\mathsf{Size}\, T$ *and* $\Gamma \vdash a : \mathsf{Size}$ *and* $\Gamma^{\oplus} \vdash T[a] \leq T'$ *for some* T.
(9) *If* $\Gamma \vdash t\,\langle a\rangle : T'$ *then* $\Gamma \vdash t : \forall\, T$ *and* $\Gamma^{\oplus} \vdash a, b : \mathsf{Size}$ *and* $\Gamma^{\oplus} \vdash T[b] \leq T'$ *for some* T, b.
(10) *If* $\Gamma \vdash \mathsf{zero}\langle a\rangle : T'$ *then* $\Gamma^{\oplus} \vdash a, b : \mathsf{Size}$ *and* $\Gamma^{\oplus} \vdash \mathsf{Nat}\,(b+1) \leq T'$ *for some* b.
(11) *If* $\Gamma \vdash \mathsf{suc}\langle a\rangle t : T'$ *then* $\Gamma^{\oplus} \vdash a, b : \mathsf{Size}$ *and* $\Gamma \vdash t : \mathsf{Nat}\, b$ *and* $\Gamma^{\oplus} \vdash \mathsf{Nat}\,(b+1) \leq T'$.
(12) *If* $\Gamma \vdash u\,\mathsf{case}_\ell\, T\, t_z\, t_s : T'$ *then* $\Gamma \vdash u : \mathsf{Nat}\,(a+1)$ *and* $\Gamma \vdash T : \mathsf{Nat}\,(a+1) \to \mathsf{Set}_\ell$ *and* $\Gamma \vdash t_z : T\,(\mathsf{zero}\langle a\rangle)$ *and* $\Gamma \vdash t_s : (x{:}\mathsf{Nat}\, a) \to T\,(\mathsf{suc}\langle a\rangle x)$ *and* $\Gamma^{\oplus} \vdash T\, u \leq T'$ *for some* a.
(13) *If* $\Gamma \vdash u\,\mathsf{fix}_\ell\, T\, t : T'$ *then* $\Gamma \vdash u : \mathsf{Nat}\, a$ *and* $\Gamma \vdash T : \mathsf{Adm}\ \ell$ *and* $\Gamma \vdash t : \mathsf{FixT}\, T$ *and* $\Gamma^{\oplus} \vdash T\, a\, u \leq T'$.

Proof. Each by induction on the typing derivation, gathering applications of the conversion rule via transitivity of subtyping. □

4 SEMANTICS AND COMPLETENESS OF NORMALIZATION BY EVALUATION

In this section we present an operational semantics of our language, define the NbE algorithm, construct a PER model, and demonstrate that NbE is complete for definitional equality, i. e., if $\Gamma \vdash t = t' : T$, then t and t' have the same normal form up to annotations.

$$\begin{array}{llll} \mathsf{Ne} \ni m & ::= \mathsf{v}_i \mid m\,v \mid m\,a \mid m\,\langle a\rangle \mid m\,\mathsf{case}_\ell\,V\,v_z\,v_s \mid m\,\mathsf{fix}_\ell\,V\,v & & \text{neutral n.f.} \\ \mathsf{Nf} \ni v & ::= m \mid \lambda v \mid \mathsf{zero}\langle a\rangle \mid \mathsf{suc}\langle a\rangle v \mid \mathsf{Set}_\ell \mid \mathsf{Nat}\,a \mid \Pi\,V_u\,V_t \mid \Pi^\star\mathsf{Size}\,V & & \text{normal form} \end{array}$$

For the operational semantics, instead of defining a separate language of values, we extend the syntax of expressions by de Bruijn levels x_k to be used as generic values (unknowns), and type annotations $\uparrow^A n$ and $\downarrow^A t$ for lazy realizations of the reflection and reification operations of NbE. *Terms* are expressions that do not contain these new expression forms. *Values* $\boxed{f, g, A, B, F \in \mathsf{D}}$ are expressions with no free de Bruijn indices, where each neutral n is under a reflection marker $\uparrow^A n$. The types A that direct reflection $\uparrow^A n$ and reification $\downarrow^A f$ also live in the value world.

$$\begin{array}{llllll} \mathsf{NeExp} & \ni & n & ::= & \cdots \mid \mathsf{x}_k & \text{de Bruijn level } k \\ \mathsf{Up} & \ni & N & ::= & \uparrow^A n & \text{reflection of neutral term } n \text{ as value of type } A \\ \mathsf{Whnf} & \ni & w & ::= & \cdots \mid N & \text{reflected neutral is weak head normal} \\ \mathsf{Exp} & \ni & t & ::= & \cdots \mid \downarrow^A f & \text{reification of value } f \text{ at type } A \end{array}$$

De Bruijn levels are the mirror images of de Bruijn indices. While de Bruijn indices index the context from the right, i. e., v_0 refers the last type that entered the context, de Bruijn levels index it from the left, i. e., x_0 refers to the first type in the context. This way, de Bruijn levels are stable under context extensions, and suitable to represent unknowns.

Size values $\boxed{\alpha, \beta \in \mathcal{S}ize}$ are size expressions that use de Bruijn levels instead of de Bruijn indices. Comparison of size values $\alpha \leq \beta$ is analogous to comparison of size terms $a \leq b$. In the following, we will reuse letter a for a value if it cannot be confused for a size term.

Finally, we identify two expression classes for NbE. Neutrals $n \in \mathsf{DNe}$ are the ones that will appear in values under the reflection marker $\uparrow^A$. Reified values $d \in \mathsf{DNf}$ are values under a reification marker $\downarrow^A$.

$$\begin{array}{lllll} \mathsf{DNe} & \ni n & ::= & \mathsf{v}_i \mid n\,d \mid n\,\alpha \mid n\,\langle\alpha\rangle \mid n\,\mathsf{case}_\ell\,D\,d_z\,d_s \mid n\,\mathsf{fix}_\ell\,D\,d & \text{unreflected neutral value} \\ \mathsf{DNf} & \ni d & ::= & \downarrow^A f & \text{reified value} \end{array}$$

Figure 6, adapted from Abel [2013] summarized the syntactic categories and main operations involved in NbE in what is called *locally nameless style*. The red path $\mathsf{Exp} \to \mathsf{D} \to \mathsf{D}_{\mathsf{nf}} \to \mathsf{Nf}$ decomposes $\beta\eta$-normalization into three steps.

(1) First, we close the term t with an environment η that maps the free de Bruijn indices of t to reflected de Bruijn levels. Reflection of de Bruijn levels follows the blue path Level $\to$ DNe $\to$ D: Levels embed via constructor x into semantic neutrals DNe which are labeled with their type $A \in \mathsf{D}$ to become an element $\uparrow^A \mathsf{x}_j \in \mathsf{D}$.
(2) Then, we label value $t\eta \in \mathsf{D}$ with its type A to obtain $\downarrow^A t\eta \in \mathsf{DNf}$.
(3) Finally, *read back* $\mathsf{R}_k \downarrow^A t\eta$ produces a long normal form $v \in \mathsf{Nf}$, converting de Bruijn levels back to indices. Herein, k should be the length of the context the original term t lived in. If this is the case, each de Bruijn level encountered during read back is below k and can be safely converted to a de Bruijn index.

4.1 Weak head reduction

We define the operational semantics of our language by the weak head evaluation relation $\boxed{t \searrow w}$ which is defined on expressions, thus works on values as well as on terms. It is defined mutually

$$
\begin{tikzcd}
\text{Semantics }(\beta) & & f \in \mathsf{D} \arrow[d, "\downarrow^A"'] & & \\
\text{Semantics }(\beta\eta) & & d \in \mathsf{DNf} \arrow[d, "\mathsf{R}_k"'] & n \in \mathsf{DNe} \arrow[ul, "\uparrow^A"'] \arrow[d, "\mathsf{R}^{\mathsf{ne}}_k"'] & j \in \mathsf{Level} \arrow[l, "\mathsf{x}"'] \arrow[d, leftrightarrow, "k \dot{-} (1+\_)"'] \\
\text{Syntax} & t \in \mathsf{Exp} \arrow[uur, bend left, "\_^{\eta}"] & v \in \mathsf{Nf} \arrow[l, "\supseteq"'] & m \in \mathsf{Ne} \arrow[l, "\supseteq"'] & i \in \mathsf{Index} \arrow[l, "\mathsf{v}"']
\end{tikzcd}
$$

Fig. 6. Type-assignment NbE in locally nameless style.

with auxiliary relation $\boxed{w \mathbin{@} e \searrow w'}$ stating that weak head normal form w is eliminated by e into weak head normal form w'.

$\boxed{t \searrow w}$ and $\boxed{w \mathbin{@} e \searrow w'}$

$$
\frac{}{w \searrow w}
\qquad
\frac{t \searrow w \qquad w \mathbin{@} e \searrow w'}{t\, e \searrow w'}
$$

$$
\frac{t[u] \searrow w}{(\lambda t) \mathbin{@} u \searrow w}
\qquad
\frac{t[\alpha] \searrow w}{(\lambda t) \mathbin{@} \alpha \searrow w}
\qquad
\frac{t[\alpha] \searrow w}{(\lambda t) \mathbin{@} \langle\alpha\rangle \searrow w}
\qquad
\frac{t_z \searrow w}{(\mathsf{zero}\langle\alpha\rangle) \mathbin{@} \mathsf{case}_\ell\, T\, t_z\, t_s \searrow w}
$$

$$
\frac{t_s\, t \searrow w}{(\mathsf{suc}\langle\alpha\rangle t) \mathbin{@} \mathsf{case}_\ell\, T\, t_z\, t_s \searrow w}
\qquad
\frac{t\, \langle\alpha\rangle\, (\lambda x.\, x\, \mathsf{fix}_\ell\, T\, t)\, c \searrow w}{c \mathbin{@} \mathsf{fix}_\ell\, T\, t \searrow w}\ c \in \{\mathsf{zero}\langle\alpha\rangle, \mathsf{suc}\langle\alpha\rangle u\}
$$

For NbE, we add evaluation rules that deal with elimination of delayed reflection:

$$
\frac{A' \searrow \Pi\, A\, B}{(\uparrow^{A'} n) \mathbin{@} u \searrow \uparrow^{B[u]}(n \downarrow^{A} u)}
\qquad
\frac{A \searrow \Pi\, \mathsf{Size}\, B}{(\uparrow^{A} n) \mathbin{@} \alpha \searrow \uparrow^{B[\alpha]}(n\, \alpha)}
\qquad
\frac{A \searrow \forall\, B}{(\uparrow^{A} n) \mathbin{@} \langle\alpha\rangle \searrow \uparrow^{B[\alpha]}(n\langle\alpha\rangle)}
$$

$$
\frac{}{(\uparrow^{A} n) \mathbin{@} \mathsf{case}_\ell\, B\, f_z\, f_s \searrow \uparrow^{B\,(\uparrow^{A} n)}\, n\, \mathsf{case}_\ell\, (\downarrow^{\mathsf{Nat}\,\infty \to \mathsf{Set}_\ell} B)\, (\downarrow^{B\,\mathsf{zero}\langle\infty\rangle} f_z)\, (\downarrow^{(x:\mathsf{Nat}\,\infty) \to B\,(\mathsf{suc}\langle\infty\rangle x)} f_s)}
$$

$$
\frac{}{(\uparrow^{A} n) \mathbin{@} \mathsf{fix}_\ell\, B\, f \searrow n\, \mathsf{fix}_\ell\, (\downarrow^{\mathsf{FixK}} B)\, (\downarrow^{\mathsf{FixT}\, B} f)}
$$

4.2 Read back

The *read back* phase of NbE [Grégoire and Leroy 2002] transforms a reified value d into a normal form v. It is specified via an inductively defined relation $\mathsf{R}_k\, d \searrow v$ and several auxiliary relations. The number k, will be instantiated by the length of the context Γ later. It allows us to transform a de Bruijn level l into a de Bruijn index i, via the law $i + l + 1 = k$. At this point, we do not ensure that the k is large enough to accommodate the de Bruijn levels in d. Levels $l \geq k$ which are to big will simply be mapped to de Bruijn index 0. The correct k is later ensured by our logical relation (Section 5). Even though read back operates on values in practice, formally it is defined on expressions.

$\boxed{\mathsf{R}_k\, d \searrow v}$ Read back reified value d.

$$\frac{U \searrow s \qquad \mathsf{R}^{\mathsf{ty}}_k\, T \searrow V}{\mathsf{R}_k \downarrow^U T \searrow V} \qquad \frac{U \searrow \mathsf{Nat}\,\alpha \qquad \mathsf{R}^{\mathsf{nat}}_k\, u \searrow v}{\mathsf{R}_k \downarrow^U u \searrow v} \qquad \frac{U \searrow N \qquad \mathsf{R}^{\mathsf{up}}_k\, u \searrow m}{\mathsf{R}_k \downarrow^U u \searrow m}$$

$$\frac{U \searrow \Pi\, A\, B \qquad \mathsf{R}_{k+1} \downarrow^{B[\uparrow^A \mathsf{x}_k]} (f \uparrow^A \mathsf{x}_k) \searrow v}{\mathsf{R}_k \downarrow^U f \searrow \lambda v}$$

$$\frac{U \searrow \Pi\, \mathsf{Size}\, B \qquad \mathsf{R}_{k+1} \downarrow^{B[\mathsf{x}_k]} (f\, \mathsf{x}_k) \searrow v}{\mathsf{R}_k \downarrow^U f \searrow \lambda v} \qquad \frac{U \searrow \forall B \qquad \mathsf{R}_{k+1} \downarrow^{B[\mathsf{x}_k]} (f\, \langle \mathsf{x}_k \rangle) \searrow v}{\mathsf{R}_k \downarrow^U f \searrow \lambda v}$$

$\boxed{\mathsf{R}^{\mathsf{up}}_k\, t \searrow m}$ Read back neutrals under annotation. (The annotation is ignored.)

$$\frac{t \searrow \uparrow^T n \qquad \mathsf{R}^{\mathsf{ne}}_k\, n \searrow m}{\mathsf{R}^{\mathsf{up}}_k\, t \searrow m}$$

$\boxed{\mathsf{R}^{\mathsf{nat}}_k\, t \searrow v}$ Read back natural number value.

$$\frac{\mathsf{R}^{\mathsf{up}}_k\, t \searrow m}{\mathsf{R}^{\mathsf{nat}}_k\, t \searrow m} \qquad \frac{t \searrow \mathsf{zero}\langle \alpha \rangle \qquad \mathsf{R}^{\mathsf{size}}_k\, \alpha \searrow a}{\mathsf{R}^{\mathsf{nat}}_k\, t \searrow \mathsf{zero}\langle a \rangle} \qquad \frac{t \searrow \mathsf{suc}\langle \alpha \rangle u \qquad \mathsf{R}^{\mathsf{size}}_k\, \alpha \searrow a \qquad \mathsf{R}^{\mathsf{nat}}_k\, u \searrow v}{\mathsf{R}^{\mathsf{nat}}_k\, t \searrow \mathsf{suc}\langle a \rangle v}$$

$\boxed{\mathsf{R}^{\mathsf{size}}_k\, \alpha \searrow a}$ Read back size value α.

$$\frac{}{\mathsf{R}^{\mathsf{size}}_k\, \infty \searrow \infty} \qquad \frac{}{\mathsf{R}^{\mathsf{size}}_k\, o \searrow o} \qquad \frac{}{\mathsf{R}^{\mathsf{size}}_k\, \mathsf{x}_j + o \searrow \mathsf{v}_{k \dot{-} (1+j)} + o}$$

$\boxed{\mathsf{R}^{\mathsf{ne}}_k\, n \searrow m}$ and $\boxed{\mathsf{R}^{\mathsf{elim}}_k\, e \searrow e^v}$ Read back unreflected neutral.

$$\frac{\mathsf{R}^{\mathsf{elim}}_k\, e_i \searrow e^v_i \text{ for all } i}{\mathsf{R}^{\mathsf{ne}}_k\, \mathsf{x}_j\, \vec{e} \searrow \mathsf{v}_{k \dot{-} (1+j)}\, \vec{e}^v} \qquad \frac{\mathsf{R}_k\, d \searrow v}{\mathsf{R}^{\mathsf{elim}}_k\, d \searrow v} \qquad \frac{\mathsf{R}^{\mathsf{size}}_k\, \alpha \searrow b}{\mathsf{R}^{\mathsf{elim}}_k\, \alpha \searrow b} \qquad \frac{\mathsf{R}^{\mathsf{size}}_k\, \alpha \searrow b}{\mathsf{R}^{\mathsf{elim}}_k\, \langle \alpha \rangle \searrow \langle b \rangle}$$

$$\frac{\mathsf{R}_k\, D \searrow V \qquad \mathsf{R}_k\, d_z \searrow v_z \qquad \mathsf{R}_k\, d_s \searrow v_s}{\mathsf{R}^{\mathsf{elim}}_k\, (\mathsf{case}_\ell\, D\, d_z\, d_s) \searrow \mathsf{case}_\ell\, V\, v_z\, v_s} \qquad \frac{\mathsf{R}_k\, D \searrow V \qquad \mathsf{R}_k\, d \searrow v}{\mathsf{R}^{\mathsf{elim}}_k\, (\mathsf{fix}_\ell\, D\, d) \searrow \mathsf{fix}_\ell\, V\, v}$$

$\boxed{\mathsf{R}^{\mathsf{ty}}_k\, T \searrow V}$ Read back type value.

$$\frac{T \searrow \mathsf{Set}_\ell}{\mathsf{R}^{\mathsf{ty}}_k\, T \searrow \mathsf{Set}_\ell} \qquad \frac{T \searrow \mathsf{Nat}\,\alpha \qquad \mathsf{R}^{\mathsf{size}}_k\, \alpha \searrow b}{\mathsf{R}^{\mathsf{ty}}_k\, T \searrow \mathsf{Nat}\, b} \qquad \frac{\mathsf{R}^{\mathsf{up}}_k\, T \searrow m}{\mathsf{R}^{\mathsf{ty}}_k\, T \searrow m}$$

$$\frac{T \searrow \Pi\, A\, B \qquad \mathsf{R}^{\mathsf{ty}}_k\, A \searrow V_a \qquad \mathsf{R}^{\mathsf{ty}}_{k+1}\, B \searrow V_b}{\mathsf{R}^{\mathsf{ty}}_k\, T \searrow \Pi\, V_a\, V_b} \qquad \frac{T \searrow \Pi^\star \mathsf{Size}\, B \qquad \mathsf{R}^{\mathsf{ty}}_{k+1}\, B \searrow V}{\mathsf{R}^{\mathsf{ty}}_k\, T \searrow \Pi^\star \mathsf{Size}\, V}$$

4.3 Partial equivalence relations

A type T will be interpreted as a partial equivalence relation (PER) $\mathcal{A}$ on terms, i. e., a relation which is symmetric and transitive. The domain $\mathrm{dom}(\mathcal{A})$ of the relation can be thought of as the set of terms which denotes the extension of the type; on $\mathrm{dom}(\mathcal{A}) = \{a \mid \exists a'.\, (a, a') \in \mathcal{A}\}$ the relation $\mathcal{A}$ is in fact an equivalence relation. We write $a = a' \in \mathcal{A}$ for relatedness in $\mathcal{A}$ and $a \in \mathcal{A}$ if $a \in \mathrm{dom}(\mathcal{A})$.

The PERs $\mathcal{N}e$ and $\mathcal{N}f$ characterize (neutral) normalizing values. For instance, two values n and n' are related in $\mathcal{N}e$ if at any $k \in \mathbb{N}$ they can be read back to neutral normal forms m and m' which are identical up to annotations.

$$
\begin{array}{llllllll}
\boxed{n = n' \in \mathcal{N}e} & :\iff & \mathsf{R}^{\mathsf{ne}}_k \; n \searrow m & \text{and} & \mathsf{R}^{\mathsf{ne}}_k \; n' \searrow m' & \text{and} & m \approx m' & \text{for all } k \\
\boxed{d = d' \in \mathcal{N}f} & :\iff & \mathsf{R}_k \; d \searrow v & \text{and} & \mathsf{R}_k \; d' \searrow v' & \text{and} & v \approx v' & \text{for all } k \\
\boxed{e = e' \in \mathcal{E}lim} & :\iff & \mathsf{R}^{\mathsf{elim}}_k \, e \searrow e_v & \text{and} & \mathsf{R}^{\mathsf{elim}}_k \, e' \searrow e'_v & \text{and} & e_v \approx e'_v & \text{for all } k \\
\boxed{A = A' \in \mathcal{T}y} & :\iff & \mathsf{R}^{\mathsf{ty}}_k \; A \searrow V & \text{and} & \mathsf{R}^{\mathsf{ty}}_k \; A' \searrow V' & \text{and} & V \approx V' & \text{for all } k
\end{array}
$$

Once we have established useful closure properties of these PERs, they abstract most of the reasoning about the read-back relation from our proofs. This idea is due to Coquand [Abel et al. 2009].

Lemma 4.1 (Closure properties of $\mathcal{N}e$).

(1) $\mathsf{x}_k = \mathsf{x}_k \in \mathcal{N}e$.
(2) *If* $n = n' \in \mathcal{N}e$ *and* $e = e' \in \mathcal{E}lim$ *then* $n\,e = n'\,e' \in \mathcal{N}e$.

Lemma 4.2 (Closure properties of $\mathcal{E}lim$).

(1) *If* $d = d' \in \mathcal{N}f$ *then* $d = d' \in \mathcal{E}lim$.
(2) *If* $\alpha \in \mathcal{S}ize$ *then* $\alpha = \alpha \in \mathcal{E}lim$.
(3) *If* $\alpha, \alpha' \in \mathcal{S}ize$ *then* $\langle\alpha\rangle = \langle\alpha'\rangle \in \mathcal{E}lim$.
(4) *If* $A = A' \in \mathcal{T}y$ *and* $d_z = d'_z \in \mathcal{N}f$ *and* $d_s = d'_s \in \mathcal{N}f$ *then* $\mathsf{case}_\ell\, A\, d_z\, d_s = \mathsf{case}_\ell\, A'\, d'_z\, d'_s \in \mathcal{E}lim$.
(5) *If* $D = D' \in \mathcal{N}f$ *and* $d = d' \in \mathcal{N}f$ *then* $\mathsf{fix}_\ell\, D\, d = \mathsf{fix}_\ell\, D'\, d' \in \mathcal{E}lim$.

Now we define some PERs and PER constructors on values. All these PERs $\mathcal{A}$ are closed under weak head equality, meaning if $a = b \in \mathcal{A}$ and a' has the same weak head normal form as a, then $a' = b \in \mathcal{A}$. (By symmetry, $\mathcal{A}$ is also closed under weak head equality on the second argument.)

PER $\mathcal{NE}$ interprets all neutral types.

$$\boxed{t = t' \in \mathcal{NE}} \;:\iff\; t \searrow \uparrow^T n \text{ and } t' \searrow \uparrow^{T'} n' \text{ and } n = n' \in \mathcal{N}e.$$

$\boxed{\mathcal{N}at(\alpha)}$ interprets $\mathsf{Nat}\,\alpha$ and is defined inductively by the following rules.

$$
\frac{t = t' \in \mathcal{NE}}{t = t' \in \mathcal{N}at(\beta)}
\qquad
\frac{t \searrow \mathsf{zero}\langle\alpha\rangle \quad t' \searrow \mathsf{zero}\langle\alpha'\rangle}{t = t' \in \mathcal{N}at(\beta + 1)}
\qquad
\frac{t \searrow \mathsf{suc}\langle\alpha\rangle u \quad t' \searrow \mathsf{suc}\langle\alpha'\rangle u' \quad u = u' \in \mathcal{N}at(\beta)}{t = t' \in \mathcal{N}at(\beta + 1)}
$$

$\boxed{\mathcal{S}ize}$ interprets Size and is a discrete PER of size values:

$$
\frac{}{\infty = \infty \in \mathcal{S}ize}
\qquad
\frac{}{o = o \in \mathcal{S}ize}
\qquad
\frac{}{\mathsf{x}_k + o = \mathsf{x}_k + o \in \mathcal{S}ize}
$$

Let $\mathcal{A}$ be a PER (including $\mathcal{A} = \mathcal{S}ize$) and $\mathcal{F}$ a family of PERs over $\mathcal{A}$ such that $\mathcal{F}(u) = \mathcal{F}(u')$ whenever $u = u' \in \mathcal{A}$. We define

$$\boxed{\prod \mathcal{A}\mathcal{F}} \;:\iff\; \{(t, t') \mid t\,u = t'\,u' \in \mathcal{F}(u) \text{ for all } u = u' \in \mathcal{A}\}.$$

For a family $\mathcal{F}$ over $\mathcal{S}ize$ we also have the irrelevant function space

$$\boxed{\forall\,\mathcal{F}} \;:\iff\; \{(t, t') \mid t\langle\alpha\rangle = t'\langle\alpha'\rangle \in \mathcal{F}(\beta) \text{ for all } \alpha, \alpha', \beta \in \mathcal{S}ize\}.$$

$$\frac{T = T' \in \mathcal{NE}}{T = T' \in \mathcal{S}et_\ell} \qquad \mathcal{E}\ell_\ell(T) = \mathcal{NE}$$

$$\frac{T \searrow \mathsf{Nat}\ \alpha \qquad T' \searrow \mathsf{Nat}\ \alpha}{T = T' \in \mathcal{S}et_\ell} \qquad \mathcal{E}\ell_\ell(T) = \mathcal{N}at(\alpha)$$

$$\frac{T \searrow \mathsf{Set}_{\ell'} \qquad T' \searrow \mathsf{Set}_{\ell'}}{T = T' \in \mathcal{S}et_\ell}\ \ell' < \ell \qquad \mathcal{E}\ell_\ell(T) = \mathcal{S}et_{\ell'}$$

$$\frac{\begin{array}{l} T \searrow \Pi\, A\, B \qquad T' \searrow \Pi\, A'\, B' \qquad A = A' \in \mathcal{S}et_\ell \\ B[u] = B'[u'] \in \mathcal{S}et_\ell \text{ for all } u = u' \in \mathcal{E}\ell_\ell(A) \end{array}}{T = T' \in \mathcal{S}et_\ell} \qquad \mathcal{E}\ell_\ell(T) = \prod(\mathcal{E}\ell_\ell(A),\ u \mapsto \mathcal{E}\ell_\ell(B[u]))$$

$$\frac{\begin{array}{l} T \searrow \Pi\, \mathsf{Size}\, B \qquad T' \searrow \Pi\, \mathsf{Size}\, B' \\ B[\alpha] = B'[\alpha] \in \mathcal{S}et_\ell \text{ for all } \alpha \in \mathcal{S}ize \end{array}}{T = T' \in \mathcal{S}et_\ell} \qquad \mathcal{E}\ell_\ell(T) = \prod(\mathcal{S}ize,\ \alpha \mapsto \mathcal{E}\ell_\ell(B[\alpha]))$$

$$\frac{\begin{array}{l} T \searrow \forall B \qquad T' \searrow \forall B' \\ B[\alpha] = B'[\alpha] \in \mathcal{S}et_\ell \text{ for all } \alpha \in \mathcal{S}ize \end{array}}{T = T' \in \mathcal{S}et_\ell} \qquad \mathcal{E}\ell_\ell(T) = \forall(\alpha \mapsto \mathcal{E}\ell_\ell(B[\alpha]))$$

Fig. 7. Semantic types and their interpretation.

4.4 PER model

Semantic types and their interpretation as PERs are now defined via a family of inductive-recursive definitions [Dybjer 2000], one for each universe level ℓ. The construction follows Abel et al. [2007].

By induction on $\ell \in \mathbb{N}$ we define the PER family $\_ = \_ \in \mathsf{Set}_\ell$ of types together with the extension $\mathcal{E}\ell_\ell T$ (for $T = T' \in \mathsf{Set}_\ell$) which is a PER of values of type T. The rules for $\boxed{T = T' \in \mathcal{S}et_\ell}$ are listed in Fig. 7. All relations involved here are closed under weak head equality.

Lemma 4.3 (Well-definedness). *Let* $\mathcal{D} :: T_1 = T_2 \in \mathcal{S}et_\ell$.

(1) *Symmetry:* $T_2 = T_1 \in \mathcal{S}et_\ell$.
(2) *Transitivity: If* $T_2 = T_3 \in \mathcal{S}et_\ell$ *then* $T_1 = T_3 \in \mathcal{S}et_\ell$.
(3) *Extension:* $\mathcal{E}\ell_\ell(T_1) = \mathcal{E}\ell_\ell(T_2)$ *and "both" are PERs.*

Proof. Simultaneously by induction on $\mathcal{D}$. □

Lemma 4.4 (Derivation independence of extension). *If* $\mathcal{D}_1 :: T = T_1 \in \mathcal{S}et_{\ell_1}$ *and* $\mathcal{D}_2 :: T_2 = T \in \mathcal{S}et_{\ell_2}$ *then* $\mathcal{E}\ell_{\ell_1}(T) = \mathcal{E}\ell_{\ell_2}(T)$.

Proof. By induction on $\mathcal{D}_1$ and cases on $\mathcal{D}_2$. □

Since $\mathcal{E}\ell_\ell(T)$ does not depend on ℓ nor the derivation that introduced $T = T' \in \mathcal{S}et_\ell$, we may simply write $t = t' \in \mathcal{E}\ell(T)$ or even $t = t' \in T$.

4.5 Subtyping

The semantic types (PERs) admit subsumption:

Lemma 4.5 (Subsumption).

(1) *If* $\alpha \leq \beta$ *then* $\mathcal{N}at(\alpha) \subseteq \mathcal{N}at(\beta)$.

(2) *If* $\mathcal{F}(\alpha) \subseteq \mathcal{F}'(\alpha)$ *for all* $\alpha \in \mathcal{S}ize$, *then* $\forall\mathcal{F} \subseteq \forall\mathcal{F}'$.
(3) *If* $\mathcal{A}' \subseteq \mathcal{A}$ *and* $\mathcal{F}(u) \subseteq \mathcal{F}'(u)$ *for all* $u \in \mathcal{A}'$, *then* $\prod \mathcal{A}\mathcal{F} \subseteq \prod \mathcal{A}'\,\mathcal{F}'$.
(4) *If* $\ell \leq \ell'$ *then* $\mathcal{S}et_\ell \subseteq \mathcal{S}et_{\ell'}$.

Proof. Propositions (1–3) are clear. For (4), prove $T = T' \in \mathcal{S}et_{\ell'}$ by induction on $T = T' \in \mathcal{S}et_\ell$. For the base types this is direct, let us look at the function space.

$$\frac{T \searrow \Pi\, A\, B \qquad T' \searrow \Pi\, A'\, B' \qquad A = A' \in \mathcal{S}et_\ell \qquad B[u] = B'[u'] \in \mathcal{S}et_\ell \text{ for all } u = u' \in \mathcal{E}\ell_\ell(A)}{T = T' \in \mathcal{S}et_\ell}$$

By induction hypothesis, $A = A' \in \mathcal{S}et_{\ell'}$, and we have $\mathcal{E}\ell_{\ell'}(A) = \mathcal{E}\ell_\ell(A)$ by Lemma 4.4. Assuming $u = u' \in \mathcal{E}\ell(A)$, we get $B[u] = B'[u'] \in \mathcal{S}et_{\ell'}$ by induction hypothesis on $B[u] = B'[u'] \in \mathcal{S}et_\ell$. □

We define subtyping of type values $\boxed{T \leq T' \in \mathcal{T}ype}$ by induction on $T \in \mathcal{S}et_\ell$ and $T' \in \mathcal{S}et_{\ell'}$. Simultaneously, we need to prove correctness, namely that $T \leq T' \in \mathcal{T}ype$ implies $\mathcal{E}\ell(T) \subseteq \mathcal{E}\ell(T')$. The correctness follows from Lemma 4.5 and we do not spell it out here.

$$\frac{T = T' \in \mathcal{NE}}{T \leq T' \in \mathcal{T}ype} \qquad \frac{T \searrow \mathsf{Nat}\,\alpha \quad T' \searrow \mathsf{Nat}\,\alpha' \quad \alpha \leq \alpha'}{T \leq T' \in \mathcal{T}ype} \qquad \frac{T \searrow \mathsf{Set}_{\ell_0} \quad T' \searrow \mathsf{Set}_{\ell_0'} \quad \ell_0 \leq \ell_0'}{T \leq T' \in \mathcal{T}ype}$$

$$\frac{T \searrow \Pi\, A\, B \qquad T' \searrow \Pi\, A'\, B' \qquad A' \leq A \in \mathcal{T}ype \qquad B[u] \leq B'[u'] \in \mathcal{T}ype \text{ for all } u = u' \in A'}{T \leq T' \in \mathcal{T}ype}$$

$$\frac{T \searrow \Pi^\star \mathsf{Size}\, B \qquad T' \searrow \Pi^\star \mathsf{Size}\, B' \qquad B[\alpha] \leq B'[\alpha] \in \mathcal{T}ype \text{ for all } \alpha \in \mathcal{S}ize}{T \leq T' \in \mathcal{T}ype}$$

Lemma 4.6 (Subtyping is a preorder).
(1) *If* $T = T' \in \mathcal{S}et_\ell$ *then* $T \leq T' \in \mathcal{T}ype$.
(2) *If* $T_1 \leq T_2 \in \mathcal{T}ype$ *and* $T_2 \leq T_3 \in \mathcal{T}ype$ *then* $T_1 \leq T_3 \in \mathcal{T}ype$.

4.6 Type shapes

Reflection and reification perform η-expansion so that we arrive at an η-long β-normal form. To perform the η-expansion, the precise type is not needed, just the approximate shape, in particular, whether it is a function type (do expand) or a base type (do not expand). For the logical framework, the shape of a dependent type is just its underlying simple type [Harper and Pfenning 2005]. However, in the presence of universes and large eliminations, there is no underlying simple type. Of course, we can take a type as its own shape, but we want at least that $\mathsf{Nat}\,\alpha$ and $\mathsf{Nat}\,\beta$ have the same shape even for different α, β. Also all neutral types can be summarized under a single shape.

We make our intuition precise by defining a relation $T \lessapprox S$ between type values, to express that S is a possible shape of type T. The asymmetry of this relation stems from the case for function types. At function types $\Pi\, A\, B \lessapprox \Pi\, R\, S$, we take S to be a family over domain A, not R! We cannot take R since we have to compare families B and S at a common domain, and A and R are not equal.

Fig. 8 defines $\boxed{T \lessapprox S}$ for $T \in \mathcal{S}et_\ell$. We call T the *template* and S one of its possible *shapes.* Note that $T \in \mathcal{S}et_\ell$ and $T \lessapprox S$ do not imply $S \in \mathcal{S}et_\ell$. Type shapes are not well-defined types in general. For instance, assume a term $F : \mathsf{Nat}\,0 \to \mathsf{Set}_0$ which diverges if applied to a successor term. Then $T := (x : \mathsf{Nat}\,0) \to F\,x$ is a well-defined type; we have $T \in \mathcal{S}et_0$. Now consider $S := (x : \mathsf{Nat}\,\infty) \to F\,x$. We have $T \lessapprox S$, but S is not well-defined; $S \notin \mathcal{S}et_0$.

Lemma 4.7 (Types are their own shapes). *If* $T = T' \in \mathcal{S}et_\ell$ *then* $T \lessapprox T'$.

Lemma 4.8 (Templates are up to equality). *If* $T = T' \in \mathcal{S}et_\ell$ *and* $T' \lessapprox S$ *then* $T \lessapprox S$.

$$\frac{T \searrow N \quad S \searrow N'}{T \mathrel{\lessapprox} S} \qquad \frac{T \searrow \Pi\, A\, B \quad S \searrow \Pi\, A'\, B' \quad A \mathrel{\lessapprox} A' \quad B[u] \mathrel{\lessapprox} B'[u'] \text{ for all } u = u' \in A}{T \mathrel{\lessapprox} S}$$

$$\frac{T \searrow \mathsf{Set}_\ell \quad S \searrow \mathsf{Set}_\ell}{T \mathrel{\lessapprox} S} \qquad \frac{T \searrow \Pi\, \mathsf{Size}\, B \quad S \searrow \Pi\, \mathsf{Size}\, B' \quad B[\alpha] \mathrel{\lessapprox} B'[\alpha] \text{ for all } \alpha \in \mathcal{S}ize}{T \mathrel{\lessapprox} S}$$

$$\frac{T \searrow \mathsf{Nat}\, \alpha \quad S \searrow \mathsf{Nat}\, \beta}{T \mathrel{\lessapprox} S} \qquad \frac{T \searrow \forall B \quad S \searrow \forall B' \quad B[\alpha] \mathrel{\lessapprox} B'[\alpha'] \text{ for all } \alpha, \alpha' \in \mathcal{S}ize}{T \mathrel{\lessapprox} S}$$

Fig. 8. Type shapes $\boxed{T \mathrel{\lessapprox} S}$.

However, templates are not closed under subtyping in either direction because subtyping is contravariant for function type domains but the shape relation is covariant.

Further, it is not true that equal types make equally good shapes. We do not have that $T \mathrel{\lessapprox} S$ and $S = S' \in \mathcal{S}et_\ell$ imply $T \mathrel{\lessapprox} S'$. This property fails for function types. Given $\Pi\, U\, T \mathrel{\lessapprox} \Pi\, R\, S$ and $\Pi\, R\, S = \Pi\, R'\, S' \in \mathcal{S}et_\ell$ we would need to show that $T[u] \mathrel{\lessapprox} S'[u']$ for all $u = u' \in U$, but we only have $S[u] = S'[u'] \in \mathcal{S}et_\ell$ for all $u = u' \in R$, thus the induction does not go through. The fact that $U \mathrel{\lessapprox} R$ does not give us a handle on their inhabitants, we would need a stronger relation such as $U \le R \in \mathcal{T}ype$. It is possible to construct an actual counterexample, using $\Pi\, R'\, S' = (x : \mathsf{Nat}\, 0) \to F\, x$ from above and $\Pi\, R\, S = (x : \mathsf{Nat}\, 0) \to G\, x$ such that G is defined on all of $\mathsf{Nat}\, \infty$ but agrees with F only on $x \in \mathsf{Nat}\, 0$. Then $\Pi\, U\, T = (x : \mathsf{Nat}\, \infty) \to G\, x$ gives the desired counterexample.

Shapes are used to direct η-expansion when we reflect neutrals into semantic types and reify semantic values to long normal forms. The following theorem is the heart of our technical development.

Theorem 4.9 (Reflection and reification). *Let $T \in \mathcal{S}et_\ell$ and $T \mathrel{\lessapprox} S_1$ and $T \mathrel{\lessapprox} S_2$.*

(1) *If $n_1 = n_2 \in \mathcal{N}e$ then $\uparrow^{S_1} n_1 = \uparrow^{S_2} n_2 \in T$.*
(2) *If $t_1 = t_2 \in T$ then $\downarrow^{S_1} t_1 = \downarrow^{S_2} t_2 \in \mathcal{N}f$.*

Proof. By induction on $T \in \mathcal{S}et_\ell$ and cases on $T \mathrel{\lessapprox} S_1$ and $T \mathrel{\lessapprox} S_2$.

Case $T \searrow \Pi\, A\, B$ with $A \in \mathcal{S}et_\ell$ and $B[u] = B[u'] \in \mathcal{S}et_\ell$ for all $u = u' \in A$

$$\frac{S_1 \searrow \Pi\, A_1\, B_1 \quad A \mathrel{\lessapprox} A_1 \quad B[u] \mathrel{\lessapprox} B_1[u'] \text{ for all } u = u' \in A}{T \mathrel{\lessapprox} S_1}$$

$$\frac{S_2 \searrow \Pi\, A_2\, B_2 \quad A \mathrel{\lessapprox} A_2 \quad B[u] \mathrel{\lessapprox} B_2[u'] \text{ for all } u = u' \in A}{T \mathrel{\lessapprox} S_2}$$

(1) To show $\uparrow^{S_1} n_1 = \uparrow^{S_2} n_2 \in T$ assume arbitrary $u_1 = u_2 \in A$. Let $d_i = \downarrow^{A_i} u_i$. By induction hypothesis (2) with shapes $A \mathrel{\lessapprox} A_1$ and $A \mathrel{\lessapprox} A_2$ we get $d_1 = d_2 \in \mathcal{N}f$. Thus, $n_1\, d_1 = n_2\, d_2 \in \mathcal{N}e$ by Lemma 4.1, and by induction hypothesis (1) with shapes $B[u_1] \mathrel{\lessapprox} B_1[u_1]$ and $B[u_1] \mathrel{\lessapprox} B_2[u_2]$ we obtain $\uparrow^{B_1[u_1]}(n_1\, d_1) = \uparrow^{B_2[u_2]}(n_2\, d_2) \in B[u_1]$. With $(\uparrow^{S_i} n_i)\, u_i \searrow \uparrow^{B_i[u_i]}(n_i\, d_i)$ we are done by definition of $\mathcal{E}\ell(T)$.
(2) We assume $k \in \mathbb{N}$ and show $\mathsf{R}_k \downarrow^{S_i} t_i \searrow \lambda v_i$ for some normal forms $v_1 \approx v_2$. Let $u_i = \uparrow^{A_i} \mathsf{x}_k$. Note that $u_1 = u_2 \in A$ by induction hypothesis, since $\mathsf{x}_k = \mathsf{x}_k \in \mathcal{N}e$ by Lemma 4.1. It is sufficient to show $\mathsf{R}_{k+1} \downarrow^{B_i[u_i]}(t_i\, u_i) \searrow v_i$. By definition of $\mathcal{E}\ell(T)$ we have $t_1\, u_1 = t_2\, u_2 \in B[u_1]$, thus, by induction hypothesis, $\downarrow^{B_1[u_1]}(t_1\, u_1) = \downarrow^{B_2[u_2]}(t_2\, u_2) \in \mathcal{N}f$, which delivers v_1 and v_2 for $k + 1$.

Case $T \searrow \forall B$ with $B[\alpha] \in \mathcal{S}et_\ell$ for all $\alpha \in \mathcal{S}ize$

$$\frac{S_1 \searrow \forall B_1 \qquad B[\alpha] \mathrel{\underline{\approx}} B_1[\alpha'] \text{ for all } \alpha,\alpha' \in \mathcal{S}ize}{T \mathrel{\underline{\approx}} S_1}$$

$$\frac{S_2 \searrow \forall B_2 \qquad B[\alpha] \mathrel{\underline{\approx}} B_2[\alpha'] \text{ for all } \alpha,\alpha' \in \mathcal{S}ize}{T \mathrel{\underline{\approx}} S_2}$$

(1) To show $\uparrow^{S_1} n_1 = \uparrow^{S_2} n_2 \in T$ assume arbitrary $\alpha_1, \alpha_2 \in \mathcal{S}ize$. Since $n_1\langle\alpha_1\rangle = n_2\langle\alpha_2\rangle \in \mathcal{N}e$ by Lemma 4.1, we obtain $\uparrow^{B_1[\alpha_1]}(n_1\langle\alpha_1\rangle) = \uparrow^{B_2[\alpha_2]}(n_2\langle\alpha_2\rangle) \in B[\alpha_1]$ by induction hypothesis. Thus, $(\uparrow^{S_1} n_1)\langle\alpha_1\rangle = (\uparrow^{S_2} n_2)\langle\alpha_2\rangle \in B[\alpha_1]$ by weak head expansion, which entails the goal by definition of $\mathcal{E}\ell(T)$.

(2) Assume $k \in \mathbb{N}$ and note that $\mathsf{x}_k = \mathsf{x}_k \in \mathcal{S}ize$, hence, $t_1\,\langle \mathsf{x}_k\rangle = t_2\,\langle \mathsf{x}_k\rangle \in B[\mathsf{x}_k]$. Thus, by induction hypothesis, $\mathsf{R}_{k+1} \downarrow^{B_i[\mathsf{x}_k]}(t_i\,\langle \mathsf{x}_k\rangle) \searrow v_i$ with $v_1 \approx v_2$, and finally $\mathsf{R}_k \downarrow^{S_i} t_i \searrow \lambda v_i$ by definition of read back. □

Corollary 4.10. *Let* $T \in \mathcal{S}et_\ell$.

(1) *If* $n = n' \in \mathcal{N}e$ *then* $\uparrow^T n = \uparrow^T n' \in T$.

(2) *If* $t = t' \in T$ *then* $\downarrow^T t = \downarrow^T t' \in \mathcal{N}f$.

4.7 Computation with natural numbers

In this section we show that the eliminations for natural numbers are accurately modeled.

Lemma 4.11 (Case). *If* $a = a' \in \mathsf{Nat}\,(\alpha + 1)$ *and* $B = B' \in \mathsf{Nat}\,(\alpha+1) \to \mathsf{Set}_\ell$ *and* $f_z = f_z' \in B\,(\mathsf{zero}\langle\beta\rangle)$ *and* $f_s = f_s' \in (x\!:\!\mathsf{Nat}\,\alpha) \to B\,(\mathsf{suc}\langle\gamma\rangle x)$ *then* $a\,\mathsf{case}_\ell\,B\,f_z\,f_s = a'\,\mathsf{case}_\ell\,B'\,f_z'\,f_s' \in B\,a$.

Proof. By induction on $a = a' \in \mathsf{Nat}\,(\alpha + 1)$.

Case $a \searrow \mathsf{zero}\langle\beta\rangle$ and $a' \searrow \mathsf{zero}\langle\beta'\rangle$. Since our PERs are closed under weak head equality, and, for instance, $a\,\mathsf{case}_\ell\,B\,f_z\,f_s$ has the same weak head normal form as f_z, it suffices to show $f_z = f_z' \in B\,(\mathsf{zero}\langle\beta\rangle)$, which is one of our assumptions.

Case $a \searrow \mathsf{suc}\langle\beta\rangle b$ and $a' \searrow \mathsf{suc}\langle\beta'\rangle b'$ with $b = b' \in \mathsf{Nat}\,\alpha$. Again, it suffices to show $f_s\,b = f_s'\,b' \in B\,(\mathsf{suc}\langle\beta\rangle b)$, which is an instance of our last assumption.

Case $a \searrow \uparrow^T n$ and $a' \searrow \uparrow^{T'} n'$ with $n = n' \in \mathcal{N}e$. Let $D = \downarrow^{\mathsf{Nat}\,\infty \to \mathsf{Set}_\ell} B$ and $d_z = \downarrow^{B\,(\mathsf{zero}\langle\infty\rangle)} f_z$ and $d_s = \downarrow^{(x:\mathsf{Nat}\,\infty)\to B\,(\mathsf{suc}\langle\infty\rangle x)} f_s$ and $e = \mathsf{case}_\ell\,D\,d_z\,d_s$. Let D', d_z', d_s', e' be defined analogously from B', f_z', f_s'. It suffices to show $e = e' \in \mathcal{E}lim$, since then we have $n\,e = n'\,e' \in \mathcal{N}e$ by the closure properties of $\mathcal{N}e$ (Lemma 4.1), and $\uparrow^{B[a]}(n\,e) = \uparrow^{B'[a']}(n'\,e') \in B[a]$ by reflection (Theorem 4.9). The remaining goal $\mathsf{case}_\ell\,D\,d_z\,d_s = \mathsf{case}_\ell\,D'\,d_z'\,d_s' \in \mathcal{E}lim$ follows by the closure properties for eliminations (Lemma 4.2), since $D = D' \in \mathcal{N}f$ and $d_z = d_z' \in \mathcal{N}f$ and $d_s = d_s' \in \mathcal{N}f$ all hold by reification (Theorem 4.9). □

Lemma 4.12 ($\mathcal{N}at$ is cocontinuous). $\mathcal{N}at(\infty) = \bigcup_{\alpha<\infty} \mathcal{N}at(\alpha)$.

Proof. By induction on $a = a' \in \mathcal{N}at(\infty)$, we can easily show $a = a' \in \mathcal{N}at(\alpha)$ for some $\alpha < \infty$. For instance, α could be the number of uses of the successor rule plus one. □

As the semantic counterpart of judgement $\Gamma \vdash T : \mathsf{Adm}\,\ell$, let us write $\boxed{B = B' \in \mathsf{Adm}\,\ell}$ iff $B = B' \in \mathsf{FixK}\,\ell$ and for all $\beta \in \mathcal{S}ize$ and $a \in \mathsf{Nat}\,\beta$ we have $B\,\beta\,a \leq B\,\infty\,a \in \mathcal{T}ype$ and $B'\,\beta\,a \leq B'\,\infty\,a \in \mathcal{T}ype$.

Lemma 4.13 (Fix). *Let* $g = a\,\mathsf{fix}_\ell\,B\,f$ *and* $g' = a'\,\mathsf{fix}_\ell\,B'\,f'$. *If* $a = a' \in \mathsf{Nat}\,\alpha$ *and* $B = B' \in \mathsf{Adm}\,\ell$ *and* $f = f' \in \mathsf{FixT}\,B$ *then* $g = g' \in B\,\alpha\,a$.

Proof. By well-founded induction on α.

Case $\alpha < \infty$ and $a \searrow \uparrow^T n$ and $a' \searrow \uparrow^{T'} n'$ and $n = n' \in \mathcal{N}e$. In this case g and g' evaluate to neutral applications of fix. The proof proceeds analogously to Lemma 4.11.

Case $\alpha < \infty$ and $a \searrow c$ and $a' \searrow c'$. Then $\alpha = \beta + 1$ with $\beta < \infty$. The weak head normal form of g equals the weak head normal form of $h := f\,\langle\gamma\rangle\,(\lambda x.\, x\,\mathsf{fix}_\ell\, B\, f)\, c$ where γ is the size annotation of c. It suffices to show $h = h' \in B\,\alpha\, a$ for h' defined analogously from B', f', c'. This follows from the assumption on f, f' if we manage to show $(\lambda x.\, x\,\mathsf{fix}_\ell\, B\, f) = (\lambda x.\, x\,\mathsf{fix}_\ell\, B'\, f') \in (x : \mathsf{Nat}\,\beta) \to B\,\beta\, x$. To this end, assume $b = b' \in \mathsf{Nat}\,\beta$ and show $b\,\mathsf{fix}_\ell\, B\, f = b'\,\mathsf{fix}_\ell\, B'\, f' \in B\,\beta\, b$. However, this is an instance of the induction hypothesis thanks to $\beta < \alpha$.

Case $\alpha = \infty$. Note that $a = a' \in \mathsf{Nat}\,\alpha$ for some $\alpha < \infty$ by Lemma 4.12. By induction hypothesis, $g = g' \in B\,\alpha\, a$. Since $B\,\alpha\, a \leq B\,\infty\, a$ by admissibility of B, the goal $g = g' \in B\,\infty\, a$ follows by subsumption. □

4.8 Fundamental Theorem

In this section we show that the declarative judgements are sound, in particular, well-formed syntactic types map to semantic types, and definitionally equal terms map to related values in the PER model. The proof runs the usual course. First, we define inductively a PER of substitutions $\boxed{\eta = \eta' \mathrel{\dot{=}} \rho \in \Gamma}$.

$$\frac{}{() = () \mathrel{\dot{=}} () \in ()} \qquad \frac{\eta = \eta' \mathrel{\dot{=}} \rho \in \Gamma \qquad T\rho \in \mathcal{S}et_\ell \qquad u = u' = t \in T\rho}{(\eta, u) = (\eta', u') \mathrel{\dot{=}} (\rho, t) \in \Gamma.T}$$

$$\frac{\eta = \eta' \mathrel{\dot{=}} \rho \in \Gamma \qquad \alpha \in \mathcal{S}ize}{(\eta, \alpha) = (\eta', \alpha) \mathrel{\dot{=}} (\rho, \alpha) \in \Gamma.\mathsf{Size}} \qquad \frac{\eta = \eta' \mathrel{\dot{=}} \rho \in \Gamma \qquad \alpha, \alpha', \beta \in \mathcal{S}ize}{(\eta, \alpha) = (\eta', \alpha') \mathrel{\dot{=}} (\rho, \beta) \in \Gamma.^{\div}\mathsf{Size}}$$

We write $\rho \in \Gamma$ for $\rho = \rho \mathrel{\dot{=}} \rho \in \Gamma$.

Lemma 4.14 (Resurrection). *If $\eta = \eta' \mathrel{\dot{=}} \rho \in \Gamma$ then $\rho \in \Gamma^\oplus$.*

Then, in Fig. 9, we define semantic counterparts of our declarative judgements by recursion on the length of the context.

Theorem 4.15 (Fundamental theorem).

(1) *If $\vdash \Gamma$ then $\models \Gamma$.*
(2) *If $\Gamma \vdash J$ then $\Gamma \models J$.*

Proof. Simultaneously, by induction on the derivation.

Case $\forall$-introduction.

$$\frac{\Gamma.^{\div}\mathsf{Size} \vdash t = t' : T}{\Gamma \vdash \lambda t = \lambda t' : \forall T}$$

First $\models \Gamma$ follows from the induction hypothesis $\models \Gamma.^{\div}\mathsf{Size}$. To show $\Gamma^\oplus \models \forall T$, assume $\eta = \eta' \mathrel{\dot{=}} \rho \in \Gamma^\oplus$ and show $(\forall T)\eta = (\forall T)\eta' \in \mathcal{S}et_\ell$ for some ℓ. To this end, assume $\alpha \in \mathcal{S}ize$ and show $T(\eta, \alpha) = T(\eta', \alpha) \in \mathcal{S}et_\ell$. Note that $(\eta, \alpha) = (\eta', \alpha) \mathrel{\dot{=}} (\rho, \alpha) \in \Gamma.^{\div}\mathsf{Size}^\oplus = \Gamma^\oplus.\mathsf{Size}$, thus, we can instantiate the induction hypothesis and obtain our goal.

For the main goal, assume $\eta = \eta' \mathrel{\dot{=}} \rho \in \Gamma$ and show $(\lambda t)\eta = (\lambda t')\eta' \in (\forall T)\rho$. To this end, assume arbitrary $\alpha, \alpha', \beta \in \mathcal{S}ize$ and show $t(\eta, \alpha) = t'(\eta', \alpha') \in T(\rho, \beta)$. Since $(\eta, \alpha) = (\eta', \alpha') \mathrel{\dot{=}} (\rho, \beta) \in \Gamma.^{\div}\mathsf{Size}$, we conclude by induction hypothesis.

Case $\forall$-elimination.

$$\frac{\Gamma \vdash t = t' : \forall T \qquad \Gamma \vdash a, a' : \mathsf{Size} \qquad \Gamma^\oplus \vdash b : \mathsf{Size}}{\Gamma \vdash t\langle a\rangle = t'\langle a'\rangle : T[b]}$$

$\models ()$	$:\Longleftrightarrow$	true
$\models \Gamma.{}^\star\mathsf{Size}$	$:\Longleftrightarrow$	$\models \Gamma$
$\models \Gamma.s$	$:\Longleftrightarrow$	$\models \Gamma$
$\models \Gamma.T$	$:\Longleftrightarrow$	$\models \Gamma$ and $\Gamma^{\oplus} \models T$
$\Gamma \models s$	$:\Longleftrightarrow$	$\models \Gamma$
$\Gamma \models T$	$:\Longleftrightarrow$	$\Gamma \models T = T$
$\Gamma \models T = T'$	$:\Longleftrightarrow$	$\Gamma \models T = T' : s$ for some s
$\Gamma \models T : \mathsf{Adm}\ \ell$	$:\Longleftrightarrow$	$\Gamma \models T : \mathsf{FixK}\,\ell$ and $T\eta = T'\eta' \in \mathsf{Adm}\ \ell$ for all $\eta = \eta' \doteqdot \rho \in \Gamma$
$\Gamma \models T \leq T'$	$:\Longleftrightarrow$	$\Gamma \models T$ and $\Gamma \models T'$ and $T\eta \leq T'\eta' \in \mathcal{T}ype$ for all $\eta = \eta' \doteqdot \rho \in \Gamma$
$\Gamma \models t : T$	$:\Longleftrightarrow$	$\Gamma \models t = t : T$
$\Gamma \models t = t' : T$	$:\Longleftrightarrow$	$\models \Gamma.T$ and $t\eta = t'\eta' \in T\rho$ for all $\eta = \eta' \doteqdot \rho \in \Gamma$
$\Gamma \models \sigma : \Delta$	$:\Longleftrightarrow$	$\Gamma \models \sigma = \sigma \doteqdot \sigma : \Delta$
$\Gamma \models \sigma = \sigma' \doteqdot \tau : \Delta$	$:\Longleftrightarrow$	$\models \Gamma$ and $\models \Delta$ and $\sigma\eta = \sigma'\eta' \doteqdot \tau\rho \in \Delta$ for all $\eta = \eta' \doteqdot \rho \in \Gamma$

Fig. 9. Semantic judgements.

First, $\models \Gamma$ follows by induction hypothesis. For goal $\Gamma^{\oplus} \models T[b]$, assume $\eta = \eta' \doteqdot \rho \in \Gamma^{\oplus}$ and show $T[b]\eta = T[b]\eta' \in \mathcal{S}et_\ell$ for some ℓ. By induction hypothesis, $b\eta = b\eta' \in \mathcal{S}ize$. By another induction hypothesis, $(\forall T)\eta = (\forall T)\eta' \in \mathcal{S}et_\ell$, which by definition entails our goal $T(\eta, b\eta) = T(\eta', b\eta') \in \mathcal{S}et_\ell$.

For the remaining main goal, assume $\eta = \eta' \doteqdot \rho \in \Gamma$ and show $t\langle a\rangle\eta = t'\langle a'\rangle\eta' \in T[b]\rho$. By definition of substitution, it suffices to show $t\eta\langle a\eta\rangle = t'\eta'\langle a'\eta'\rangle \in T(\rho, b\rho)$. By induction hypothesis, $t\eta = t'\eta' : (\forall T)\rho$, thus, by definition of this PER, $t\eta\langle \alpha_1\rangle = t'\eta'\langle \alpha_2\rangle \in T(\rho, b')$ for any size values α_1, α_2, and b'. We conclude by choosing $\alpha_1 = a\eta$ and $\alpha_2 = a'\eta'$ and $b' = b\rho$. We now argue that this choice is possible, namely that $a\eta, a'\eta, b\rho \in \mathcal{S}ize$. Note that the induction hypothesis gives us $a\eta = a\eta' \in \mathcal{S}ize$ and $a'\eta = a'\eta' \in \mathcal{S}ize$. By resurrection (Lemma 4.14) we have $\rho \in \Gamma^{\oplus}$, thus, by induction hypothesis, $b\rho \in \mathcal{S}ize$. □

4.9 Completeness of NbE

From the fundamental theorem, we harvest completeness of NbE in this section, i. e., we show that definitionally equal terms have the same normal form. We may write simply Γ for its length $|\Gamma|$ when there is no danger of confusion, for instance in de Bruijn level x_Γ or in read back R_Γ. We define the *identity environment* $\boxed{\rho_\Gamma}$ by induction on Γ, setting $\rho_{()} = ()$ and $\rho_{\Gamma.{}^\star\mathsf{Size}} = (\rho_\Gamma, \mathsf{x}_\Gamma)$ and $\rho_{\Gamma.T} = (\rho_\Gamma, \uparrow^{T\rho_\Gamma}\mathsf{x}_\Gamma)$.

Lemma 4.16 (Identity environment). *If* $\vdash \Gamma$ *then* $\rho_\Gamma \in \Gamma$.

Proof. By induction on $\vdash \Gamma$.

Case $$\frac{\vdash \Gamma \qquad \Gamma^{\oplus} \vdash T}{\vdash \Gamma.T}$$

By induction hypothesis $\rho_\Gamma \in \Gamma$. By resurrection (Lemma 4.14) $\rho_\Gamma \in \Gamma^{\oplus}$. By the fundamental theorem (Thm. 4.15) we have $A := T\rho_\Gamma \in \mathcal{S}et_\ell$ for some ℓ. By reflection (Cor. 4.10) it follows that $\uparrow^A\mathsf{x}_\Gamma \in A$, thus $(\rho_\Gamma, \uparrow^A\mathsf{x}_\Gamma) \in \Gamma.T$.

Case $$\frac{\vdash \Gamma}{\vdash \Gamma.{}^\star\mathsf{Size}}$$

By induction hypothesis $\rho_\Gamma \in \Gamma$. Since $\mathsf{x}_\Gamma \in \mathcal{S}ize$ we conclude $(\rho_\Gamma, \mathsf{x}_\Gamma) \in \Gamma.^\star\mathrm{Size}$. □

We now define the normalization relation $\boxed{\mathsf{nbe}_\Gamma^T t \searrow v}$ $:\iff$ $\mathsf{R}_\Gamma \downarrow^{T\rho_\Gamma}(t\rho_\Gamma) \searrow v$. Whenever $\mathsf{nbe}_\Gamma^T t \searrow v$, we may write $\mathsf{nbe}_\Gamma^T t$ for v.

Theorem 4.17 (Completeness of NbE). *If $\Gamma \vdash t = t' : T$ then there are normal forms $v \approx v'$ such that $\mathsf{nbe}_\Gamma^T t \searrow v$ and $\mathsf{nbe}_\Gamma^T t' \searrow v'$.*

Proof. By the fundamental theorem, $T\rho_\Gamma \in \mathcal{S}et_\ell$ for some ℓ and $t\rho_\Gamma = t'\rho_\Gamma \in T\rho_\Gamma$. By reification (Cor. 4.10) we have $\downarrow^{T\rho_\Gamma}(t\rho_\Gamma) = \downarrow^{T\rho_\Gamma}(t'\rho_\Gamma) \in \mathcal{N}f$ which implies the theorem by read back with $k = |\Gamma|$. □

5 SOUNDNESS OF NORMALIZATION BY EVALUATION

In this section, we show that NbE is sound for judgmental equality, i.e., that *same normal form* implies *definitional equality*. The proof follows Abel et al. [2007] and Fridlender and Pagano [2013] and defines a Kripke logical relation $\Gamma \vdash t : T \circledR f \in A$ between a well-typed term $\Gamma \vdash t : T$ and a value $f \in A$. However, in contrast to the cited works, the logical relation defined in the following will also yield a weak head normalization theorem.

First, let us define some auxiliary judgements that relate a well-formed syntactic object to a value, via read back. They will constitute the logical relation for base types, but need to be strengthened for function types.

$$\begin{array}{lcl}
\Gamma \vdash a \doteq \mathsf{R}^{\mathsf{size}}\,\alpha & :\iff & \forall \xi : \Gamma' \le \Gamma.\ \mathsf{R}^{\mathsf{size}}_{\Gamma'}\,\alpha \searrow a\xi \\
\Gamma \vdash T \doteq \mathsf{R}^{\mathsf{ty}}\,A : s & :\iff & \forall \xi : \Gamma' \le \Gamma.\ \exists V.\ \mathsf{R}^{\mathsf{ty}}_{\Gamma'}\,A \searrow V \ \text{ and } \Gamma' \vdash T\xi = V : s \\
\Gamma \vdash t \doteq \mathsf{R}\,d : T & :\iff & \forall \xi : \Gamma' \le \Gamma.\ \exists v.\ \mathsf{R}_{\Gamma'}\,d \searrow v \ \text{ and } \Gamma' \vdash t\xi = v : T\xi \\
\Gamma \vdash t \doteq \mathsf{R}^{\mathsf{ne}}\,n : T & :\iff & \forall \xi : \Gamma' \le \Gamma.\ \exists m.\ \mathsf{R}^{\mathsf{ne}}_{\Gamma'}\,n \searrow m \ \text{ and } \Gamma' \vdash t\xi = m : T\xi
\end{array}$$

By definition, these relations are closed under subsumption and weakening, e.g., if $\Gamma \vdash t \doteq \mathsf{R}\,d : T$ and $\Gamma \vdash T \le T'$ then $\Gamma \vdash t \doteq \mathsf{R}\,d : T'$, and if $\xi : \Gamma' \le \Gamma$ then $\Gamma' \vdash t\xi \doteq \mathsf{R}\,d : T\xi$.

Lemma 5.1 (Fresh variable readback). *If $\Gamma \vdash U$ then $\Gamma.U \vdash \mathsf{v}_0 \doteq \mathsf{R}^{\mathsf{ne}}\,\mathsf{x}_\Gamma : U\!\uparrow$.*

Proof. Assume $\xi : \Gamma' \le \Gamma.U$. Note that $\mathsf{R}^{\mathsf{ne}}_{\Gamma'}\,\mathsf{x}_\Gamma \searrow \mathsf{v}_i$ where $i = |\Gamma'| - |\Gamma.U|$ is the length of the context extension. Since $\mathsf{v}_0\xi = \mathsf{v}_i$, we conclude $\Gamma' \vdash \mathsf{v}_0\xi = \mathsf{v}_i : U\!\uparrow\!\xi$ by weakening of the judgement $\Gamma.U \vdash \mathsf{v}_0 = \mathsf{v}_0 : U\!\uparrow$. □

Lemma 5.2 (Closure properties for neutrals).

(1) *If $\Gamma \vdash t \doteq \mathsf{R}^{\mathsf{ne}}\,n : \Pi\,U\,T$ and $\Gamma \vdash u \doteq \mathsf{R}\,d : U$ then $\Gamma \vdash t\,u \doteq \mathsf{R}^{\mathsf{ne}}\,n\,d : T[u]$.*
(2) *If $\Gamma \vdash t \doteq \mathsf{R}^{\mathsf{ne}}\,n : \Pi\,\mathrm{Size}\,T$ and $\Gamma \vdash a \doteq \mathsf{R}^{\mathsf{size}}\alpha$ then $\Gamma \vdash t\,a \doteq \mathsf{R}^{\mathsf{ne}}\,n\,\alpha : T[a]$.*
(3) *If $\Gamma \vdash t \doteq \mathsf{R}^{\mathsf{ne}}\,n : \forall\,T$ and $\Gamma^\oplus \vdash a, b : \mathrm{Size}$ and $\alpha \in \mathcal{S}ize$ then $\Gamma \vdash t\,\langle a\rangle \doteq \mathsf{R}^{\mathsf{ne}}\,n\,\langle\alpha\rangle : T[b]$.*

Proof. For (3), assume $\xi : \Gamma' \le \Gamma$. We have to show that there exists a neutral normal form m' such that $\mathsf{R}^{\mathsf{ne}}_{\Gamma'}\,n\,\langle\alpha\rangle \searrow m'$ and $\Gamma' \vdash (t\,\langle a\rangle)\xi = m' : T[b]\xi$. Sizes can always be read back, thus, let $\mathsf{R}^{\mathsf{size}}_{\Gamma'}\,\alpha \searrow a_0$, which guarantees $\Gamma'^\oplus \vdash a_0 : \mathrm{Size}$. By assumption $\Gamma \vdash t \doteq \mathsf{R}^{\mathsf{ne}}\,n : \forall\,T$, there is an m with $\mathsf{R}^{\mathsf{ne}}_{\Gamma'}\,n \searrow m$ and $\Gamma' \vdash t\xi = m : (\forall\,T)\xi$. The latter equation implies $\Gamma' \vdash t\xi\langle a\xi\rangle = m\langle a_0\rangle : T(\xi, b\xi)$ by irrelevant size application to $\Gamma'^\oplus \vdash a\xi, a_0, b\xi : \mathrm{Size}$, thus we are done with $m' := m\langle a_0\rangle$. □

Let $\boxed{\Gamma \vdash T \searrow W : s}$ denote the conjunction of $T \searrow W$ and $\Gamma \vdash T = W : s$. We simultaneously define $\boxed{\Gamma \vdash T' \circledR A' \in s}$ for $\Gamma \vdash T' : s$ and $\boxed{\Gamma \vdash t : T' \circledR f \in A'}$ for $\Gamma \vdash t : T'$ and $f \in A'$ by induction on $A' \in s$.

Case $A' \searrow N$ neutral.

$$\begin{array}{lcl} \Gamma \vdash T' \circledR A' \in s & :\Longleftrightarrow & \Gamma \vdash T' \searrow n : s \text{ for some neutral } n \text{ and } \Gamma \vdash T' \doteq \mathsf{R}^{\mathsf{ty}}\, A' : s. \\ \Gamma \vdash t : T' \circledR f \in A' & :\Longleftrightarrow & \Gamma \vdash t \doteq \mathsf{R}{\downarrow}^{A'} f : T'. \end{array}$$

Case $A' \searrow \mathsf{Nat}\,\alpha$.

$$\begin{array}{lcl} \Gamma \vdash T' \circledR A' \in s & :\Longleftrightarrow & \Gamma \vdash T' \searrow \mathsf{Nat}\, a : s \text{ for some } a \text{ and } \Gamma \vdash a \doteq \mathsf{R}^{\mathsf{size}}\alpha. \\ \Gamma \vdash t : T' \circledR f \in A' & :\Longleftrightarrow & \Gamma^{\oplus} \vdash T' \searrow \mathsf{Nat}\, a : s \text{ for some } a \text{ and } \Gamma^{\oplus} \vdash a \doteq \mathsf{R}^{\mathsf{size}}\alpha \end{array}$$
$$\quad \text{and } \Gamma \vdash t \doteq \mathsf{R}{\downarrow}^{A'} f : \mathsf{Nat}\, a.$$

Case $A' \searrow \mathsf{Set}_{\ell'}$.

$$\begin{array}{lcl} \Gamma \vdash T' \circledR A' \in s & :\Longleftrightarrow & \Gamma \vdash T' \searrow \mathsf{Set}_{\ell'} : s. \\ \Gamma \vdash U : T' \circledR B \in A' & :\Longleftrightarrow & \Gamma^{\oplus} \vdash T' \searrow \mathsf{Set}_{\ell'} : s \text{ and } \Gamma \vdash U \circledR B \in \mathsf{Set}_{\ell'}. \end{array}$$

Case $A' \searrow \Pi\, A\, B$.

$$\begin{array}{lcl} \Gamma \vdash T' \circledR A' \in s & :\Longleftrightarrow & \Gamma \vdash T' \searrow \Pi\, U\, T : s \text{ for some } U, T \text{ and } \Gamma \vdash U \circledR A \in s \end{array}$$
$$\quad \text{and } \forall \xi : \Gamma' \leq \Gamma.\ \Gamma' \vdash u : U\xi \circledR a \in A \implies \Gamma' \vdash T(\xi, u) \circledR B[a] \in s.$$

$$\begin{array}{lcl} \Gamma \vdash t : T' \circledR f \in A' & :\Longleftrightarrow & \Gamma^{\oplus} \vdash T' \searrow \Pi\, U\, T : s \text{ for some } U, T \text{ and } \Gamma^{\oplus} \vdash U \circledR A \in s \end{array}$$
$$\quad \text{and } \forall \xi : \Gamma' \leq \Gamma.\ \Gamma' \vdash u : U\xi \circledR a \in A \implies \Gamma' \vdash t\xi\, u : T(\xi, u) \circledR f\, a \in B[a].$$

Case $A' \searrow \Pi\, \mathsf{Size}\, B$.

$$\begin{array}{lcl} \Gamma \vdash T' \circledR A' \in s & :\Longleftrightarrow & \Gamma \vdash T' \searrow \Pi\, \mathsf{Size}\, T : s \text{ for some } T \end{array}$$
$$\quad \text{and } \forall \xi : \Gamma' \leq \Gamma.\ \Gamma' \vdash a \doteq \mathsf{R}^{\mathsf{size}}\alpha \implies \Gamma' \vdash T(\xi, a) \circledR B[\alpha] \in s.$$

$$\begin{array}{lcl} \Gamma \vdash t : T' \circledR f \in A' & :\Longleftrightarrow & \Gamma^{\oplus} \vdash T' \searrow \Pi\, \mathsf{Size}\, T : s \text{ for some } T \end{array}$$
$$\quad \text{and } \forall \xi : \Gamma' \leq \Gamma.\ \Gamma' \vdash a \doteq \mathsf{R}^{\mathsf{size}}\alpha \implies \Gamma' \vdash t\xi\, a : T(\xi, a) \circledR f\, \alpha \in B[\alpha].$$

Case $A' \searrow \forall B$.

$$\begin{array}{lcl} \Gamma \vdash T' \circledR A' \in s & :\Longleftrightarrow & \Gamma \vdash T' \searrow \forall T : s \text{ for some } T \end{array}$$
$$\quad \text{and } \forall \xi : \Gamma' \leq \Gamma,\ \Gamma' \vdash b : \mathsf{Size},\ \beta \in \mathcal{S}ize.\ \Gamma' \vdash b \doteq \mathsf{R}^{\mathsf{size}}\beta \implies \Gamma' \vdash T(\xi, b) \circledR B[\beta] \in s.$$

$$\begin{array}{lcl} \Gamma \vdash t : T' \circledR f \in A' & :\Longleftrightarrow & \Gamma^{\oplus} \vdash T' \searrow \forall T : s \text{ for some } T \end{array}$$
$$\quad \text{and } \forall \xi : \Gamma' \leq \Gamma,\ \Gamma'^{\oplus} \vdash a, b : \mathsf{Size},\ \alpha, \beta \in \mathcal{S}ize.$$
$$\qquad \Gamma'^{\oplus} \vdash b \doteq \mathsf{R}^{\mathsf{size}}\beta \implies \Gamma' \vdash t\xi\, \langle a \rangle : T(\xi, b) \circledR f\, \langle \alpha \rangle \in B[\beta].$$

We may prove theorems “by induction on $\Gamma \vdash T \circledR A \in s$”, even if in reality this will be proofs by induction on $A \in s$ and cases on $\Gamma \vdash T \circledR A \in s$. We write $\boxed{\Gamma \vdash T \circledR A}$ if $\Gamma \vdash T \circledR A \in s$ for some sort s.

Lemma 5.3 (Weakening). *Let* $\xi : \Gamma' \leq \Gamma$.

(1) *If* $\Gamma \vdash T \circledR A \in s$ *then* $\Gamma' \vdash T\xi \circledR A \in s$.
(2) *If* $\Gamma \vdash t : T \circledR f \in A$ *then* $\Gamma' \vdash t\xi : T\xi \circledR f \in A$.

Theorem 5.4 (Into and out of the logical relation). *Let* $\Gamma \vdash T \circledR A \in s$ *and* $A \mathrel{\underline{\in}} S$. *Then:*

(1) *If* $\Gamma \vdash t \doteq \mathsf{R}^{\mathsf{ne}}\, n : T$ *then* $\Gamma \vdash t : T \circledR {\uparrow}^S n \in A$.
(2) *If* $\Gamma \vdash t : T \circledR f \in A$ *then* $\Gamma \vdash t \doteq \mathsf{R}{\downarrow}^S f : T$.
(3) $\Gamma \vdash T \doteq \mathsf{R}^{\mathsf{ty}}\, A : s$.

Proof. We prove the propositions for A', S', and T' (instead of A, S, and T) simultaneously by induction on $A' \in s$.
Case $A' \searrow \Pi\, A\, B$ and $S' \searrow \Pi\, R\, S$ and $\Gamma^{\oplus} \vdash T' = \Pi\, U\, T$.

(1) The premise is, after type conversion, $\Gamma \vdash t \doteq \mathsf{R}^{\mathsf{ne}}\, n : \Pi\, U\, T$. To demonstrate $\Gamma \vdash t : T' \circledR \uparrow^{S'} n \in A'$ we assume $\xi : \Gamma' \leq \Gamma$ and $\Gamma' \vdash u : U\xi \circledR a \in A$ and show $\Gamma' \vdash (t\xi\, u) : T(\xi, u) \circledR \uparrow^{S[a]}(n \downarrow^{A} a) \in B[a]$. By induction hypothesis (2) we have $\Gamma' \vdash u \doteq \mathsf{R}\downarrow^{R} a : U\xi$, and together with the weakened premise $\Gamma' \vdash t\xi \doteq \mathsf{R}^{\mathsf{ne}}\, n : (\Pi\, U\, T)\xi$ we get $\Gamma' \vdash t\xi\, u \doteq \mathsf{R}^{\mathsf{ne}}\, (n \downarrow^{R} a) : T(\xi, u)$. The goal follows by induction hypothesis (1) for $\Gamma' \vdash T(\xi, u) \circledR B[a] \in s$.

(2) The premise is $\Gamma \vdash t : T' \circledR f \in A'$, which means that for all $\xi : \Gamma' \leq \Gamma$ and $\Gamma' \vdash u : U\xi \circledR a \in A$ we have $\Gamma' \vdash t\xi\, u : T(\xi, u) \circledR f\, a \in B[a]$.
To show $\Gamma \vdash t \doteq \mathsf{R}\downarrow^{S'} f : T'$ we assume $\xi : \Gamma' \leq \Gamma$ and produce a normal form v such that $\mathsf{R}_{\Gamma'} \downarrow^{S'} f \searrow \lambda v$ and $\Gamma' \vdash t\xi = \lambda v : T'\xi$. Induction hypothesis (1) on $\Gamma'.U\xi \vdash \mathsf{v}_0 \doteq \mathsf{R}^{\mathsf{ne}}\, \mathsf{x}_{\Gamma'} : U\xi\uparrow$ gives us $\Gamma'.U\xi \vdash \mathsf{v}_0 : U\xi\uparrow \circledR a \in A$ with $a := \uparrow^{R} \mathsf{x}_{\Gamma'}$. Thus, we can instantiate the assumption $\Gamma \vdash t : T' \circledR f \in A'$ to obtain $\Gamma'.U\xi \vdash (t\xi\uparrow)\, \mathsf{v}_0 : T(\xi\uparrow, \mathsf{v}_0) \circledR f\, a \in B[a]$. Now induction hypothesis (2) yields a normal form v such that $\mathsf{R}_{\Gamma'.U\xi} \downarrow^{S[a]}(f\, a) \searrow v$ and $\Gamma'.U\xi \vdash (t\xi\uparrow)\, \mathsf{v}_0 = v : T(\xi\uparrow, \mathsf{v}_0)$. Since a is the semantic version of the last bound variable, $\mathsf{R}_{\Gamma'} \downarrow^{S'} f \searrow \lambda v$ follows by definition of reification. For the final goal, we λ-abstract the definitional equality to $\Gamma' \vdash \lambda.\, (t\xi\uparrow)\, \mathsf{v}_0 = \lambda v : \Pi\, (U\xi)\, (T(\xi\uparrow, \mathsf{v}_0))$ and with η-equality and the substitution laws we get $\Gamma' \vdash t\xi = \lambda v : (\Pi\, U\, T)\xi$.

Case $A' \searrow \forall A$ and $S' \searrow \forall S$ and $\Gamma^{\oplus} \vdash T' = \forall T$.

(1) We assume $\xi : \Gamma' \leq \Gamma$ and $\Gamma'^{\oplus} \vdash a, b : \mathrm{Size}$ and $\alpha, \beta \in \mathcal{S}ize$ with $\Gamma'^{\oplus} \vdash b \doteq \mathsf{R}^{\mathsf{size}}\beta$ and show $\Gamma' \vdash t\xi\langle a\rangle : T(\xi, b) \circledR (\uparrow^{S'} n)\langle\alpha\rangle \in A[\beta]$. By the evaluation rules for reflection, it is sufficient to show $\Gamma' \vdash t\xi\langle a\rangle : T(\xi, b) \circledR \uparrow^{S[\alpha]}(n\langle\alpha\rangle) \in A[\beta]$. This follows by induction hypothesis (1) if $\Gamma' \vdash T(\xi, b) \circledR A[\beta] \in s$ and $A[\beta] \leqq S[\alpha]$ and $\Gamma' \vdash t\xi\langle a\rangle \doteq \mathsf{R}^{\mathsf{ne}}\, n\langle\alpha\rangle : T(\xi, b)$. The first two of these subgoals are immediate. The third follows by Lemma 5.2 (3) from the weakened assumption $\Gamma' \vdash t\xi \doteq \mathsf{R}^{\mathsf{ne}}\, n : T'\xi$.

(2) To show $\Gamma \vdash t \doteq \mathsf{R}\downarrow^{S'} f : T'$ we assume $\xi : \Gamma' \leq \Gamma$ and produce a normal form v such that $\mathsf{R}_{\Gamma'} \downarrow^{S'} f \searrow \lambda v$ and $\Gamma' \vdash t\xi = \lambda v : T'\xi$. It is sufficient to show $\mathsf{R}_{\Gamma'.\div\mathrm{Size}} \downarrow^{S[\mathsf{x}_{\Gamma'}]}(f\langle \mathsf{x}_{\Gamma'}\rangle) \searrow v$ and $\Gamma'.\div\mathrm{Size} \vdash (t\xi\uparrow)\, \mathsf{v}_0 = v : T(\xi\uparrow, \mathsf{v}_0)$. These goals, in turn, follow by induction hypothesis (2) on $\Gamma'.\div\mathrm{Size} \vdash (t\xi\uparrow)\, \mathsf{v}_0 : T(\xi\uparrow, \mathsf{v}_0) \circledR f\langle \mathsf{x}_{\Gamma'}\rangle \in A[\mathsf{x}_{\Gamma'}]$ which is an instance of our assumption $\Gamma \vdash t : T' \circledR f \in A'$. □

Corollary 5.5 (Fresh variable). *If* $\vdash \Gamma.T$ *and* $\Gamma \vdash T \circledR A$ *then* $\Gamma.T \vdash \mathsf{v}_0 : (T\uparrow) \circledR (\uparrow^{A} \mathsf{x}_{\Gamma}) \in A$.

Proof. Since $\Gamma.T \vdash \mathsf{v}_0 \doteq \mathsf{R}^{\mathsf{ne}}\, \mathsf{x}_{\Gamma} : T\uparrow$ by Lemma 5.1, the goal follows from Thm. 5.4 part (1). □

Corollary 5.6 (One-to-one).

(1) *If* $\Gamma \vdash a \doteq \mathsf{R}^{\mathsf{size}}\alpha$ *and* $\Gamma \vdash a' \doteq \mathsf{R}^{\mathsf{size}}\alpha$ *then* $a = a'$.

(2) *If* $\Gamma \vdash T \circledR A \in \mathrm{Set}_{\ell}$ *and* $\Gamma \vdash T' \circledR A \in \mathrm{Set}_{\ell'}$ *then* $\Gamma \vdash T = T'$.

Lemma 5.7 (Semantic implies judgmental subtyping [Fridlender and Pagano 2013]).

(1) *If* $\Gamma \vdash a \doteq \mathsf{R}^{\mathsf{size}}\alpha$ *and* $\Gamma \vdash b \doteq \mathsf{R}^{\mathsf{size}}\beta$ *and* $\alpha \leq \beta$ *then* $a \leq b$.

(2) *If* $\Gamma \vdash T \circledR A$ *and* $\Gamma \vdash T' \circledR A'$ *and* $A \leq A' \in \mathcal{T}ype$ *then* $\Gamma \vdash T \leq T'$.

Proof. The proof is analogous to the one for algorithmic subtyping to come (Lemma 6.2). □

Lemma 5.8 (Subsumption for the logical relation [Fridlender and Pagano 2013]). *If* $\Gamma \vdash T \circledR A$ *and* $\Gamma \vdash T' \circledR A'$ *and* $A \leq A' \in \mathcal{T}ype$ *then* $\Gamma \vdash t : T \circledR f \in A$ *implies* $\Gamma \vdash t : T' \circledR f \in A'$.

Fig. 10 defines a logical relation for substitutions $\boxed{\Gamma \vdash \sigma \risingdotseq \tau : \Delta \circledR \eta \risingdotseq \rho}$. We write $\boxed{\Gamma \vdash \tau : \Delta \circledR \rho}$ for $\Gamma \vdash \tau \risingdotseq \tau : \Delta \circledR \rho \risingdotseq \rho$.

Lemma 5.9 (Properties of the logical relation for substitutions).
Let $\Gamma \vdash \sigma \risingdotseq \tau : \Delta \circledR \eta \risingdotseq \rho$. *Then:*

$$\frac{\vdash \Gamma}{\Gamma \vdash () \doteqdot () : () \circledR () \doteqdot ()} \qquad \frac{\Gamma \vdash \sigma \doteqdot \tau : \Delta \circledR \eta \doteqdot \rho \qquad \Gamma \vdash a : \mathsf{Size} \qquad \Gamma \vdash a \doteq \mathsf{R}^{\mathsf{size}}\alpha}{\Gamma \vdash (\sigma, a) \doteqdot (\tau, a) : \Delta.\mathsf{Size} \circledR (\eta, \alpha) \doteqdot (\rho, \alpha)}$$

$$\frac{\Gamma \vdash \sigma \doteqdot \tau : \Delta \circledR \eta \doteqdot \rho \qquad \Gamma^{\oplus} \vdash a, b : \mathsf{Size} \qquad \alpha, \beta \in \mathcal{S}ize \qquad \Gamma^{\oplus} \vdash b \doteq \mathsf{R}^{\mathsf{size}}\beta}{\Gamma \vdash (\sigma, a) \doteqdot (\tau, b) : \Delta.^{\div}\mathsf{Size} \circledR (\eta, \alpha) \doteqdot (\rho, \beta)}$$

$$\frac{\Gamma \vdash \sigma \doteqdot \tau : \Delta \circledR \eta \doteqdot \rho \qquad \Delta^{\oplus} \vdash T \qquad \Gamma \vdash u = t : T\tau \qquad \Gamma \vdash t : T\tau \circledR f \in T\rho \qquad f = g \in T\rho}{\Gamma \vdash (\sigma, u) \doteqdot (\tau, t) : \Delta.T \circledR (\eta, f) \doteqdot (\rho, q)}$$

Fig. 10. Logical relation for substitutions $\boxed{\Gamma \vdash \sigma \doteqdot \tau : \Delta \circledR \eta \doteqdot \rho}$.

(1) *Well-typedness:* $\Gamma \vdash \sigma : \Delta$ *and* $\Gamma \vdash \sigma = \sigma \doteqdot \tau : \Delta$ *[which in turn implies* $\Gamma \vdash \tau : \Delta$*].*
(2) *Weakening: If* $\xi : \Gamma' \leq \Gamma$ *then* $\Gamma' \vdash \sigma\xi \doteqdot \tau\xi : \Delta \circledR \eta \doteqdot \rho$.
(3) *Resurrection:* $\Gamma^{\oplus} \vdash \tau : \Delta^{\oplus} \circledR \rho$.
(4) *Size substitution: If* $\Delta^{\oplus} \vdash a : \mathsf{Size}$ *then* $a\eta \in \mathcal{S}ize$.

Proof. For part (4), the only interesting case $a = \mathsf{v}_i + o$ can be proved by observing that $\eta(i) \in \mathcal{S}ize$. □

The following judgements are used to state the fundamental theorem of typing.

$$\Gamma \Vdash t : T \quad :\Longleftrightarrow \quad \Gamma' \vdash t\sigma : T\tau \circledR t\eta \in T\rho \text{ for all } \Gamma' \vdash \sigma \doteqdot \tau : \Gamma \circledR \eta \doteqdot \rho$$

$$\Gamma \Vdash \sigma_0 : \Delta \quad :\Longleftrightarrow \quad \Gamma' \vdash \sigma_0\sigma \doteqdot \sigma_0\tau : \Delta \circledR \sigma_0\eta \doteqdot \sigma_0\rho \text{ for all } \Gamma' \vdash \sigma \doteqdot \tau : \Gamma \circledR \eta \doteqdot \rho$$

Theorem 5.10 (Fundamental theorem of typing).

(1) *If* $\Gamma \vdash t : T$ *then* $\Gamma \Vdash t : T$.
(2) *If* $\Gamma \vdash \sigma : \Delta$ *then* $\Gamma \Vdash \sigma : \Delta$.

Proof. Each by induction on the derivation.

Case $\forall$-introduction.

$$\frac{\Gamma.^{\div}\mathsf{Size} \vdash t : T}{\Gamma \vdash \lambda t : \forall T}$$

Assume $\Delta \vdash \sigma \doteqdot \tau : \Gamma \circledR \eta \doteqdot \rho$ and show $\Delta \vdash (\lambda t)\sigma : (\forall T)\tau \circledR (\lambda t)\eta \in (\forall T)\rho$. To this end, assume a weakening $\xi : \Delta' \leq \Delta$ and size expressions $\Delta'^{\oplus} \vdash a, b : \mathsf{Size}$ and size values α, β with $\Delta'^{\oplus} \vdash b \doteq \mathsf{R}^{\mathsf{size}}\beta$ and show $\Delta' \vdash (\lambda t)\sigma\xi\langle a\rangle : T(\tau{\uparrow}, \mathsf{v}_0)(\xi, b) \circledR (\lambda t)\eta\langle\alpha\rangle \in T(\rho{\uparrow}, \mathsf{v}_0)[\beta]$. It suffices to show (weak head reduction, substitution laws) that $\Delta' \vdash t(\sigma\xi, a) : T(\tau\xi, b) \circledR t(\eta, \alpha) \in T(\rho, \beta)$. This follows from the induction hypothesis, since $\Delta' \vdash (\sigma\xi, a) \doteqdot (\tau\xi, b) : \Gamma.^{\div}\mathsf{Size} \circledR (\eta, \alpha) \doteqdot (\rho, \beta)$ by Lemma 5.9 and substitution extension.

Case $\forall$-elimination.

$$\frac{\Gamma \vdash t : \forall T \qquad \Gamma^{\oplus} \vdash a, b : \mathsf{Size}}{\Gamma \vdash t\langle a\rangle : T[b]}$$

Assume $\Delta \vdash \sigma \doteqdot \tau : \Gamma \circledR \eta \doteqdot \rho$ and show $\Delta \vdash t\langle a\rangle\sigma : T[b]\tau \circledR t\langle a\rangle\eta \in T[b]\rho$. By induction hypothesis we have $\Delta \vdash t\sigma : (\forall T)\tau \circledR t\eta \in (\forall T)\rho$. It suffices to show the following:

(1) $\Delta^{\oplus} \vdash a\sigma : \mathsf{Size}$: follows from $\Gamma^{\oplus} \vdash a : \mathsf{Size}$ via $\Delta \vdash \sigma : \Gamma$ and $\Delta^{\oplus} \vdash \sigma : \Gamma^{\oplus}$ and substitution.
(2) $a\eta \in \mathcal{S}ize$: follows from $\Gamma^{\oplus} \vdash a : \mathsf{Size}$ by Lemma 5.9, part (4).
(3) $\Delta^{\oplus} \vdash b\tau \doteq \mathsf{R}^{\mathsf{size}}b\rho$: follows from induction hypothesis on $\Gamma^{\oplus} \vdash b : \mathsf{Size}$ via $\Delta^{\oplus} \vdash \tau \doteqdot \tau : \Gamma^{\oplus} \circledR \rho \doteqdot \rho$ (from Lemma 5.9). □

Lemma 5.11 (Identity environment). *If* $\vdash \Gamma$ *then* $\Gamma \vdash \mathsf{id} : \Gamma \circledR \rho_\Gamma$.

Proof. By induction on $\vdash \Gamma$. Consider case:

$$\frac{\vdash \Gamma \qquad \Gamma^{\oplus} \vdash T}{\vdash \Gamma.T}$$

Let $f = \uparrow^{T\rho_\Gamma} \mathsf{x}_\Gamma$. We have to show $\Gamma.T \vdash \mathsf{id} : \Gamma.T \circledR (\rho_\Gamma, f)$. Note that $\mathsf{id} = (\uparrow, \mathsf{v}_0)$, thus it remains to show that $\Gamma.T \vdash \uparrow : \Gamma \circledR \rho_\Gamma$ and $\Gamma.T \vdash \mathsf{v}_0 : T\!\uparrow \circledR f \in T\rho_\Gamma$. The first goal follows by weakening from the induction hypothesis $\Gamma \vdash \mathsf{id} : \Gamma \circledR \rho_\Gamma$. Since $\rho_\Gamma \in \Gamma^{\oplus}$, we get $T\rho_\Gamma \in \mathcal{S}et_\ell$ for some ℓ by the first fundamental theorem (Thm. 4.15). Thus, the second goal follows by Cor. 5.5. □

Corollary 5.12 (Soundness of NbE).

(1) *If* $\Gamma \vdash t : T$ *then* $\Gamma \vdash t = \mathsf{nbe}^T_\Gamma t : T$.
(2) *If* $\Gamma \vdash t, t' : T$ *and* $\mathsf{nbe}^T_\Gamma t \approx \mathsf{nbe}^T_\Gamma t'$ *then* $\Gamma \vdash t = t' : T$.

Proof. (1) For the identity environment $\Gamma \vdash \mathsf{id} : \Gamma \circledR \rho_\Gamma$ (Lemma 5.11) the Fundamental Theorem for Typing gives $\Gamma \vdash t : T \circledR t\rho_\Gamma \in T\rho_\Gamma$. This implies $\mathsf{R}_\Gamma \downarrow^{(T\rho_\Gamma)}(t\rho_\Gamma) \searrow v$ for some normal form v and $\Gamma \vdash t = v : T$ by Thm. 5.4. Then (2): From (1), using Lemma 3.1: $\Gamma \vdash t = \mathsf{nbe}^T_\Gamma t = \mathsf{nbe}^T_\Gamma t' = t' : T$. □

Corollary 5.13 (Decidability of judgemental equality). *If* $\Gamma \vdash t, t' : T$ *then the test whether* $\mathsf{nbe}^T_\Gamma t \approx \mathsf{nbe}^T_\Gamma t'$ *terminates and decides* $\Gamma \vdash t = t' : T$.

Proof. Follows directly from soundness (including termination) and completeness of NbE. □

For normalization of types, we introduce $\boxed{\mathsf{Nbe}_\Gamma T}$ via the relation $\mathsf{Nbe}_\Gamma T \searrow V :\Longleftrightarrow \mathsf{R}^{\mathsf{ty}}_\Gamma\, T\rho_\Gamma \searrow V$. Analogously, we show that the test $\mathsf{Nbe}_\Gamma T \approx \mathsf{Nbe}_\Gamma T'$ decides type equality $\Gamma \vdash T = T'$.

Corollary 5.14 (Type weak head normalization). *If* $\Gamma \vdash T$ *then* $T \searrow W$ *for some* W.

Proof. By the fundamental theorem of typing, $\Gamma \vdash T \circledR T\rho_\Gamma$, which implies $T \searrow W$ by definition of the logical relation. □

Corollary 5.15 (Type constructor injectivity).

(1) *If* $\Gamma \vdash \mathsf{Set}_\ell = \mathsf{Set}_{\ell'} : s$ *then* $\ell = \ell'$.
(2) *If* $\Gamma \vdash \mathsf{Nat}\, a = \mathsf{Nat}\, b : s$ *then* $a = b$.
(3) *If* $\Gamma \vdash \Pi\, U\, T = \Pi\, U'\, T' : s$ *then* $\Gamma \vdash U = U' : s$ *and* $\Gamma.U \vdash T = T' : s$.
(4) *If* $\Gamma \vdash \Pi^\star \mathsf{Size}\, T = \Pi\, \mathsf{Size}\, T' : s$ *then* $\Gamma.\mathsf{Size} \vdash T = T' : s$.

Proof. Statement (1) follows by inversion on $\mathsf{Set}_\ell = \mathsf{Set}_{\ell'} \in s$, which is a direct consequence of the fundamental theorem (Thm. 4.15).

For (2), observe that $\mathsf{Nat}\,(a\rho_\Gamma) = \mathsf{Nat}\,(b\rho_\Gamma) \in s$ by the fundamental theorem (Thm. 4.15), which by definition implies $a\rho_\Gamma = b\rho_\Gamma$. By the fundamental theorem of typing (Thm. 5.10), $\Gamma \vdash \mathsf{Nat}\, a \circledR \mathsf{Nat}\,(a\rho_\Gamma) \in s$ which by definition implies $\Gamma \vdash a \doteq \mathsf{R}^{\mathsf{size}} a\rho_\Gamma$. Analogously, we get $\Gamma \vdash b \doteq \mathsf{R}^{\mathsf{size}} b\rho_\Gamma$. By Cor. 5.6, $a = b$.

Last, we prove statement (3), the last statement follows analogously. By the fundamental theorem, $(\Pi\, U\, T)\rho_\Gamma = (\Pi\, U'\, T')\rho_\Gamma \in s$ which by definition means $U\rho_\Gamma = U'\rho_\Gamma \in s$ and $T(\rho_\Gamma, u) = T'(\rho_\Gamma, u') \in s$ for all $u = u' \in s$. By reification (Thm. 4.9), $\mathsf{R}^{\mathsf{ty}}_\Gamma\, U\rho_\Gamma \searrow V$ and $\mathsf{R}^{\mathsf{ty}}_\Gamma\, U'\rho_\Gamma \searrow V'$ for some normal forms $V \approx V'$. Since by inversion we have $\Gamma \vdash U, U' : s$ we get $\Gamma \vdash U = U' : s$ by soundness of NbE. Now, choosing $u := u' := \uparrow^{U\rho_\Gamma} \mathsf{x}_\Gamma \in U$, we obtain $T\rho_{\Gamma.U} = T'\rho_{\Gamma.U} \in s$, which analogously gives us $\Gamma.U \vdash T = T' : s$. □

Lemma 5.16 (Type constructor discrimination). *Different type constructors are not related by subtyping. For instance,* $\Gamma \vdash \mathsf{Nat}\, a \leq \Pi\, U\, T$ *and* $\Gamma \vdash \Pi^{\div} \mathsf{Size}\, T \leq \Pi\, \mathsf{Size}\, T'$ *are both impossible.*

Proof. Follows directly by the fundamental theorem (Thm. 4.15) and inversion on semantic subtyping. □

Lemma 5.17 (Inversion of subtyping [Fridlender and Pagano 2013]).

(1) *If* $\Gamma \vdash \mathsf{Set}_\ell \leq \mathsf{Set}_{\ell'} : s$ *then* $\ell \leq \ell'$.
(2) *If* $\Gamma \vdash \mathsf{Nat}\, a \leq \mathsf{Nat}\, b : s$ *then* $a \leq b$.
(3) *If* $\Gamma \vdash \Pi\, U\, T \leq \Pi\, U'\, T' : s$ *then* $\Gamma \vdash U' \leq U : s$ *and* $\Gamma.U' \vdash T \leq T' : s$.
(4) *If* $\Gamma \vdash \Pi^\star \mathsf{Size}\, T \leq \Pi\, \mathsf{Size}\, T' : s$ *then* $\Gamma.\mathsf{Size} \vdash T \leq T' : s$.

Proof. Similar to the proof of Cor. 5.15, using Lemma 5.7. We recapitulate the proof for (3). By the fundamental theorems, we get $\Gamma \vdash \Pi\, U\, T \circledR (\Pi\, U\, T)\rho_\Gamma$ and $\Gamma \vdash \Pi\, U'\, T' \circledR (\Pi\, U'\, T')\rho_\Gamma$ and $(\Pi\, U\, T)\rho_\Gamma \leq (\Pi\, U'\, T')\rho_\Gamma \in \mathcal{T}ype$. By inversion, first $\Gamma \vdash U \circledR U\rho_\Gamma$ and $\Gamma \vdash U' \circledR U'\rho_\Gamma$ and $U'\rho_\Gamma \leq U\rho_\Gamma \in \mathcal{T}ype$ which imply $\Gamma \vdash U' \leq U$ by Lemma 5.7. Recall that $\rho_{\Gamma.U'} = (\rho_\Gamma, u)$ where $u = \uparrow^{U'\rho_\Gamma} \mathsf{x}_\Gamma \in U'\rho_\Gamma$ and $\Gamma.U' \vdash \mathsf{v}_0 : U'\!\uparrow \circledR u \in U'\rho_\Gamma$ by Cor. 5.5. From $\Gamma.U' \vdash T \circledR T(\rho_{\Gamma.U'}) \in s$ and $\Gamma.U' \vdash T' \circledR T'(\rho_{\Gamma.U'}) \in s$ and $T(\rho_{\Gamma.U'}) \leq T'(\rho_{\Gamma.U'}) \in s$, again by Lemma 5.7, $\Gamma.U' \vdash T \leq T' : s$. □

Lemma 5.18 (Strong inversion for abstraction).

(1) *If* $\Gamma \vdash \lambda t : \Pi\, U\, T$ *then* $\Gamma.U \vdash t : T$.
(2) *If* $\Gamma \vdash \lambda t : \Pi^\star \mathsf{Size}\, T$ *then* $\Gamma.{}^\star\mathsf{Size} \vdash t : T$.

Proof. By inversion of typing (Lemma 3.8), type constructor discrimination (Lemma 5.16) and inversion of subtyping (Lemma 5.17).

For instance, inversion on $\Gamma \vdash \lambda t : \Pi\, U\, T$ gives us $\Gamma.U' \vdash t : T'$ with $\Gamma \vdash \Pi\, U'\, T' \leq \Pi\, U\, T$ (the other case, $\Pi^\star \mathsf{Size}\, T$ is excluded by discrimination). By inversion of subtyping, $\Gamma \vdash U \leq U'$ and $\Gamma.U \vdash T' \leq T$. Since $\mathrm{id} : \Gamma.U \leq \Gamma.U'$, we have $\Gamma.U \vdash t : T'$ by weakening, and our goal follows by subsumption. □

Lemma 5.19 (Strong inversion of redexes).

(1) *If* $\Gamma \vdash (\lambda t)\, u : T'$ *then* $\Gamma.U \vdash t : T$ *and* $\Gamma \vdash u : U$ *and* $\Gamma \vdash T[u] \leq T'$ *for some* U, T.
(2) *If* $\Gamma \vdash (\lambda t)\, a : T'$ *then* $\Gamma.\mathsf{Size} \vdash t : T$ *and* $\Gamma \vdash a : \mathsf{Size}$ *and* $\Gamma \vdash T[a] \leq T'$ *for some* T.
(3) *If* $\Gamma \vdash (\lambda t)\, \langle a \rangle : T'$ *then* $\Gamma.{}^\div\mathsf{Size} \vdash t : T$ *and* $\Gamma^\oplus \vdash a : \mathsf{Size}$ *and* $\Gamma \vdash T[a] \leq T'$ *for some* T.

Proof. For (3), from $\Gamma \vdash (\lambda t)\langle a \rangle : T'$ we get $\Gamma \vdash \lambda t : \forall T$ and $\Gamma^\oplus \vdash a, b : \mathsf{Size}$ and $\Gamma \vdash T[b] \leq T'$. Strong inversion for abstraction gives us $\Gamma.{}^\div\mathsf{Size} \vdash t : T$. Since $\Gamma \vdash [a] = [a] \doteq [b] : \Gamma.{}^\div\mathsf{Size}$, the substitution lemma yields $\Gamma \vdash t[a] : T[b]$, thus $\Gamma \vdash t[a] : T'$ by subsumption. □

Theorem 5.20 (Subject reduction).

(1) *If* $\mathcal{D} :: \Gamma \vdash t : T$ *and* $\mathcal{D}' :: t \searrow w$ *then* $\Gamma \vdash t = w : T$.
(2) *If* $\mathcal{D} :: \Gamma \vdash w\, e : T$ *and* $\mathcal{D}' :: w@e \searrow w'$ *then* $\Gamma \vdash w\, e = w : T$.

Proof. Simultaneously by induction on $\mathcal{D}'$ using inversion (Lemma 3.8) and strong inversion (Lemma 5.19) on the typing derivation $\mathcal{D}$. □

6 ALGORITHMIC SUBTYPING

Fig 11 defines an incremental subtyping algorithm $\boxed{\Gamma \vdash T <: T'}$. Neutral types are subtypes iff they are equal, which is checked using NbE.

Lemma 6.1 (Soundness of algorithmic subtyping). *If* $\Gamma \vdash T <: T'$ *then* $\Gamma \vdash T \leq T'$.

Proof. By induction on $\Gamma \vdash T <: T'$, soundness of NbE, and subject reduction (Thm. 5.20). □

$$\frac{T \searrow n \qquad T' \searrow n' \qquad \mathsf{Nbe}_\Gamma n \approx \mathsf{Nbe}_\Gamma n'}{\Gamma \vdash T <: T'} \qquad \frac{T \searrow \mathsf{Set}_\ell \qquad T' \searrow \mathsf{Set}_{\ell'}}{\Gamma \vdash T <: T'}\ \ell \leq \ell'$$

$$\frac{T \searrow \mathsf{Nat}\, a \qquad T' \searrow \mathsf{Nat}\, a'}{\Gamma \vdash T <: T'}\ a \leq a' \qquad \frac{T_1' \searrow \Pi^\star \mathsf{Size}\, T_1 \qquad T_2' \searrow \Pi^\star \mathsf{Size}\, T_2 \qquad \Gamma.\mathsf{Size} \vdash T_1 <: T_2}{\Gamma \vdash T_1' <: T_2'}$$

$$\frac{T_1' \searrow \Pi\, U_1\, T_1 \qquad T_2' \searrow \Pi\, U_2\, T_2 \qquad \Gamma \vdash U_2 <: U_1 \qquad \Gamma.U_2 \vdash T_1 <: T_2}{\Gamma \vdash T_1' <: T_2'}$$

Fig. 11. Algorithmic subtyping $\boxed{\Gamma \vdash T <: T'}$.

Lemma 6.2 (Semantic subtyping implies algorithmic subtyping). *If* $\Gamma \vdash T \circledR A$ *and* $\Gamma \vdash T' \circledR A'$ *and* $A \leq A' \in \mathcal{T}ype$ *then* $\Gamma \vdash T <: T'$.

Proof. By induction on $\Gamma \vdash T_1' \circledR A_1' \in s_1$ and $\Gamma \vdash T_2' \circledR A_2' \in s_2$ and cases on $A_1' \leq A_2' \in \mathcal{T}ype$ we prove $\Gamma \vdash T_1' <: T_2'$.

Case $A_i' \searrow N_i$ and $T_i' \searrow n_i$ and $N_1 = N_2 \in \mathcal{N}e$ and $\Gamma \vdash n_i \doteq \mathsf{R}^{\mathsf{ty}}\, N_i : s_i$. By Cor. 5.6 $\Gamma \vdash n_1 = n_2$, thus, by completeness of NbE, $\mathsf{Nbe}_\Gamma n_1 \approx \mathsf{Nbe}_\Gamma n_2$, which entails $\Gamma \vdash T_1' <: T_2'$.

Case $A_i' \searrow \Pi\, A_i\, B_i$ and $T_i' \searrow \Pi\, U_i\, T_i$. From $\Gamma \vdash U_i \circledR A_i \in s_i$ and $A_2 \leq A_1 \in \mathcal{T}ype$ we get $\Gamma \vdash U_2 <: U_1$ by induction hypothesis on $A_i \in s_i$. Let $a := \uparrow^{A_2} \mathsf{x}_\Gamma$. With $\Gamma.U_2 \vdash \mathsf{v}_0 : (U_2\uparrow) \circledR a \in A_2$ we get $\Gamma.U_2 \vdash T_i \circledR B_i[a] \in s$ and $B_1[a] \leq B_2[a] \in \mathcal{T}ype$. Thus by induction hypothesis on $B_i[a] \in s_i$ we obtain $\Gamma.U_2 \vdash T_1 \leq T_2$, together $\Gamma \vdash T_1' <: T_2'$. □

Corollary 6.3 (Completeness of algorithmic subtyping). *If* $\Gamma \vdash T \leq T'$ *then* $\Gamma \vdash T <: T'$.

Proof. By the fundamental theorems $\Gamma \vdash T \circledR T\rho_\Gamma$ and $\Gamma \vdash T' \circledR T'\rho_\Gamma$ and $T\rho_\Gamma \leq T'\rho_\Gamma \in \mathcal{T}ype$. By Lemma 6.2, $\Gamma \vdash T <: T'$. □

Since the algorithmic subtyping relation is equivalent to the declarative one, we can freely swap one relation for the other.

Lemma 6.4 (Termination of algorithmic subtyping). *If* $\Gamma \vdash T \circledR A$ *and* $\Gamma \vdash T' \circledR A'$ *then the query* $\Gamma \vdash T <: T'$ *terminates.*

Proof. By induction on $A \in s$ and $A' \in s'$ and cases on $\Gamma \vdash T \circledR A$ and $\Gamma \vdash T' \circledR A'$. □

Theorem 6.5 (Decidability of subtyping). *If* $\Gamma \vdash T, T'$, *then* $\Gamma \vdash T \leq T'$ *is decided by the query* $\Gamma \vdash T <: T'$.

Proof. By the fundamental theorem of typing, $\Gamma \vdash T \circledR A$ and $\Gamma \vdash T' \circledR A'$, thus, the query $\Gamma \vdash T <: T'$ terminates by Lemma 6.4. If successfully, then $\Gamma \vdash T \leq T'$ by soundness of algorithmic equality. Otherwise $\Gamma \vdash T \leq T'$ is impossible by completeness of algorithmic equality. □

7 TYPE CHECKING

In this section, we show that type checking for normal forms is decidable, and succeeds for those which can be typed via the restricted rule for size polymorphism elimination:

$$\frac{\Gamma \vdash_{\mathsf{s}} t : \forall T \qquad \Gamma^{\oplus} \vdash a : \mathsf{Size}}{\Gamma \vdash_{\mathsf{s}} t\,\langle a\rangle : T[a]}$$

We refer to the restricted typing judgement as $\boxed{\Gamma \vdash_{\mathsf{s}} t : T}$, and obviously, if $\Gamma \vdash_{\mathsf{s}} t : T$ then $\Gamma \vdash t : T$.

Checking $\boxed{\Gamma \vdash t \leftleftarrows T}$. Input: Γ, t, T. Output: *yes/no.*

$$\frac{T' \searrow s \qquad \Gamma \vdash a : \mathsf{Size}}{\Gamma \vdash \mathsf{Nat}\, a \leftleftarrows T'} \qquad \frac{T' \searrow \mathsf{Set}_{\ell'} \qquad \ell < \ell'}{\Gamma \vdash \mathsf{Set}_{\ell} \leftleftarrows T'}$$

$$\frac{T' \searrow s \qquad \Gamma \vdash U \leftleftarrows s \qquad \Gamma.U \vdash T \leftleftarrows s}{\Gamma \vdash \Pi\, U\, T \leftleftarrows T'} \qquad \frac{T' \searrow s \qquad \Gamma.\mathsf{Size} \vdash T \leftleftarrows s}{\Gamma \vdash \Pi^{\star} \mathsf{Size}\, T \leftleftarrows T'}$$

$$\frac{T' \searrow \mathsf{Nat}\, b \qquad \Gamma^{\oplus} \vdash a+1 \le b : \mathsf{Size}}{\Gamma \vdash \mathsf{zero}\langle a\rangle \leftleftarrows T'} \quad \frac{T' \searrow \mathsf{Nat}\, b \qquad \Gamma^{\oplus} \vdash a+1 \le b : \mathsf{Size} \qquad \Gamma \vdash t \leftleftarrows \mathsf{Nat}\, a}{\Gamma \vdash \mathsf{suc}\langle a\rangle t \leftleftarrows T'}$$

$$\frac{T' \searrow \Pi^{\star} U\, T \qquad \Gamma.^{\star}U \vdash t \leftleftarrows T}{\Gamma \vdash \lambda t \leftleftarrows T'} \qquad \frac{\Gamma \vdash t \rightrightarrows T \qquad \Gamma \vdash T <: T'}{\Gamma \vdash t \leftleftarrows T'}$$

Inference $\boxed{\Gamma \vdash t \rightrightarrows T}$. Input: Γ, t. Output: T or *no.*

$$\frac{\Gamma(i) = {}^{:}T}{\Gamma \vdash \mathsf{v}_i \rightrightarrows T} \qquad \frac{\Gamma \vdash t \rightrightarrows T' \qquad T' \searrow \Pi\, U\, T \qquad \Gamma \vdash u \leftleftarrows U}{\Gamma \vdash t\, u \rightrightarrows T[u]}$$

$$\frac{\Gamma \vdash t \rightrightarrows T' \qquad T' \searrow \Pi\, \mathsf{Size}\, T \qquad \Gamma \vdash a : \mathsf{Size}}{\Gamma \vdash t\, a \rightrightarrows T[a]} \quad \frac{\Gamma \vdash t \rightrightarrows T' \qquad T' \searrow \Pi^{\div} \mathsf{Size}\, T \qquad \Gamma^{\oplus} \vdash a : \mathsf{Size}}{\Gamma \vdash t\, \langle a\rangle \rightrightarrows T[a]}$$

$$\frac{\begin{array}{c}\Gamma \vdash u \rightrightarrows \mathsf{Nat}\,(a+1) \\ \Gamma^{\oplus} \vdash T \leftleftarrows \mathsf{Nat}\,(a+1) \to \mathsf{Set}_{\ell} \qquad \Gamma \vdash t_z \leftleftarrows T\,(\mathsf{zero}\langle a\rangle) \qquad \Gamma \vdash t_s \leftleftarrows (x{:}\mathsf{Nat}\, a) \to T\,(\mathsf{suc}\langle a\rangle x)\end{array}}{\Gamma \vdash u\, \mathsf{case}_{\ell}\, T\, t_z\, t_s \rightrightarrows T\, u}$$

$$\frac{\Gamma \vdash u \rightrightarrows \mathsf{Nat}\, a \qquad \Gamma^{\oplus} \vdash T \leftleftarrows \mathsf{FixK}\, \ell \qquad \Gamma \vdash t \leftleftarrows \mathsf{FixT}\, T}{\Gamma \vdash u\, \mathsf{fix}_{\ell}\, T\, t \rightrightarrows T\, a\, u}$$

Fig. 12. Bidirectional type-checking of normal forms.

Figure 12 displays the rules for bidirectional typing of normal forms. Note that we could go beyond normal forms, by adding inference rules for the Nat -constructors:

$$\frac{\Gamma^{\oplus} \vdash a : \mathsf{Size}}{\Gamma \vdash \mathsf{zero}\langle a\rangle \rightrightarrows \mathsf{Nat}\,(a+1)} \qquad \frac{\Gamma^{\oplus} \vdash a : \mathsf{Size} \qquad \Gamma \vdash t \leftleftarrows \mathsf{Nat}\, a}{\Gamma \vdash \mathsf{suc}\langle a\rangle t \rightrightarrows \mathsf{Nat}\,(a+1)}$$

Theorem 7.1 (Soundness of type checking). *Let* $\vdash \Gamma$.

(1) *If* $\Gamma^{\oplus} \vdash T$ *and* $\mathcal{D} :: \Gamma \vdash t \leftleftarrows T$ *then* $\Gamma \vdash_{\mathsf{s}} t : T$.
(2) *If* $\mathcal{D} :: \Gamma \vdash t \rightrightarrows T$ *then* $\Gamma^{\oplus} \vdash T$ *and* $\Gamma \vdash_{\mathsf{s}} t : T$.

Proof. Simultaneously by induction on $\mathcal{D}$, using subject reduction (Thm. 5.20) and soundness of algorithmic subtyping (Lemma 6.1). □

Lemma 7.2 (Weak head reduction of subtypes). *Let* $\mathcal{D} :: \Gamma \vdash T <: T'$.

(1) *If* $T' \searrow \mathsf{Nat}\, a'$ *then* $T \searrow \mathsf{Nat}\, a$ *and* $\Gamma \vdash a <: a' : \mathsf{Size}$.
(2) *If* $T' \searrow \mathsf{Set}_{\ell'}$ *then* $T \searrow \mathsf{Set}_{\ell}$ *and* $\ell <: \ell'$.
(3) *If* $T' \searrow \Pi\, A'\, B'$ *then* $T \searrow \Pi\, A\, B$ *and* $\Gamma \vdash A' <: A$ *and* $\Gamma.A' \vdash B <: B'$.

(4) *If* $T' \searrow \Pi^\star \mathsf{Size}\, B'$ *and* $T \searrow \Pi^\star \mathsf{Size}\, B$ *and* $\Gamma.\mathsf{Size} \vdash B <: B'$.

Proof. By cases on $\mathcal{D}$, since weak head evaluation is deterministic. □

This lemma also holds in the other direction of subtyping, i. e., when $T <: T'$ and T weak head evaluates, then T' weak head evaluates to a type of the same form.

Lemma 7.3 (Weak head reduction of supertypes). *Let* $\mathcal{D} :: \Gamma \vdash T <: T'$.

(1) *If* $T \searrow \mathsf{Nat}\, a$ *then* $T' \searrow \mathsf{Nat}\, a'$ *and* $\Gamma \vdash a <: a' : \mathsf{Size}$.
(2) *If* $T \searrow \mathsf{Set}_\ell$ *then* $T' \searrow \mathsf{Set}_{\ell'}$ *and* $\ell <: \ell'$.
(3) *If* $T \searrow \Pi\, A\, B$ *then* $T' \searrow \Pi\, A'\, B'$ *and* $\Gamma \vdash A' <: A$ *and* $\Gamma.A' \vdash B <: B'$.
(4) *If* $T \searrow \Pi^\star \mathsf{Size}\, B$ *and* $T' \searrow \Pi^\star \mathsf{Size}\, B'$ *and* $\Gamma.\mathsf{Size} \vdash B <: B'$.

Proof. By cases on $\mathcal{D}$, since weak head evaluation is deterministic. □

Lemma 7.4 (Subsumption for type checking). *Let* $\mathsf{id} : \Gamma' \leq \Gamma$.

(1) *If* $\mathcal{D} :: \Gamma \vdash t \leftleftarrows T$ *and* $\Gamma^\oplus \vdash T \leq T'$ *then* $\Gamma' \vdash t \leftleftarrows T'$.
(2) *If* $\mathcal{D} :: \Gamma \vdash t \rightrightarrows T$ *then* $\Gamma' \vdash t \rightrightarrows T'$ *and* $\Gamma'^\oplus \vdash T \leq T'$.

Proof. Simultaneously by induction on $\mathcal{D}$, using lemmata 7.2 and 7.3, and soundness and completeness of algorithmic subtyping. □

Theorem 7.5 (Completeness of type checking for normal terms).

(1) *If* $\mathcal{D} :: \Gamma \vdash_{\mathsf{s}} v : T$ *then* $\Gamma \vdash v \leftleftarrows T$.
(2) *If* $\mathcal{D} :: \Gamma \vdash_{\mathsf{s}} m : T$ *then* $\Gamma \vdash m \rightrightarrows U$ *and* $\Gamma^\oplus \vdash U \leq T$.

Proof. Simultaneously by induction on $\mathcal{D}$, using (strong) inversion and Lemma 7.4. □

Lemma 7.6 (Termination of type checking). *Let* $\vdash \Gamma$.

(1) *The query* $\Gamma \vdash t \rightrightarrows ?$ *terminates.*
(2) *If* $\Gamma^\oplus \vdash T$ *then the query* $\Gamma \vdash t \leftleftarrows T$ *terminates.*

Proof. By induction on t, using type weak head normalization and soundness of type checking, to maintain well-formedness of types. And, of course, decidability of subtyping. □

Theorem 7.7 (Decidability of type checking for normal terms). *Let* $\vdash \Gamma$ *and* $\Gamma^\oplus \vdash T$. *Then* $\Gamma \vdash_{\mathsf{s}} v : T$ *is decided by* $\Gamma \vdash v \leftleftarrows T$.

8 DISCUSSION AND CONCLUSIONS

In this article, we have described the first successful integration of higher-rank size polymorphism into a core type theory with dependent function types, a sized type of natural numbers, a predicative hierarchy of universes, subtyping, and η-equality. This is an important stepping stone for the smooth integration of sized types into dependently-typed proof assistants. In these final paragraphs, we discuss some questions and insights that follow from our work and go beyond it.

It is now straightforward to add a unit type $\mathbf{1}$ with extensional equality $t = * : \mathbf{1}$ for all $t : \mathbf{1}$. We simply extend reification such that $\downarrow^1 a = *$. Further, $\mathbf{1}$ is a new type shape with rule $\mathbf{1} \lesssim \mathbf{1}$.

In the long run, we wish for a type-directed equality check that does not do normalization in one go, but interleaves weak head normalization with structural comparison. Such an equality test is at the heart of Agda's type checker and it generates constraints for meta variables involved in type reconstruction [Norell 2007]. However, the usual bidirectional construction [Abel and Scherer 2012] does not seem to go through as we lack uniqueness of types (and even principal types).

For now, we have only exploited shape-irrelevance of sized types, but this directly extends to universe levels. If we consider all universes as a single shape $\mathrm{Set}_{\ell_1} \lneqq \mathrm{Set}_{\ell_2}$, we can quantify over levels irrelevantly, as Set is a shape-irrelevant type constructor. This is a stepping stone for integrating universe cumulativity with Agda's explicit universe-polymorphism. If levels are no longer unique (because of subsumption), they will get in the way of proofs, analogously to sizes. With an irrelevant quantifier we can ignore levels where they do not matter. We will still respect them where they matter, thus, we keep consistency.

Our reflections on level irrelevance lead us to the question: can a type theory T with a stratified universe hierarchy be understood as a sort of refinement of the inconsistent System U (Type:Type)? Intuitively, when checking two terms of T for equality, could we ignore the stratification in the type A which directs the equality check (thus, consider A coming from U)? Such a perspective would put stratification in one pot with size assignment: Size annotations and levels are both just annotations for the termination checker, but do not bear semantic relevance. We could switch the universe checker temporarily off as we do with the termination checker—cf. the work of Stump et al. [2010] on *termination casts*.

Finally, we would like a general theory of shape-irrelevance that extends beyond size-indexed types. For instance, any data type constructor could be considered shape-irrelevant in all its indices, with the consequence that index arguments in the data constructors could be declared irrelevant. However, our notion of judgmental equality does not support irrelevant arguments of dependent type. It works for the non-dependent type Size, but we also relied on having a closed inhabitant ∞ in Size. More research is needed to tell a more general story of shape-irrelevance.

ACKNOWLEDGMENTS

This material is based upon work supported by the Swedish Research Council (Vetenskapsrådet) under Grant No. 621-2014-4864 *Termination Certificates for Dependently-Typed Programs and Proofs via Refinement Types*. The first author is grateful for recent discussions with Thierry Coquand, Nils Anders Danielsson, and Sandro Stucki which helped clarifying the thoughts leading to this work. He also acknowledges past discussions with Christoph-Simon Senjak. The incentive to write this article came during the EU Cost Action CA15123 EUTYPES meeting in Ljubljana in January 2017; thanks to Andrej Bauer for organizing it.

REFERENCES

Andreas Abel. 2008. Semi-continuous Sized Types and Termination. *Logical Methods in Computer Science* 4, 2:3 (2008), 1–33. https://doi.org/10.2168/LMCS-4(2:3)2008

Andreas Abel. 2010. Towards Normalization by Evaluation for the $\beta\eta$-Calculus of Constructions. In *Functional and Logic Programming, 10th International Symposium, FLOPS 2010, Sendai, Japan, April 19-21, 2010. Proceedings (Lecture Notes in Computer Science)*, Matthias Blume, Naoki Kobayashi, and Germán Vidal (Eds.), Vol. 6009. Springer, 224–239. https://doi.org/10.1007/978-3-642-12251-4_17

Andreas Abel. 2012. Type-Based Termination, Inflationary Fixed-Points, and Mixed Inductive-Coinductive Types. In *Proceedings of the 8th Workshop on Fixed Points in Computer Science (FICS 2012) (Electronic Proceedings in Theoretical Computer Science)*, Dale Miller and Zoltán Ésik (Eds.), Vol. 77. 1–11. http://dx.doi.org/10.4204/EPTCS.77.1

Andreas Abel. 2013. *Normalization by Evaluation: Dependent Types and Impredicativity.* Unpublished. http://www.tcs.ifi.lmu.de/~abel/habil.pdf

Andreas Abel and Thorsten Altenkirch. 2002. A Predicative Analysis of Structural Recursion. *Journal of Functional Programming* 12, 1 (2002), 1–41. https://doi.org/10.1017/S0956796801004191

Andreas Abel, Thierry Coquand, and Peter Dybjer. 2007. Normalization by Evaluation for Martin-Löf Type Theory with Typed Equality Judgements. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*. IEEE Computer Society Press, 3–12. https://doi.org/10.1109/LICS.2007.33

Andreas Abel, Thierry Coquand, and Miguel Pagano. 2009. A Modular Type-Checking Algorithm for Type Theory with Singleton Types and Proof Irrelevance. In *Typed Lambda Calculi and Applications, 9th International Conference, TLCA*

2009, Brasilia, Brazil, July 1-3, 2009, Proceedings (Lecture Notes in Computer Science), Pierre-Louis Curien (Ed.), Vol. 5608. Springer, 5–19. https://doi.org/10.1007/978-3-642-02273-9_3

Andreas Abel, Thierry Coquand, and Miguel Pagano. 2011. A Modular Type-Checking Algorithm for Type Theory with Singleton Types and Proof Irrelevance. *Logical Methods in Computer Science* 7, 2:4 (2011), 1–57. https://doi.org/10.2168/LMCS-7(2:4)2011

Andreas Abel and Brigitte Pientka. 2016. Well-founded recursion with copatterns and sized types. *Journal of Functional Programming* 26 (2016), 61. https://doi.org/10.1017/S0956796816000022

Andreas Abel and Gabriel Scherer. 2012. On Irrelevance and Algorithmic Equality in Predicative Type Theory. *Logical Methods in Computer Science* 8, 1:29 (2012), 1–36. https://doi.org/10.2168/LMCS-8(1:29)2012

AgdaTeam. 2017. The Agda Wiki. (2017). http://wiki.portal.chalmers.se/agda

Roberto M. Amadio (Ed.). 2008. *Foundations of Software Science and Computational Structures, 11th International Conference, FoSSaCS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 - April 6, 2008. Proceedings.* Lecture Notes in Computer Science, Vol. 4962. Springer. https://doi.org/10.1007/978-3-540-78499-9

Roberto M. Amadio and Solange Coupet-Grimal. 1998. Analysis of a Guard Condition in Type Theory (Extended Abstract).. In *Foundations of Software Science and Computation Structure, First International Conference, FoSSaCS'98, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98, Lisbon, Portugal, March 28 - April 4, 1998, Proceedings (Lecture Notes in Computer Science)*, Maurice Nivat (Ed.), Vol. 1378. Springer, 48–62. https://doi.org/10.1007/BFb0053541

Henk Barendregt. 1991. Introduction to Generalized Type Systems. *Journal of Functional Programming* 1, 2 (1991), 125–154.

Bruno Barras and Bruno Bernardo. 2008. The Implicit Calculus of Constructions as a Programming Language with Dependent Types, See [Amadio 2008], 365–379. https://doi.org/10.1007/978-3-540-78499-9_26

Gilles Barthe, Maria João Frade, Eduardo Giménez, Luís Pinto, and Tarmo Uustalu. 2004. Type-Based Termination of Recursive Definitions. *Mathematical Structures in Computer Science* 14, 1 (2004), 97–141. https://doi.org/10.1017/S0960129503004122

Gilles Barthe, Benjamin Grégoire, and Fernando Pastawski. 2006. CIC^: Type-Based Termination of Recursive Definitions in the Calculus of Inductive Constructions. In *Logic for Programming, Artificial Intelligence, and Reasoning, 13th International Conference, LPAR 2006, Phnom Penh, Cambodia, November 13-17, 2006, Proceedings (Lecture Notes in Computer Science)*, Miki Hermann and Andrei Voronkov (Eds.), Vol. 4246. Springer, 257–271. https://doi.org/10.1007/11916277_18

Gilles Barthe, Benjamin Grégoire, and Colin Riba. 2008a. A Tutorial on Type-Based Termination. In *LerNet ALFA Summer School (Lecture Notes in Computer Science)*, Ana Bove, Luís Soares Barbosa, Alberto Pardo, and Jorge Sousa Pinto (Eds.), Vol. 5520. Springer, 100–152. https://doi.org/10.1007/978-3-642-03153-3_3

Gilles Barthe, Benjamin Grégoire, and Colin Riba. 2008b. Type-Based Termination with Sized Products. In *Computer Science Logic, 22nd International Workshop, CSL 2008, 17th Annual Conference of the EACSL, Bertinoro, Italy, September 16-19, 2008. Proceedings (Lecture Notes in Computer Science)*, Michael Kaminski and Simone Martini (Eds.), Vol. 5213. Springer, 493–507. https://doi.org/10.1007/978-3-540-87531-4_35

Ulrich Berger and Helmut Schwichtenberg. 1991. An Inverse to the Evaluation Functional for Typed λ-calculus. In *Sixth Annual Symposium on Logic in Computer Science (LICS '91), July, 1991, Amsterdam, The Netherlands, Proceedings.* IEEE Computer Society Press, 203–211. https://doi.org/10.1109/LICS.1991.151645

Frédéric Blanqui. 2004. A Type-Based Termination Criterion for Dependently-Typed Higher-Order Rewrite Systems. In *Rewriting Techniques and Applications, 15th International Conference, RTA 2004, Aachen, Germany, June 3 – 5, 2004, Proceedings (Lecture Notes in Computer Science)*, Vincent van Oostrom (Ed.), Vol. 3091. Springer, 24–39. https://doi.org/10.1007/978-3-540-25979-4_2

Frédéric Blanqui. 2005. Decidability of Type-Checking in the Calculus of Algebraic Constructions with Size Annotations.. In *Computer Science Logic, 19th International Workshop, CSL 2005, 14th Annual Conference of the EACSL, Oxford, UK, August 22-25, 2005, Proceedings (Lecture Notes in Computer Science)*, C.-H. Luke Ong (Ed.), Vol. 3634. Springer, 135–150. https://doi.org/10.1007/11538363_11

Frédéric Blanqui and Colin Riba. 2006. Combining Typing and Size Constraints for Checking the Termination of Higher-Order Conditional Rewrite Systems. In *Logic for Programming, Artificial Intelligence, and Reasoning, 13th International Conference, LPAR 2006, Phnom Penh, Cambodia, November 13-17, 2006, Proceedings (Lecture Notes in Computer Science)*, Miki Hermann and Andrei Voronkov (Eds.), Vol. 4246. Springer, 105–119. https://doi.org/10.1007/11916277_8

Ana Bove. 2009. Another Look at Function Domains. *Electronic Notes in Theoretical Computer Science* 249 (2009), 61–74. https://doi.org/10.1016/j.entcs.2009.07.084

Ana Bove and Venanzio Capretta. 2005. Modelling general recursion in type theory. *Mathematical Structures in Computer Science* 15, 4 (2005), 671–708. https://doi.org/10.1017/S0960129505004822

Edwin Brady. 2013. Idris, a general-purpose dependently typed programming language: Design and implementation. *Journal of Functional Programming* 23, 5 (2013), 552–593. https://doi.org/10.1017/S095679681300018X

Thierry Coquand. 1996. An Algorithm for Type-Checking Dependent Types, In Mathematics of Program Construction. Selected Papers from the Third International Conference on the Mathematics of Program Construction (July 17–21, 1995, Kloster Irsee, Germany). *Science of Computer Programming* 26, 1-3, 167–177. https://doi.org/10.1016/0167-6423(95)00021-6

Olivier Danvy. 1999. Type-Directed Partial Evaluation. In *Partial Evaluation – Practice and Theory, DIKU 1998 International Summer School, Copenhagen, Denmark, June 29 - July 10, 1998 (Lecture Notes in Computer Science)*, John Hatcliff, Torben Æ. Mogensen, and Peter Thiemann (Eds.), Vol. 1706. Springer, 367–411. https://doi.org/10.1007/3-540-47018-2_16

N. G. de Bruijn. 1972. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae* 34 (1972), 381–392.

Peter Dybjer. 2000. A General Formulation of Simultaneous Inductive-Recursive Definitions in Type Theory. *The Journal of Symbolic Logic* 65, 2 (2000), 525–549. https://doi.org/10.2307/2586554

Peter Dybjer, Bengt Nordström, and Jan M. Smith (Eds.). 1995. *Types for Proofs and Programs, International Workshop TYPES'94, Båstad, Sweden, June 6-10, 1994, Selected Papers*. Lecture Notes in Computer Science, Vol. 996. Springer. https://doi.org/10.1007/3-540-60579-7

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)*, Masahito Hasegawa (Ed.), Vol. 7941. Springer, 140–155. https://doi.org/10.1007/978-3-642-38946-7_12

Herman Geuvers. 1994. A short and flexible proof of Strong Normalization for the Calculus of Constructions, See [Dybjer et al. 1995], 14–38. https://doi.org/10.1007/3-540-60579-7_2

Eduardo Giménez. 1995. Codifying Guarded Definitions with Recursive Schemes, See [Dybjer et al. 1995], 39–59. https://doi.org/10.1007/3-540-60579-7_3

Benjamin Grégoire and Xavier Leroy. 2002. A compiled implementation of strong reduction. In *Proceedings of the Seventh ACM SIGPLAN International Conference on Functional Programming (ICFP '02), Pittsburgh, Pennsylvania, USA, October 4-6, 2002 (SIGPLAN Notices)*, Vol. 37. ACM Press, 235–246. https://doi.org/10.1145/581478.581501

Benjamin Grégoire and Jorge Luis Sacchini. 2010. On Strong Normalization of the Calculus of Constructions with Type-Based Termination. In *Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010. Proceedings (Lecture Notes in Computer Science)*, Christian G. Fermüller and Andrei Voronkov (Eds.), Vol. 6397. Springer, 333–347. https://doi.org/10.1007/978-3-642-16242-8_24

Robert Harper and Frank Pfenning. 2005. On Equivalence and Canonical Forms in the LF Type Theory. *ACM Transactions on Computational Logic* 6, 1 (2005), 61–101. https://doi.org/10.1145/1042038.1042041

Gérard P. Huet. 1989. The Constructive Engine. In *A Perspective in Theoretical Computer Science - Commemorative Volume for Gift Siromoney*, R. Narasimhan (Ed.). World Scientific Series in Computer Science, Vol. 16. World Scientific, 38–69. https://doi.org/10.1142/9789814368452_0004

John Hughes, Lars Pareto, and Amr Sabry. 1996. Proving the Correctness of Reactive Systems Using Sized Types. In *Conference Record of POPL'96: The 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Papers Presented at the Symposium, St. Petersburg Beach, Florida, USA, January 21-24, 1996*, Hans-Juergen Boehm and Guy L. Steele Jr. (Eds.). ACM Press, 410–423. https://doi.org/10.1145/237721.240882

INRIA. 2016. *The Coq Proof Assistant Reference Manual* (version 8.6 ed.). INRIA. http://coq.inria.fr/

Ugo Dal Lago and Charles Grellois. 2017. Probabilistic Termination by Monadic Affine Sized Typing. In *Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings (Lecture Notes in Computer Science)*, Hongseok Yang (Ed.), Vol. 10201. Springer, 393–419. https://doi.org/10.1007/978-3-662-54434-1_15

Chin Soon Lee, Neil D. Jones, and Amir M. Ben-Amram. 2001. The Size-Change Principle for Program Termination. In *Conference Record of POPL 2001: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001*, Chris Hankin and Dave Schmidt (Eds.). ACM Press, 81–92. https://doi.org/10.1145/360204.360210

William Lovas and Frank Pfenning. 2010. Refinement Types for Logical Frameworks and Their Interpretation as Proof Irrelevance. *Logical Methods in Computer Science* 6, 4 (2010). https://doi.org/10.2168/LMCS-6(4:5)2010

Per Martin-Löf. 1975. An Intuitionistic Theory of Types: Predicative Part. In *Logic Colloquium '73*, H. E. Rose and J. C. Shepherdson (Eds.). North-Holland, 73–118.

Alexandre Miquel. 2000. A Model for Impredicative Type Systems, Universes, Intersection Types and Subtyping. In *15th Annual IEEE Symposium on Logic in Computer Science (LICS 2000), 26-29 June 2000, Santa Barbara, California, USA, Proceedings*. IEEE Computer Society Press, 18–29. https://doi.org/10.1109/LICS.2000.855752

Alexandre Miquel. 2001. The Implicit Calculus of Constructions. In *Typed Lambda Calculi and Applications, 5th International Conference, TLCA 2001, Krakow, Poland, May 2-5, 2001, Proceedings (Lecture Notes in Computer Science)*, Samson Abramsky (Ed.), Vol. 2044. Springer, 344–359. https://doi.org/10.1007/3-540-45413-6_27

Nathan Mishra-Linger and Tim Sheard. 2008. Erasure and Polymorphism in Pure Type Systems, See [Amadio 2008], 350–364. https://doi.org/10.1007/978-3-540-78499-9

Bengt Nordström. 1988. Terminating General Recursion. *BIT* 28, 3 (1988), 605–619.

Ulf Norell. 2007. *Towards a Practical Programming Language Based on Dependent Type Theory*. Ph.D. Dissertation. Department of Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden.

Frank Pfenning. 2001. Intensionality, Extensionality, and Proof Irrelevance in Modal Type Theory. In *16th IEEE Symposium on Logic in Computer Science (LICS 2001), 16-19 June 2001, Boston University, USA, Proceedings*. IEEE Computer Society Press, 221–230. https://doi.org/10.1109/LICS.2001.932499

Jorge Luis Sacchini. 2013. Type-Based Productivity of Stream Definitions in the Calculus of Constructions. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*. IEEE Computer Society Press, 233–242. https://doi.org/10.1109/LICS.2013.29

Jorge Luis Sacchini. 2014. Linear Sized Types in the Calculus of Constructions. In *Functional and Logic Programming - 12th International Symposium, FLOPS 2014, Kanazawa, Japan, June 4-6, 2014. Proceedings (Lecture Notes in Computer Science)*, Michael Codish and Eijiro Sumii (Eds.), Vol. 8475. Springer, 169–185. https://doi.org/10.1007/978-3-319-07151-0_11

Aaron Stump, Vilhelm Sjöberg, and Stephanie Weirich. 2010. Termination Casts: A Flexible Approach to Termination with General Recursion. In *Workshop on Partiality And Recursion in Interactive Theorem Provers, PAR 2010, Satellite Workshop of ITP'10 at FLoC 2010 (Electronic Proceedings in Theoretical Computer Science)*, Ana Bove, Ekaterina Komendantskaya, and Milad Niqui (Eds.), Vol. 43. 76–93. https://doi.org/10.4204/EPTCS.43.6

Martin Sulzmann, Manuel M. T. Chakravarty, Simon L. Peyton Jones, and Kevin Donnelly. 2007. System F with type equality coercions. In *Proceedings of TLDI'07: 2007 ACM SIGPLAN International Workshop on Types in Languages Design and Implementation, Nice, France, January 16, 2007*, François Pottier and George C. Necula (Eds.). ACM Press, 53–66. https://doi.org/10.1145/1190315.1190324

David Wahlstedt. 2007. *Dependent Type Theory with Parameterized First-Order Data Types and Well-Founded Recursion*. Ph.D. Dissertation. Chalmers University of Technology.

Benjamin Werner. 1992. A Normalization Proof for an Impredicative Type System with Large Eliminations over Integers. In *Proceedings of the 1992 Workshop on Types for Proofs and Programs, Båstad, Sweden, June 1992*, Bengt Nordström, Kent Petersson, and Gordon Plotkin (Eds.). 341–357. http://www.cs.chalmers.se/Cs/Research/Logic/Types/proc92.ps

Hongwei Xi. 2002. Dependent Types for Program Termination Verification. *Journal of Higher-Order and Symbolic Computation* 15, 1 (2002), 91–131. https://doi.org/10.1023/A:1019916231463