Equality, Quasi-Implicit Products, and Large Eliminations

Vilhelm Sjöberg
Computer and Information Science
University of Pennsylvania
vilhelm@cis.upenn.edu

Aaron Stump
Computer Science
The University of Iowa
astump@acm.org

Abstract

This paper presents a type theory with a form of equality reflection: provable equalities can be used to coerce the type of a term. Coercions and other annotations, including implicit arguments, are dropped during reduction of terms. We develop the metatheory for an undecidable version of the system with unannotated terms. We then devise a decidable system with annotated terms, justified in terms of the unannotated system. Finally, we show how the approach can be extended to account for large eliminations, using what we call quasi-implicit products.

1 Introduction

The main goal of this paper, as of several recent works, is to facilitate external reasoning about dependently typed programs [8, 2]. This is hampered if one must reason about specificational data occurring in terms. For example, consider the familiar example of vectors $\langle \texttt{vec}\ \phi\ l \rangle$ indexed by both the type ϕ of the elements and the length l of the vector. An example dependently typed program is the append_ϕ function (we work here with monomorphic functions, but will elide type subscripts), operating on vectors holding data of type ϕ. We can define *append* so that it has the following type, assuming a standard definition of *plus* on unary natural numbers `nat`:

$$\mathit{append}\ :\ \Pi l_1 : \texttt{nat}.\Pi l_2 : \texttt{nat}.\Pi v_1 : \langle \texttt{vec}\ \phi\ l_1 \rangle.\Pi v_2 : \langle \texttt{vec}\ \phi\ l_2 \rangle.\ \langle \texttt{vec}\ \phi\ (\mathit{plus}\ l_1\ l_2) \rangle$$

We might wish to prove that *append* is associative. In type theories such as Coq's Calculus of Inductive Constructions, we would do this by showing that the following type is inhabited:

$$\begin{array}{l}\Pi l_1 : \texttt{nat}.\Pi l_2 : \texttt{nat}.\Pi l_3 : \texttt{nat}.\Pi v_1 : \langle \texttt{vec}\ \phi\ l_1 \rangle.\Pi v_2 : \langle \texttt{vec}\ \phi\ l_2 \rangle.\Pi v_3 : \langle \texttt{vec}\ \phi\ l_3 \rangle. \\ \quad (\mathit{append}\ (\mathit{plus}\ l_1\ l_2)\ l_3\ (\mathit{append}\ l_1\ l_2\ v_1\ v_2)\ v_3) = (\mathit{append}\ l_1\ (\mathit{plus}\ l_2\ l_3)\ v_1\ (\mathit{append}\ l_2\ l_3\ v_2\ v_3))\end{array}$$

Notice how the lengths of the vectors are cluttering even the statement of this theorem. Tools like Coq allow such arguments to be elided, when they can be uniquely reconstructed. So the theorem to prove can be written in the much more palatable form:

$$\begin{array}{l}\Pi l_1 : \texttt{nat}.\Pi l_2 : \texttt{nat}.\Pi l_3 : \texttt{nat}.\Pi v_1 : \langle \texttt{vec}\ \phi\ l_1 \rangle.\Pi v_2 : \langle \texttt{vec}\ \phi\ l_2 \rangle.\Pi v_3 : \langle \texttt{vec}\ \phi\ l_3 \rangle. \\ \quad (\mathit{append}\ (\mathit{append}\ v_1\ v_2)\ v_3) = (\mathit{append}\ v_1\ (\mathit{append}\ v_2\ v_3))\end{array}$$

This is much more readable. But as others have noted, while the indices have been elided, they are not truly erased. This means that the proof of associativity of *append* must make use of associativity also of *plus*, in order for the lengths of the two vectors (on the two sides of the equation) to be equal. Indeed, even stating this equation may require some care, since the types of the two sides are not definitionally equal: one has $(\mathit{plus}\ (\mathit{plus}\ l_1\ l_2)\ l_3)$ where the other has $(\mathit{plus}\ l_1\ (\mathit{plus}\ l_2\ l_3))$. This is where techniques like heterogeneous equality come into play [6].

One solution to this problem is via intersection types, also called in this setting *implicit products*. An implicit product $\forall x : \phi.\phi'$ is the type for functions whose arguments are erased during conversion [8, 2, 7]. We seek to take these approaches further, and work with completely unannotated terms. When testing β-equivalence of terms, we will work with unannotated versions of those terms, where all type- and proof-annotations have been dropped. For associativity of *append*, the proof does not require associativity of *plus*. This is not the case in the Implicit Calculus of Constructions, for example. From the point of view of external reasoning, *append* on vectors will be indistinguishable from *append* on lists (without statically tracked length).

The $\mathtt{T}^{\mathtt{vec}}$ Type Theory. This paper studies versions of a type theory we call $\mathtt{T}^{\mathtt{vec}}$. This system is like Gödel's System T, with vectors and explicit equality proofs. We first study an undecidable version of $\mathtt{T}^{\mathtt{vec}}$ with equality reflection, where terms are completely unannotated (Section 2). We establish standard meta-theoretic results for this unannotated system (Section 3). We then devise a decidable annotated version of the language, whose soundness is justified by erasure to the unannotated system (Section 4). We consider the associativity of *append* in annotated $\mathtt{T}^{\mathtt{vec}}$, as an example (Section 4.1). This approach of studying unannotated versus annotated versions of the type theory should be contrasted with the approach taken in NuPRL, based on Martin-Löf's extensional type theory [3, 5]. There, one constructs typing derivations, as separate artifacts, for unannotated terms. Here, we unite the typing derivation and the unannotated term in a single artifact, namely the annotated term.

Large eliminations. Type-level recursion poses challenges for our approach. Because coercions by equality proofs are erased from terms, we would easily be able to assign a type to diverging or stuck terms, if we naively extended the system with large eliminations. We propose a solution based on what we call *quasi-implicit products*. These effectively serve to mark the introduction and elimination of the intersection type, and prohibit call-by-value reduction within an introduction. This saves Normalization and Progress, which would otherwise fail. We develop the meta-theory of an extension of the unannotated system with large eliminations and call-by-value reduction, including normalization (Section 5). The $\mathtt{T}^{\mathtt{vec}}$ approach has been implemented in the GURU dependently programming language, publicly available at `http://www.guru-lang.org` [9].

2 Unannotated $\mathtt{T}^{\mathtt{vec}}$

The definition of unannotated $\mathtt{T}^{\mathtt{vec}}$ uses unannotated terms a (we sometimes also write b):

$$a \ ::= \ x \mid (a\ a') \mid \lambda x.a \mid 0 \mid (S\ a) \mid (R_{\mathtt{nat}}\ a\ a'\ a'') \mid \mathtt{nil} \mid (\mathtt{cons}\ a\ a') \mid (R_{\mathtt{vec}}\ a\ a'\ a'') \mid \mathtt{join}$$

Here, x is for λ-bound variables and S is for successor (not the S combinator). $R_{\mathtt{nat}}$ is the recursor over natural numbers, and $R_{\mathtt{vec}}$ is the recursor over vectors. We have constructors `nil` and `cons` for vectors. The term construct `join` is the introduction form for equality proofs. We will not need an elimination form, since our system includes a form of equality reflection. For readability, we sometimes use meta-variable l for terms a intended as lengths of vectors. Types ϕ are defined by:

$$\phi \ ::= \ \mathtt{nat} \mid \langle \mathtt{vec}\ \phi\ a\rangle \mid \Pi x:\phi.\phi' \mid \forall x:\phi.\phi' \mid a = a'$$

The first Π-type is as usual, while the second is an intersection type abstracting a specificational x. This x need not be λ-abstracted in the corresponding term, nor supplied as an argument when that term is applied, similarly to Miquel's implicit products [7].

The reduction relation is the compatible closure under arbitrary contexts of the rules in Figure 1. Figure 2 gives type assignment rules for $\mathtt{T}^{\mathtt{vec}}$, using a standard definition of typing contexts Γ. We define $\Gamma\ Ok$ to mean that if $\Gamma \equiv \Gamma_1, x:\phi, \Gamma_2$, then $FV(\phi) \subset dom(\Gamma_1)$. We use $a \downarrow a'$ to mean that a and a' are joinable with respect to our reduction relation (i.e., there exists $\hat{a}$ such that $a \leadsto^* \hat{a}$ and $a' \leadsto^* \hat{a}$).

Perhaps surprisingly we do not track well-formedness of types, and indeed the `join` and `conv` rules can introduce untypable terms into types. However, they preserve the invariant that terms deemed equal are joinable, and that turns out to be enough to ensure type safety.

Type assignment is not syntax-directed, due to the `(conv)`, `(spec-abs)`, and `(spec-app)` rules, and not obviously decidable. This will not pose a problem here as we study the meta-theoretic

$$
\begin{array}{lll}
(\lambda x.a)\ a' & \rightsquigarrow & [a'/x]a \\
(R_{\texttt{nat}}\ a\ a'\ 0) & \rightsquigarrow & a \\
(R_{\texttt{nat}}\ a\ a'\ (S\ a'')) & \rightsquigarrow & (a'\ a''\ (R_{\texttt{nat}}\ a\ a'\ a'')) \\
(R_{\texttt{vec}}\ a\ a'\ \texttt{nil}) & \rightsquigarrow & a \\
(R_{\texttt{vec}}\ a\ a'\ (\texttt{cons}\ a_1\ a'')) & \rightsquigarrow & (a'\ a_1\ a''\ (R_{\texttt{vec}}\ a\ a'\ a''))
\end{array}
$$

Figure 1: Reduction semantics for unannotated $\texttt{T}^{\texttt{vec}}$ terms

$$\frac{\Gamma(x) \equiv \phi \quad \Gamma\, Ok}{\Gamma \vdash x : \phi}\ \texttt{var}$$

$$\frac{a \downarrow a' \quad \Gamma\, Ok}{\Gamma \vdash \texttt{join} : a = a'}\ \texttt{join} \qquad \frac{\Gamma \vdash a''' : a' = a'' \quad \Gamma \vdash a : [a'/x]\phi \quad x \notin dom(\Gamma)}{\Gamma \vdash a : [a''/x]\phi}\ \texttt{conv}$$

$$\frac{\Gamma, x : \phi' \vdash a : \phi \quad x \notin FV(a)}{\Gamma \vdash a : \forall x : \phi'.\phi}\ \texttt{spec-abs} \qquad \frac{\Gamma \vdash a : \forall x : \phi'.\phi \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash a : [a'/x]\phi}\ \texttt{spec-app}$$

$$\frac{\Gamma, x : \phi' \vdash a : \phi}{\Gamma \vdash \lambda x.a : \Pi x : \phi'.\phi}\ \texttt{abs} \qquad \frac{\Gamma \vdash a : \Pi x : \phi'.\phi \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash (a\ a') : [a'/x]\phi}\ \texttt{app}$$

$$\frac{\Gamma\, Ok}{\Gamma \vdash 0 : \texttt{nat}}\ \texttt{zero} \qquad \frac{\Gamma\, Ok}{\Gamma \vdash \texttt{nil} : \langle \texttt{vec}\ \phi\ 0 \rangle}\ \texttt{nil}$$

$$\frac{\Gamma \vdash a : \texttt{nat}}{\Gamma \vdash (S\ a) : \texttt{nat}}\ \texttt{succ} \qquad \frac{\begin{array}{l} x \notin dom(\Gamma) \\ \Gamma \vdash a'' : \texttt{nat} \\ \Gamma \vdash a : [0/x]\phi \\ \Gamma \vdash a' : \Pi y : \texttt{nat}.\Pi u : [y/x]\phi.[(S y)/x]\phi \end{array}}{\Gamma \vdash (R_{\texttt{nat}}\ a\ a'\ a'') : [a''/x]\phi}\ \texttt{Rnat}$$

$$\frac{\begin{array}{l} \Gamma \vdash a : \phi \\ \Gamma \vdash a' : \langle \texttt{vec}\ \phi\ l \rangle \end{array}}{\Gamma \vdash (\texttt{cons}\ a\ a') : \langle \texttt{vec}\ \phi\ (S\ l) \rangle}\ \texttt{cons} \qquad \frac{\begin{array}{l} x \notin dom(\Gamma) \\ \Gamma \vdash a'' : \langle \texttt{vec}\ \phi'\ l \rangle \\ \Gamma \vdash a : [0/y, \texttt{nil}/x]\phi \\ \Gamma \vdash a' : \Pi z : \phi'.\forall l : \texttt{nat}.\Pi v : \langle \texttt{vec}\ \phi'\ l \rangle.\Pi u : [l/y, v/x]\phi. \\ \qquad [(S\ l)/y, (\texttt{cons}\ z\ v)/x]\phi \end{array}}{\Gamma \vdash (R_{\texttt{vec}}\ a\ a'\ a'') : [l/y, a''/x]\phi}\ \texttt{Rvec}$$

Figure 2: Type assignment system for unannotated $\texttt{T}^{\texttt{vec}}$

properties of the system. Section 4 defines a system of annotated terms which is obviously decidable, and justifies it by translation to unannotated $\texttt{T}^{\texttt{vec}}$. We work up to syntactic identity module safe renaming of bound variables, which we denote $\equiv$.

3 Metatheory of Unannotated $\mathtt{T}^{\mathtt{vec}}$

$\mathtt{T}^{\mathtt{vec}}$ enjoys standard properties: Type Preservation, Progress (for closed terms), and Strong Normalization. These are all easily obtained, the last by dependency-erasing translation to another type theory (as done originally for LF in [4]). Here, we consider a more semantically informative approach to Strong Normalization. Omitted proofs may be found in a companion report on the second author's web page.

Theorem 1 (Type Preservation)**.** *If $\Gamma \vdash a : \phi$ and $a \rightsquigarrow a'$, then $\Gamma \vdash a' : \phi$.*

Theorem 2 (Progress)**.** *If $\Gamma \vdash a : \phi$ and $dom(\Gamma) \cap FV(a) = \emptyset$, then either a is a value or $\exists a'.a \rightsquigarrow a'$.*

3.1 Semantics of equality

For our Strong Normalization proof, a central issue is providing an interpretation for equality types in the presence of free variables. We would like to interpret equations like $(plus\ 2\ 2) = 4$ (where the numerals abbreviate terms formed with S and 0 as usual, and $plus$ has a standard recursive definition), as simply $(plus\ 2\ 2) \downarrow 4$. But when the two terms contain free variables – e.g., in $(plus\ x\ y) = (plus\ y\ x)$ – or when the context is inconsistent, the semantics should make the equation true, even though its sides are not joinable. So our semantics for equality types is joinability under all *ground instances* of the context Γ. The notation for this is $a \sim_\Gamma a'$. The definition must be given as part of the definition of the interpretation of types, because we want to stipulate that the substitutions σ replace each variable x by a ground term in the interpretation of $\sigma\Gamma(x)$. When Γ is empty, we will write $a \sim_\Gamma a'$ as $a \sim a'$. We use a similar convention for other notations subscripted by a context below.

3.2 The interpretation of types

The interpretation of types is given in Figure 3. We stipulate up front (not in the clauses in the figure) that $a \in [\![\phi]\!]_\Gamma$ requires $a \in SN$ and $\Gamma \vdash a : \phi$. The definition in Figure 3 proceeds by well-founded recursion on the triple $(|\Gamma|, d(\phi), l(a))$, in the natural lexicographic ordering. Here, $|\Gamma|$ is the cardinality of $dom(\Gamma)$, and if $a \in SN$, then we make use of a (finite) natural number $l(a)$ bounding the number of symbols in the normal form of a. We need to assume confluence of reduction elsewhere in this proof, so it does not weaken the result to assume here that each term has at most one normal form. The quantity $d(\phi)$ is the depth of ϕ, defined as follows:

$$
\begin{array}{lcllcl}
d(\mathtt{nat}) & = & 0 & d(\langle \mathtt{vec}\ \phi\ l\rangle) & = & 1 + d(\phi) \\
d(\Pi x:\phi.\phi') & = & 1 + max(d(\phi), d(\phi')) & d(\forall x:\phi.\phi') & = & 1 + max(d(\phi), d(\phi')) \\
d(a = a') & = & 0 & & &
\end{array}
$$

Note that $d(\phi) = d([a/x]\phi)$ for all a, x, and ϕ. Also, in the clause for $\mathtt{vec}$-types, since the right hand side of the clause conjoins the condition $a \in SN$, $l(a)$ is defined, and we have $l(a'') < l(\mathtt{cons}\ a'\ a'')$. The figure gives an inductive definition for when $\sigma \in [\![\Gamma]\!]_\Delta$. We call such a σ a *closable substitution*.

In general, the inductive definition of closable substitution $\sigma \in [\![\Gamma]\!]_\Delta$ allows the range of the substitution to contain open terms. When Δ is empty, σ is a *closing* substitution. The definition of $[\![\cdot]\!]$ for types uses the definition of closable substitutions in a well-founded way. We appeal only to $[\![\Gamma]\!]$ (with an empty context Δ) in the definitions of $[\![\phi]\!]_\Gamma$ and $[\![\phi]\!]^+_\Gamma$. Where the definition of $[\![\Gamma]\!]_\Delta$ appeals back to the interpretation of types, it does so only when this Γ was non-empty, and with an empty context given for the interpretation of the type. So $|\Gamma|$ has indeed decreased from one appeal to the interpretation of types to the next.

$$
\begin{array}{lcl}
a \in [\![\texttt{nat}]\!]_\Gamma & \Leftrightarrow & \top \\
a \in [\![\langle \texttt{vec}\ \phi\ l \rangle]\!]_\Gamma & \Leftrightarrow & (a \rightsquigarrow^* \texttt{nil} \Rightarrow l \sim_\Gamma 0) \land \\
& & \quad \forall a'.\forall a''.a \rightsquigarrow^* (\texttt{cons}\ a'\ a'') \Rightarrow \begin{array}[t]{l} (i)\ a' \in [\![\phi]\!]_\Gamma \land \exists l'. \\ (ii)\ a'' \in [\![\langle \texttt{vec}\ \phi\ l' \rangle]\!]_\Gamma \land \\ (iii)\ l \sim_\Gamma (S\ l') \end{array} \\
a \in [\![\Pi x : \phi'.\phi]\!]_\Gamma & \Leftrightarrow & \forall a' \in [\![\phi']\!]_\Gamma^+.\ (a\ a') \in [\![[a'/x]\phi]\!]_\Gamma \\
a \in [\![\forall x : \phi'.\phi]\!]_\Gamma & \Leftrightarrow & \forall a' \in [\![\phi']\!]_\Gamma^+.\ a \in [\![[a'/x]\phi]\!]_\Gamma \\
a \in [\![a_1 = a_2]\!]_\Gamma & \Leftrightarrow & (a \rightsquigarrow^* \texttt{join} \Rightarrow a_1 \sim_\Gamma a_2)
\end{array}
$$

<u>where</u>:

$$
\begin{array}{lcl}
a \sim_\Gamma a' & \Leftrightarrow & \forall \sigma.\ \sigma \in [\![\Gamma]\!] \Rightarrow (\sigma a) \downarrow (\sigma a') \\
a \in [\![\phi]\!]_\Gamma^+ & \Leftrightarrow & a \in [\![\phi]\!]_\Gamma \land (|\Gamma| > 0 \Rightarrow \forall \sigma \in [\![\Gamma]\!].\ \sigma a \in [\![\sigma\phi]\!])
\end{array}
$$

<u>and also</u>:

$$
\frac{}{\emptyset \in [\![\cdot]\!]_\Delta} \qquad \frac{a \in [\![\sigma\phi]\!]_\Delta^+ \quad \sigma \in [\![\Gamma]\!]_\Delta}{\sigma \cup \{(x,a)\} \in [\![\Gamma, x : \phi]\!]_\Delta}
$$

Figure 3: The interpretation $a \in [\![\phi]\!]_\Gamma$ of strongly normalizing terms with $\Gamma \vdash a : \phi$

3.3 Critical properties

A term is defined to be *neutral* iff it is of the form $(a\ a')$ or $(R_B\ a\ a'\ a'')$ (with $B \in \{\texttt{nat}, \texttt{vec}\}$), or if it is a variable. We prove three critical properties of reducibility at type ϕ, by mutual induction on $(|\Gamma|, d(\phi), l(a))$. Here we write $\mathit{next}(a) = \{a' \mid a \rightsquigarrow a'\}$.

R-Pres. $a \in [\![\phi]\!]_\Gamma \Rightarrow \mathit{next}(a) \subset [\![\phi]\!]_\Gamma$.
R-Prog. If a is neutral and $\Gamma \vdash a : \phi$, then $\mathit{next}(a) \subset [\![\phi]\!]_\Gamma \Rightarrow a \in [\![\phi]\!]_\Gamma$.
R-Join. Suppose $a_1 \sim_\Gamma a_2$; $\Gamma \vdash a' : a_1 = a_2$ for some a'; and $x \notin \mathit{dom}(\Gamma)$. Then $[\![[a_1/x]\phi]\!]_\Gamma \subset [\![[a_2/x]\phi]\!]_\Gamma$.

3.4 Soundness of typing with respect to the interpretation

Our typing rules are sound with respect to our interpretation of types (Figure 3). As usual, we must strengthen the statement of soundness for the induction to go through. We need a subcontext relationship, denoted $\Delta \subset \Gamma$, for Δ and Γ contexts: $\Delta \subset \Gamma \Leftrightarrow \forall x \in \mathit{dom}(\Delta).\ \Delta(x) = \Gamma(x)$.

Theorem 3 (Soundness for Interpretations)**.** *Suppose* $\Gamma \vdash a : \phi$. *Then for any* $\Delta\,Ok$ *with* $\Delta \subset \Gamma$ *and* $\sigma \in [\![\Gamma]\!]_\Delta$*, we have* $(\sigma a) \in [\![\sigma\phi]\!]_\Delta$.

Critically, we quantify over possibly open substitutions σ, whose ranges consist of closable terms.

Corollary 1 (Strong Normalization)**.** *If* $\Gamma \vdash a : \phi$*, then* $a \in SN$.

Corollary 2. *If* $\Gamma \vdash a : \phi$ *and* $\Gamma \vdash a' : \phi'$*, then* $a \downarrow a'$ *is decidable.*

Corollary 3 (Equational Soundness)**.** *If* $\cdot \vdash a : b_1 = b_2$*, then* $b_1 \downarrow b_2$.

Corollary 4 (Logical Soundness)**.** *There is a type* ϕ *such that* $\vdash a : \phi$ *does not hold for any* a.

Proof. By Equational Soundness, we do not have $\vdash a : 0 = (S\ 0)$ for any a.

$$
\begin{array}{lcllcl}
|x| & = & x & |(t\ t')| & = & (|t|\ |t'|) \\
|(t\ t')^-| & = & |t| & |\lambda x:\phi.t| & = & \lambda x.|t| \\
|\lambda^- x:\phi.t| & = & |t| & |0| & = & 0 \\
|(S\ t)| & = & (S\ |t|) & |(\mathtt{nil}\ \phi)| & = & \mathtt{nil} \\
|(\mathtt{cons}\ t\ t')| & = & (\mathtt{cons}\ |t|\ |t'|) & |(R_{\mathtt{nat}}\ x.\phi\ t\ t'\ t'')| & = & (R_{\mathtt{nat}}\ |t|\ |t'|\ |t''|) \\
|(R_{\mathtt{vec}}\ x.y.\phi\ t\ t'\ t'')| & = & (R_{\mathtt{vec}}\ |t|\ |t'|\ |t''|) & |(\mathtt{join}\ t\ t')| & = & \mathtt{join} \\
|(\mathtt{cast}\ x.\phi\ t\ t')| & = & |t'| & & &
\end{array}
$$

Figure 4: Translation from annotated terms to unannotated terms

4 Annotated $\mathtt{T}^{\mathtt{vec}}$

We now define a system of annotated terms t, and a decidable type computation system deriving judgments $\Gamma \Vdash t : \phi$, justified by dropping annotations via $|\cdot|$ (defined in Figure 4). The annotated terms t are the following. Annotations include types ϕ, possibly with designated free variables, as in $x.\phi$ (bound by the dot notation).

$$
\begin{array}{lll}
t & ::= & x \mid (t\ t') \mid (t\ t')^- \mid \lambda x:\phi.t \mid \lambda^- x:\phi.t \mid 0 \mid (S\ t) \mid (R_{\mathtt{nat}}\ x.\phi\ t\ t'\ t'') \\
& & \mid (\mathtt{nil}\ \phi) \mid (\mathtt{cons}\ t\ t') \mid (R_{\mathtt{vec}}\ x.y.\phi\ t\ t'\ t'') \mid (\mathtt{join}\ t\ t') \mid (\mathtt{cast}\ x.\phi\ t\ t')
\end{array}
$$

Three new constructs correspond to the typing rules `(spec-abs)`, `(spec-app)`, and `(conv)` of Figure 2: $\lambda^- x : \phi'.\phi$, $(t\ t')^-$ and $(\mathtt{cast}\ x.\phi\ t\ t')$. Figure 5 gives syntax-directed type-computation rules, which constitute a deterministic algorithm for computing a type ϕ as output from a context Γ and annotated term t as inputs. Several rules use the $|\cdot|$ function, since types ϕ (as defined in Section 2 above) may mention only unannotated terms.

Theorem 4 (Algorithmic Typing)**.** *Given Γ and a, we can, in an effective way, either find ϕ such that $\Gamma \Vdash a : \phi$, or else report that there is no such ϕ.*

This follows in a standard way from inspection of the rules, using Corollary 2 for the `join`-rule.

Theorem 5 (Soundness for Type Assignment)**.** *If $\Gamma \Vdash t : \phi$ then $\Gamma \vdash |t| : \phi$.*

4.1 Example

Now let us see versions of the examples mentioned in Section 1, available in the `guru-lang/lib/vec.g` library file for GURU (see `www.guru-lang.org`). The desired types for vector append ("*append*") and for associativity of vector append are:

$$
\begin{array}{lll}
\mathit{append} & : & \forall l_1 : \mathtt{nat}.\forall l_2 : \mathtt{nat}.\Pi v_1 : \langle \mathtt{vec}\ \phi\ l_1\rangle.\Pi v_2 : \langle \mathtt{vec}\ \phi\ l_2\rangle.\langle \mathtt{vec}\ \phi\ (\mathit{plus}\ l_1\ l_2)\rangle \\
\mathit{append\_assoc} & : & \forall l_1 : \mathtt{nat}.\forall l_2 : \mathtt{nat}.\forall l_3 : \mathtt{nat}. \\
& & \Pi v_1 : \langle \mathtt{vec}\ \phi\ l_1\rangle.\Pi v_2 : \langle \mathtt{vec}\ \phi\ l_2\rangle.\Pi v_3 : \langle \mathtt{vec}\ \phi\ l_3\rangle. \\
& & \quad (\mathit{append}\ (\mathit{append}\ v_1\ v_2)\ v_3) = (\mathit{append}\ v_1\ (\mathit{append}\ v_2\ v_3))
\end{array}
$$

We consider now annotated inhabitants of these types. The first is the following:

$$
\begin{array}{lll}
\mathit{append} & = & \lambda^- l_1 : \mathtt{nat}.\lambda^- l_2 : \mathtt{nat}.\lambda v_1 : \langle \mathtt{vec}\ \phi\ l_1\rangle.\lambda v_2 : \langle \mathtt{vec}\ \phi\ l_2\rangle. \\
& & \quad (R_{\mathtt{vec}}\ (x.y.\langle \mathtt{vec}\ \phi\ (\mathit{plus}\ x\ l_2)\rangle) \\
& & \qquad (\mathtt{cast}\ (x.\langle \mathtt{vec}\ \phi\ x\rangle)\ P_1\ v_2) \\
& & \qquad (\lambda^- l : \mathtt{nat}.\lambda x : \phi.\lambda v_1' : \langle \mathtt{vec}\ \phi\ l\rangle.\lambda r : \langle \mathtt{vec}\ \phi\ (\mathit{plus}\ l\ l_2)\rangle). \\
& & \qquad\quad (\mathtt{cast}\ (x.\langle \mathtt{vec}\ \phi\ x\rangle)\ P_2\ (\mathtt{cons}\ x\ r)) \\
& & \qquad v_1)
\end{array}
$$

$$\frac{\Gamma \Vdash t : \phi \quad \Gamma \Vdash t' : \phi' \quad |t| \downarrow |t'|}{\Gamma \Vdash (\texttt{join}\ t\ t') : |t| = |t'|} \quad \frac{\Gamma \Vdash t : a = a' \quad \Gamma \Vdash t' : [a/x]\phi}{\Gamma \Vdash (\texttt{cast}\ x.\phi\ t\ t') : [a'/x]\phi} \quad \frac{\Gamma, x : \phi' \Vdash t : \phi \quad x \notin FV(|t|)}{\Gamma \Vdash \lambda^{-} x : \phi'.t : \forall x : \phi'.\phi}$$

$$\frac{\Gamma \Vdash t : \forall x : \phi'.\phi \quad \Gamma \Vdash t' : \phi'}{\Gamma \Vdash (t\ t')^{-} : [|t'|/x]\phi} \quad \frac{\Gamma, x : \phi' \Vdash t : \phi}{\Gamma \Vdash \lambda x : \phi'.t : \Pi x : \phi'.\phi} \quad \frac{\Gamma \Vdash t : \Pi x : \phi'.\phi \quad \Gamma \Vdash t' : \phi'}{\Gamma \Vdash (t\ t') : [|t'|/x]\phi}$$

$$\frac{\begin{array}{l} \Gamma \Vdash t'' : \langle \texttt{vec}\ \phi'\ l \rangle \\ \Gamma \Vdash t : [0/x, \texttt{nil}/y]\phi \\ \Gamma \Vdash t' : \forall l : \texttt{nat}.\Pi z : \phi'.\Pi v : \langle \texttt{vec}\ \phi'\ l \rangle.\Pi u : [l/x, v/y]\phi. \\ \qquad\quad [(S\ l)/x, (\texttt{cons}\ z\ v)/y]\phi \end{array}}{\Gamma \Vdash (R_{\texttt{vec}}\ x.y.\phi\ t\ t'\ t'') : [l/x, |t''|/y]\phi}$$

Figure 5: Type-computation system for annotated $\texttt{T}^{\texttt{vec}}$ (selected rules)

The two cases in the $R_{\texttt{vec}}$ term return a type-cast version of what would standardly be returned in an unannotated version of *append*. The proofs P_1 and P_2 used in those casts show respectively that $l_2 = (plus\ 0\ l_2)$ and $(S\ (plus\ l\ l_2)) = (plus\ (S\ l)\ l_2)$. They are simple join-proofs:

$$P_1 \;=\; (\texttt{join}\ l_2\ (plus\ 0\ l_2)) \qquad\qquad P_2 \;=\; (\texttt{join}\ (S\ (plus\ l\ l_2))\ (plus\ (S\ l)\ l_2))$$

Now for *append_assoc*, we can use the following annotated term:

$$\begin{array}{rcl} append\_assoc & = & \lambda^{-} l_1 : \texttt{nat}.\lambda^{-} l_2 : \texttt{nat}.\lambda^{-} l_3 : \texttt{nat}. \\ & & \lambda v_1 : \langle \texttt{vec}\ \phi\ l_1 \rangle.\lambda v_2 : \langle \texttt{vec}\ \phi\ l_2 \rangle.\lambda v_3 : \langle \texttt{vec}\ \phi\ l_3 \rangle. \\ & & \quad (R_{\texttt{vec}}\ (x.y.(append\ (append\ v_1\ v_2)\ v_3) = (append\ v_1\ (append\ v_2\ v_3))) \\ & & \qquad (\texttt{join}\ (append\ (append\ \texttt{nil}\ v_2)\ v_3) = (append\ \texttt{nil}\ (append\ v_2\ v_3))) \\ & & \qquad (\lambda^{-} l : \texttt{nat}.\lambda x : \phi.\lambda v_1' : \langle \texttt{vec}\ \phi\ l \rangle. \\ & & \qquad\ \ \lambda r : (append\ (append\ v_1'\ v_2)\ v_3) = (append\ v_1'\ (append\ v_2\ v_3)). \\ & & \qquad\quad P_3)) \end{array}$$

The omitted proof P_3 is an easy equational proof of the following type:

$$(append\ (append\ (\texttt{cons}\ x\ v_1')\ v_2)\ v_3) = (append\ (\texttt{cons}\ x\ v_1')\ (append\ v_2\ v_3))$$

5 $\texttt{T}^{\texttt{vec}}$ with Large Eliminations

Next we study an extended version of $\texttt{T}^{\texttt{vec}}$ with large eliminations, i.e. types defined by pattern matching on terms. This extended language no longer is normalizing under general β-reduction $\leadsto$, but we will prove that well-typed closed terms normalize under call-by-value evaluation $\leadsto_v$. In particular, the language is type safe and logically consistent.

The additions to the language and type system are shown in in figure 6. The type language is extended with type variables α, and a recursion form which is introduced and eliminated by the `fold` and `unfold` rules. While type conversion and type folding/unfolding are completely implicit, we replace the `spec-abs/app` rules with new rules `spec-abs'/app'` which require the place where we introduce or eliminate the $\forall$-type to be marked by new *quasi-implicit* forms $(\lambda.a)$ and (a). These forms do not mention the quantified variable or the term it is instantiated with, so we retain the advantages of specificational reasoning. The point of these forms is their evaluation behavior: $((\lambda.a)) \leadsto_v a$, and

$$\phi ::= \ldots \mid \alpha \mid R\ a\ \phi\ (\alpha.\phi') \qquad a ::= \ldots \mid (\lambda.a) \mid (a) \qquad v ::= \ldots \mid (\lambda.a)$$

$$\frac{\Gamma, x:\phi' \vdash a:\phi \quad x \notin FV(a)}{\Gamma \vdash (\lambda.a) : \forall x:\phi'.\phi}\ \texttt{spec-abs}' \qquad \frac{\Gamma \vdash a : \forall x:\phi'.\phi \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash (a) : [a'/x]\phi}\ \texttt{spec-app}'$$

$$\frac{\Gamma \vdash a:\phi}{\Gamma \vdash a : R\ 0\ \phi\ (\alpha.\phi')}\ \texttt{foldZ} \qquad \frac{\Gamma \vdash a : R\ 0\ \phi\ (\alpha.\phi')}{\Gamma \vdash a:\phi}\ \texttt{unfoldZ}$$

$$\frac{\Gamma \vdash a : [R\ a'\phi\ (\alpha.\phi')/\alpha]\phi' \quad \Gamma \vdash a' : \texttt{nat}}{\Gamma \vdash a : R\ (S\ a')\ \phi\ (\alpha.\phi')}\ \texttt{foldS} \qquad \frac{\Gamma \vdash a : R\ (S\ a')\ \phi\ (\alpha.\phi') \quad \Gamma \vdash a' : \texttt{nat}}{\Gamma \vdash a : [R\ a'\ \phi\ (\alpha.\phi')/\alpha]\phi'}\ \texttt{unfoldS}$$

Figure 6: Types, terms, values, and typing rules for $\texttt{T}^{\texttt{vec}}$ with large eliminations.

$(\lambda.a)$ counts as a value so CBV evaluation will never reduce inside it. Besides this, the CBV operational semantics is standard, so we omit it here.

In the language with large eliminations we no longer have normalization or type safety for arbitrary open terms. This is because the richer type system lets us make use of absurd equalities: whenever we have $\Gamma \vdash a : \phi$ and $\Gamma \vdash p : (S\ a') = 0$, we can show $\Gamma \vdash a : \phi'$ for any ϕ' by going via the intermediate type $(R\ 0\ \phi(\alpha.\phi'))$. In particular, this means we can show judgements like

$$p : 1 = 0 \vdash (\lambda x.x\ x)\ (\lambda x.x\ x) : \texttt{nat} \qquad \text{and} \qquad p : 1 = 0 \vdash 0\ 0 : \texttt{nat}.$$

This is also the reason we introduce the quasi-implicit products. Using our old rule `spec-abs` we would be able to show $\vdash 0\ 0 : \forall p : 1 = 0.\texttt{nat}$, despite $0\ 0$ being a stuck term in our operational semantics.

Because of this *quod libet* property it is no longer convenient to prove Progress and Preservation before Normalization. While the proof of Preservation is not hard, Progress as we have seen depends on the logical consistency of the language, which is exactly what we hope to establish through Normalization. To cut this circle we design an interpretation of types that lets us prove type safety, Canonical forms and Normalization in a single induction. This interpretation (figure 8) has several interesting features.

5.1 Semantics of Equality

We need to pick an interpretation for equality types. Since we are only interested in closed terms, this can be less elaborate than in section 3. Perhaps surprisingly, even though we are interested in CBV-evaluation of programs, we can still interpret equality as joinability $\downarrow$ under unrestricted β-reduction. In the interpretation we use $\rightsquigarrow_v$ for the program being evaluated, but $\rightsquigarrow$ whenever we talk about terms occuring in types (namely in `vec`, $=$, and R-types). The `join` typing rule is specified in terms of $\rightsquigarrow$, so when doing symbolic evaluation of programs at typechecking time the typechecker can use unrestricted reduction, which gives a powerful type system than can prove many equalities.

5.2 Unfolding of R-Types

Since we are only interested in closed terms, we can interpret R-types by simply expanding them out – we need not worry about types like $(R\ x\ \phi\ (\alpha.\phi'))$ which is stuck on a variable. There is however a small subtlety: since we interpret R-types by unfolding them, in order to prove that the interpretation is well-defined, we need to show that we will not encounter an infinite sequence of unfoldings. In other words, before we can show normalization of terms, we need to show normalization of type unfolding.

$$\frac{\begin{array}{c} b \rightsquigarrow^* 0 \\ \phi_1 \longmapsto \phi_1' \end{array}}{R\ b\ \phi_1\ (\alpha.\phi_2) \longmapsto \phi_1'}\ \texttt{rwrR}-\texttt{beta1} \qquad \frac{\begin{array}{c} b \rightsquigarrow^* (S\ n) \\ \phi_1 \longmapsto \phi_1' \\ \phi_2 \longmapsto \phi_2' \end{array}}{R\ b\ \phi_1\ (\alpha.\phi_2) \longmapsto [R\ n\ \phi_1'\ (\alpha.\phi_2')/\alpha]\phi_2'}\ \texttt{rwrR}-\texttt{beta2}$$

Figure 7: Type rewriting $\phi \longmapsto \phi'$ (excerpt: congruence and reflexivity rules omitted)

Define $[\![\phi]\!] = [\![\triangledown\phi]\!]$ if $\phi \longmapsto \phi'$ for some ϕ'. Otherwise, define

$$\begin{array}{lll}
a \in [\![\texttt{nat}]\!] & \Leftrightarrow & \exists n. a \rightsquigarrow_v^* n \\
a \in [\![\langle \texttt{vec}\ \phi\ l\rangle]\!] & \Leftrightarrow & (a \rightsquigarrow_v^* \texttt{nil} \wedge l \rightsquigarrow^* 0) \vee \\
 & & \quad \exists v\ v'\ n. \quad a \rightsquigarrow_v^* (\texttt{cons}\ v\ v') \wedge l \rightsquigarrow^* (S\ n) \\
 & & \qquad\qquad\qquad \wedge\ v \in [\![\phi]\!]\ \wedge\ v' \in [\![\langle \texttt{vec}\ \phi\ n\rangle]\!] \\
a \in [\![\Pi x:\phi'.\phi]\!] & \Leftrightarrow & \exists a'. a \rightsquigarrow_v^* (\lambda x.a')\ \wedge\ \forall a' \in [\![\phi']\!].\ (a\ a') \in [\![[a'/x]\phi]\!] \\
a \in [\![\forall x:\phi'.\phi]\!] & \Leftrightarrow & \exists a'. a \rightsquigarrow_v^* (\lambda a')\ \wedge\ \forall a' \in [\![\phi']\!].\ (a) \in [\![[a'/x]\phi]\!] \\
a \in [\![a_1 = a_2]\!] & \Leftrightarrow & a \rightsquigarrow_v^* \texttt{join}\ \wedge\ a_1 \downarrow a_2 \\
a \in [\![R\ b\ \phi\ (\alpha.\phi')]\!] & \Leftrightarrow & \text{False}
\end{array}$$

$$\frac{}{\emptyset \in [\![\cdot]\!]} \qquad \frac{v \in [\![\sigma\phi]\!] \quad \sigma \in [\![\Gamma]\!]}{\sigma \cup \{(x,v)\} \in [\![\Gamma, x:\phi]\!]}$$

Figure 8: Type interpretation $a \in [\![\phi]\!]$ and context interpretation $\sigma \in [\![\Gamma]\!]$ for $\texttt{T}^{\texttt{vec}}$ with large eliminations

For our type language, this is not an onerous requirement. Furthermore, we design the interpretation to only require proof of weak normalization and confluence, rather than strong normalization. We proceed by defining a rewrite-relation $\longmapsto$ on types, with the β-rules given in figure 7. It is routine[10] to show that $\longmapsto^*$ is confluent, and it also easy to see that there exists a rewriting strategy which always terminates (rewrite innermost redexes first). Together, these two facts prove uniqueness of normal forms.

Lemma 1. *Every type ϕ has a unique $\longmapsto$-normal form, which we will denote $\triangledown\phi$.*

Lemma 2 (Properties of type unfolding)**.**

- $\triangledown\langle \texttt{vec}\ \phi\ a\rangle = \langle \texttt{vec}\ \phi\ \triangledown a\rangle$, $\triangledown(\forall x:\phi'.\phi) = \forall x:\triangledown\phi'.\triangledown\phi$, *and* $\triangledown(\Pi x:\phi'.\phi) = \Pi x:\triangledown\phi'.\triangledown\phi$.
- *For all σ, ϕ, we have $\triangledown\sigma\triangledown\phi = \triangledown\sigma\phi$.*
- *If $a_1 \downarrow a_2$, then for any ϕ there exists a ψ such that $\triangledown[a_1/x]\phi = [a_1/x]\psi$ and $\triangledown[a_2/x]\phi = [a_2/x]\psi$.*

5.3 Normalization to Canonical Form

Now we can define the interpretation $[\![\ ]\!]$ as in figure 8. The way we define it means that $[\![\phi]\!] = [\![\triangledown\phi]\!]$. This allows us to prove the following lemma.

Lemma 3. *For all types ϕ (not just types such that $\phi \not\longmapsto$), if ϕ is not an R-type then the equivalences in figure 8 hold.*

With this lemma in hand, the proof proceeds much like the proof for open terms in section 3:
R-Canon. If $a \in [\![\phi]\!]$, then $a \rightsquigarrow_v^* v$ for some v. Furthermore, if the top-level constructor of ϕ is $\texttt{nat}$, Π, $\forall$, $=$, or $\texttt{vec}$, then v is the corresponding introduction form.
R-Pres. If $a \in [\![\phi]\!]$ and $a \rightsquigarrow_v a'$, then $a' \in [\![\phi]\!]$.
R-Prog. If $a \rightsquigarrow_v a'$, and $a' \in [\![\phi]\!]$, then $a \in [\![\phi]\!]$.
R-Join. If $a_1 \downarrow a_2$, then $a \in [\![[a_1/x]\phi]\!]$ implies $a \in [\![[a_2/x]\phi]\!]$.

Theorem 6. *If* $\Gamma \vdash a : \phi$ *and* $\sigma \in [\![\Gamma]\!]$*, then* $\sigma a \in [\![\sigma\phi]\!]$.

Corollary 5 (Type Safety)**.** *If* $\vdash a : \phi$*, then* $a \leadsto_v^* v$.

Corollary 6 (Logical Soundness)**.** $\vdash a : 1{=}0$ *does not hold for any a.*

6 Conclusion and Future Work

The $\mathtt{T}^{\mathtt{vec}}$ type theory includes intersection types and a form of equality reflection, justified by translation to an undecidable unannotated system. The division into annotated and unannotated systems enables us to reason about terms without annotations, while retaining decidable type checking. We have seen how this approach extends to a language including large eliminations, by introducing a novel kind of *quasi-implicit* products. The quasi-implicit products allow convenient reasoning about specificational data, while permitting a simple proof of normalization of closed terms. Possible future work includes formalizing the metatheory, and extending to a polymorphic type theory. Adding an extensional form of equality while retaining decidability would also be of interest, as in [1].

Acknowledgments: Thanks to members of the TRELLYS team, especially Stephanie Weirich and Tim Sheard, for discussions on this and related systems. This work was partially supported by the the U.S. National Science Foundation under grants 0910510 and 0910786.

References

[1] T. Altenkirch, C. McBride, and W. Swierstra. Observational Equality, Now! In A. Stump and H. Xi, editors, *PLPV '07: Proceedings of the 2007 Workshop on Programming Languages meets Program Verification*, pages 57–68, 2007.

[2] B. Barras and B. Bernardo. The Implicit Calculus of Constructions as a Programming Language with Dependent Types. In Roberto M. Amadio, editor, *Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008*, volume 4962 of *Lecture Notes in Computer Science*, pages 365–379. Springer, 2008.

[3] R. Constable and the PRL group. *Implementing mathematics with the Nuprl proof development system*. Prentice-Hall, 1986.

[4] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, 1979.

[5] R. Harper, F. Honsell, and G. Plotkin. A Framework for Defining Logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993.

[6] P. Martin-Löf. *Intuitionistic type theory*. Bibliopolis, 1984.

[7] C. McBride. *Dependently Typed Functional Programs and Their Proofs*. PhD thesis, 1999.

[8] A. Miquel. The Implicit Calculus of Constructions. In *Typed Lambda Calculi and Applications*, pages 344–359, 2001.

[9] N. Mishra-Linger and T. Sheard. Erasure and Polymorphism in Pure Type Systems. In Roberto M. Amadio, editor, *Foundations of Software Science and Computational Structures, 11th International Conference (FOSSACS)*, pages 350–364. Springer, 2008.

[10] A. Stump, M. Deters, A. Petcher, T. Schiller, and T. Simpson. Verified Programming in Guru. In T. Altenkirch and T. Millstein, editors, *Programming Languges meets Program Verification (PLPV)*, 2009.

[11] Masako Takahashi. Parallel reductions in λ-calculus. *Inf. Comput.*, 118(1):120–127, 1995.

A Proof of Type Preservation (Theorem 1)

More about contexts. In more detail, we consider a contexts Γ to be a function from a finite set of variables to types, together with a total ordering on its domain, and write $\Gamma, x : \phi$ for the function that behaves just like Γ, except that it returns ϕ for x, and places x after the variables in $dom(\Gamma)$. When $\Gamma \equiv \Gamma_1 \cup \Gamma_2$ with all the variables in Γ_2 greater than those of Γ_1 in the ordering, we write Γ_1, Γ_2 (implying also that the domains of Γ_1 and Γ_2 are disjoint).

The proof of Type Preservation is by induction on the structure of the assumed typing derivation. We list all cases. Unless we introduce meta-variable b for another purpose, in each case we will assume the term in question reduces to b. In cases where the term in question is a normal form, this will lead to a contradiction.

Case:

$$\frac{\Gamma(x) \equiv \phi}{\Gamma \vdash x : \phi}$$

This case cannot arise, since x is a normal form and so cannot reduce.

Case:

$$\frac{a \downarrow a'}{\Gamma \vdash \texttt{join} : a = a'}$$

This case cannot arise, since `join` is a normal form.

Case:

$$\frac{\Gamma \vdash a''' : a' = a'' \quad \Gamma \vdash a : [a'/x]\phi \quad x \notin dom(\Gamma)}{\Gamma \vdash a : [a''/x]\phi}$$

(Recall that by convention in this proof, our second assumption is $a \rightsquigarrow b$.) By the induction hypothesis, we have $\Gamma \vdash b : [a'/x]\phi$. We may then reapply this rule to conclude $\Gamma \vdash b : [a''/x]\phi$.

Case:

$$\frac{\Gamma, x : \phi' \vdash a : \phi \quad x \notin FV(a)}{\Gamma \vdash a : \forall x : \phi'.\phi}$$

By the induction hypothesis, we have $\Gamma, x : \phi' \vdash b : \phi$. Reduction cannot increase the set of free variables, so $x \notin FV(b)$. We may then reapply this rule to obtain $\Gamma \vdash b : \forall x : \phi'.\phi$.

Case:

$$\frac{\Gamma \vdash a : \forall x : \phi'.\phi \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash a : [a'/x]\phi}$$

By the induction hypothesis, we have $\Gamma \vdash b : \forall x : \phi'.\phi$. We may then reapply this rule to obtain $\Gamma \vdash b : [a'/x]\phi$.

Case:

$$\frac{\Gamma, x : \phi' \vdash a : \phi}{\Gamma \vdash \lambda x.a : \Pi x : \phi'.\phi}$$

By the induction hypothesis, we have $\Gamma, x : \phi' \vdash b : \phi$. We may then reapply this rule to obtain $\Gamma \vdash \lambda x.b : \Pi x : \phi'.\phi$.

Case:

$$\frac{\Gamma \vdash a : \Pi x : \phi'.\phi \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash (a\ a') : [a'/x]\phi}$$

Suppose the reduction is from $a \rightsquigarrow b$, so we have $(a\ a') \rightsquigarrow (b\ a')$. Then we apply the induction hypothesis to the first premise to obtain $\Gamma \vdash b : \Pi x : \phi'.\phi$, and then reapply this rule to obtain $\Gamma \vdash (b\ a') : [a'/x]\phi$. Suppose now that the reduction is from $a' \rightsquigarrow b'$, so we have $(a\ a') \rightsquigarrow (a\ b')$. Then we apply the induction hypothesis to the second premise to obtain $\Gamma \vdash b' : \phi'$. Reapplying this rule then gives us $\Gamma \vdash (a\ b') : [b'/x]\phi$. We must now apply the `conv` rule (of Figure 2), using as the first premise the judgment $\Gamma \vdash \mathtt{join} : a' = b'$, which is derivable since $a' \downarrow b'$ (because $a' \rightsquigarrow b'$). This gives us the desired result: $\Gamma \vdash (a\ b') : [a'/x]\phi$.

Finally, suppose the reduction is because we have $a \equiv \lambda x.b$ for some x and b, and the application itself is being β-reduced. In this case, we need a lemma in order to limit the cases arising from inversion on the derivation of $\Gamma \vdash \lambda x.b : \Pi x : \phi'.\phi$. We now need this lemma (proof in Section A.1):

Lemma 4 (Simplifying Inversion)**.** *Suppose $\Gamma \vdash a : \phi$ is derivable, where a is an introduction form (i.e., of the form `join`, 0, $(S\ b)$, `nil`, $(\mathtt{cons}\ b\ b')$, or $\lambda x.b$), and ϕ has the corresponding form of type (e.g., a Π-type for a λ-abstraction). Then $\Gamma \vdash a : \phi$ is also derivable by a derivation starting with the corresponding introduction rule for the form of a, using the same context Γ, and followed by a sequence of (`conv`) inferences.*

Using Simplifying Inversion on the derivation $\Gamma \vdash \lambda x.b : \Pi x : \phi'.\phi$, we may assume this derivation starts like this:

$$\frac{\Gamma, x : \psi' \vdash b : \psi}{\Gamma \vdash \lambda x.b : \Pi x : \psi'.\psi}$$

The derivation then uses a sequence S of (`conv`) inferences to end in $\Gamma \vdash \lambda x.b : \Pi x : \phi'.\phi$. Let S^{-1} be the sequence which is just the same except that for every first premise $\Gamma \vdash d : c = c'$ of a (`conv`)-inference in S, we have a (`conv`) inference with first premise $\Gamma \vdash \mathtt{join} : c' = c$, easily derived from $\Gamma \vdash d : c = c'$. We now wish to show that the result of substituting our a' for x in b has the expected type $[a'/x]\phi$. For this, we must first apply the sequence S^{-1} to $\Gamma \vdash a' : \phi'$. This gives us $\Gamma \vdash a' : \psi'$. Now we apply Substitution (proved in Section A.2 below):

Lemma 5 (Substitution)**.** *If $\Gamma, x : \phi, \Gamma' \vdash a' : \phi'$ and $\Gamma \vdash a : \phi$, then $\Gamma, [a/x]\Gamma' \vdash [a/x]a' : [a/x]\phi'$.*

This gives us $\Gamma \vdash [a/x]a' : [a/x]\psi$. We may apply S now to obtain $\Gamma \vdash [a/x]a' : [a/x]\phi$.

Case:

$$\frac{}{\Gamma \vdash 0 : \mathtt{nat}}$$

This case cannot arise since 0 is a normal form.

Case:

$$\frac{}{\Gamma \vdash \mathtt{nil} : \langle \mathtt{vec}\ \phi\ 0 \rangle}$$

This case cannot arise since $\mathtt{nil}$ is a normal form.

Case:

$$\frac{\Gamma \vdash a : \mathtt{nat}}{\Gamma \vdash (S\ a) : \mathtt{nat}}$$

By the induction hypothesis, we have $\Gamma \vdash b : \mathtt{nat}$, and we may then reapply this rule to obtain $\Gamma \vdash (S\ b) : \mathtt{nat}$.

Case:

$$\frac{\begin{array}{l} \Gamma \vdash a'' : \mathtt{nat} \\ \Gamma \vdash a : [0/x]\phi \\ \Gamma \vdash a' : \Pi y : \mathtt{nat}.\Pi u : [y/x]\phi.[(Sy)/x]\phi \end{array}}{\Gamma \vdash (R_{\mathtt{nat}}\ a\ a'\ a'') : [a''/x]\phi}$$

If the reduction arises from $a \rightsquigarrow b$ or $a' \rightsquigarrow b$ or $a'' \rightsquigarrow b$, then we apply the induction hypothesis to the corresponding premise and then reapply this typing rule. If the reduction arises because $a'' \equiv 0$ and the $R_{\mathtt{nat}}$-term is itself being reduced to a, then we have $\Gamma \vdash a : [a''/x]\phi$ from the second premise to the rule, and the fact that $a'' \equiv 0$. Suppose the reduction arises because $a'' \equiv (S\ b)$ and the $R_{\mathtt{nat}}$-term is itself being reduced to $(a'\ b\ (R_{\mathtt{nat}}\ a\ a'\ b))$. Applying Simplifying Inversion (Lemma 4), we obtain $\Gamma \vdash b : \mathtt{nat}$. So we may apply the $R_{\mathtt{nat}}$ typing rule to obtain $\Gamma \vdash (R_{\mathtt{nat}}\ a\ a'\ b) : [b/x]\phi$. Applying the application typing rule twice gives us then $\Gamma \vdash (a'\ b\ (R_{\mathtt{nat}}\ a\ a'\ b)) : [(S\ b)/x]\phi$. This is the desired typing, since $a'' \equiv (S\ b)$.

Case:

$$\frac{\begin{array}{l} \Gamma \vdash a : \phi \\ \Gamma \vdash a' : \langle \mathtt{vec}\ \phi\ l \rangle \end{array}}{\Gamma \vdash (\mathtt{cons}\ a\ a') : \langle \mathtt{vec}\ \phi\ (S\ l) \rangle}$$

The reduction must arise from $a \rightsquigarrow b$ or $a' \rightsquigarrow b$, so we apply the induction hypothesis to the corresponding premise, and then reapply this typing rule.

Case:

$$\frac{\begin{array}{l} \Gamma \vdash a'' : \langle \mathtt{vec}\ \phi'\ l \rangle \\ \Gamma \vdash a : [0/y, \mathtt{nil}/x]\phi \\ \Gamma \vdash a' : \Pi z : \phi'.\forall l : \mathtt{nat}.\Pi v : \langle \mathtt{vec}\ \phi'\ l \rangle.\Pi u : [l/y, v/x]\phi. \\ \qquad [(S\ l)/y, (\mathtt{cons}\ z\ v)/x]\phi \end{array}}{\Gamma \vdash (R_{\mathtt{vec}}\ a\ a'\ a'') : [l/y, a''/x]\phi}$$

If the reduction arises from $a \rightsquigarrow b$ or $a' \rightsquigarrow b$ or $a'' \rightsquigarrow b$, then we apply the induction hypothesis to the corresponding premise and then reapply this typing rule. If the reduction arises because $a'' \equiv \texttt{nil}$ and the $R_{\texttt{nat}}$-term is itself being reduced to a, then we have $\Gamma \vdash a : [0/y, a''/x]\phi$ from the second premise to the rule, and the fact that $a'' \equiv \texttt{nil}$. By Simplifying Inversion (Lemma 4), we know there is a derivation of $\Gamma \vdash \texttt{nil} : \langle \texttt{vec}\ \phi'\ l \rangle$ which starts with $\Gamma \vdash \texttt{nil} : \langle \texttt{vec}\ \phi''\ 0 \rangle$ and then has a sequence S of (conv)-inferences. We may use this same series S to change 0 to l in $\Gamma \vdash a : [0/y, a''/x]\phi$, yielding the desired conclusion. Finally, suppose the reduction arises because $a'' \equiv (\texttt{cons}\ b'\ b'')$ for some b' and b'', and the $R_{\texttt{vec}}$-term itself is reduced to $(a'\ b'\ b''\ (R_{\texttt{vec}}\ a\ a'\ b''))$. By Simplifying Inversion again, we have a derivation of $\Gamma \vdash (\texttt{cons}\ b'\ b'') : \langle \texttt{vec}\ \phi'\ l \rangle$ starting from a cons-introduction deriving $\Gamma \vdash (\texttt{cons}\ b'\ b'') : \langle \texttt{vec}\ \phi''\ (S\ l'') \rangle$ from premises $\Gamma b' : \phi''$ and $\Gamma b'' : \langle \texttt{vec}\ \phi''\ l'' \rangle$; and then using a sequence S of (conv) inferences.

We may apply the sequence S of (conv) inferences to the typing for b'' to obtain $\Gamma b'' : \langle \texttt{vec}\ \phi'\ \hat{l} rangle$, for some $\hat{l}$ where $(S\ \hat{l}) \equiv l$. With this, we can reapply the typing rule for $R_{\texttt{vec}}$ to get $\Gamma \vdash (R_{\texttt{vec}}\ a\ a'\ b'') :$ $[\hat{l}/y, b''/x]\phi$. Using this and the typing for b'' we derived just previously, we can obtain $\Gamma \vdash (a'\ b'\ b''\ (R_{\texttt{vec}}\ a\ a'\ b'')) :$ $[(S\ \hat{l})/y, (\texttt{cons}\ b'\ b'')/x]\phi$. Since $a \equiv (\texttt{cons}\ b'\ b'')$ and $(S\ \hat{l}) \equiv l$, the type is equivalent to the desired one.

A.1 Proof of Simplifying Inversion (Lemma 4)

For the proof of this lemma, we begin by simplifying the derivation $\mathscr{D}$ of $\Gamma \vdash a : \phi$ by applying two transformations to the maximal path ending with the conclusion $\mathscr{D}$, which is assigning a type to a (rather than a strict subterm of a). First, we remove all inferences of the following form:

$$\dfrac{\dfrac{\Gamma, x : \phi' \vdash a : \phi}{\Gamma \vdash a : \forall x : \phi'.\phi} \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash a : [a'/x]\phi}$$

We may replace this with the result of applying the Substitution Lemma proved in the next section, in the special case where $x \notin FV(a)$. By inspection of the proof of Substitution, we see that while inferences of the form we are eliminating can be created, they must be created in a part of $\mathscr{D}$ typing a strict subterm of a. This is because a is an introduction form.

The second transformation simplifies this inference:

$$\dfrac{\dfrac{\Gamma \vdash \hat{a} : a_1 = a_2 \quad \Gamma \vdash a : \forall y : [a_1/x]\phi'.[a_1/x]\phi}{\Gamma \vdash a : \forall y : [a_2/x]\phi'.[a_2/x]\phi} \quad \Gamma \vdash a' : [a_2/x]\phi'}{\Gamma \vdash a : [a'/y][a_2/x]\phi}$$

Observing that the substitutions in question commute, we reduce this to the following, where $\hat{a}'$ is easily constructed to show $a_2 = a_1$ from $\hat{a} : a_1 = a_2$:

$$\dfrac{\Gamma \vdash \hat{a} : a_1 = a_2 \quad \dfrac{\Gamma \vdash a : \forall y : [a_1/x]\phi'.[a_1/x]\phi \quad \dfrac{\Gamma \vdash \hat{a}' : a_2 = a_1 \quad \Gamma \vdash a' : [a_2/x]\phi'}{\Gamma \vdash a' : [a_1/x]\phi'}}{\Gamma \vdash a : [a_1/x][a'/y]\phi}}{\Gamma \vdash a : [a_2/x][a'/y]\phi}$$

For each of these transformations, the following measure is strictly decreased in the lexicographic ordering (combining two copies of the natural number ordering): the pair of the sum of the distances of occurrences of a (conv) inference on the maximal path typing a; and the number of inferences of the form removed by the first transformation in that same path. Note that the `(conv)` inference introduced by the second transformation in the topmost rightmost position show above is not on the maximal path typing a (it is typing a'). Strict decrease of the stated measure implies that the transformations terminate. They also preserve the form of the derived judgment.

We can now prove the lemma by induction on the simplified derivation $\mathscr{D}$. It cannot end in a use of `(spec-abs)`, since then ϕ would be a $\forall$-type (and by assumption it is of the form corresonding to the introduction form which a is assumed to have). It also cannot be a `(spec-app)` inference, for the following reason. Consider the maximal consecutive sequence S of `(spec-app)` inferences ending at the conclusion of $\mathscr{D}$, and typing a. These inferences cannot start with the conclusion of either a `(conv)` or a `(spec-abs)` inference, since such patterns of inference have been eliminated by the above transformations. But these are the only possibilities, since a is an introduction form. Therefore, the derivation $\mathscr{D}$ ends in a sequence of `(conv)` inferences, starting from a use of the introduction rule for a. Since `(conv)` does not change the context, this introduction inference for a uses the same context, as required by the statement of the lemma.

A.2 Proof of Substitution (Lemma 5)

The proof is written using different variable names than the statement of the lemma, in order to not clash with the variable names in the typing rules. We prove:

If $\Gamma, y : \psi, \Gamma' \vdash a : \phi$ and $\Gamma \vdash b : \psi$, then $\Gamma, [b/y]\Gamma' \vdash [b/y]a : [b/y]\phi$.

The proof is by induction on the depth of $\Gamma, y : \psi, \Gamma' \vdash a : \phi$. The cases are:

Case:

$$\frac{(\Gamma, y : \psi, \Gamma')(x) \equiv \phi \quad \Gamma Ok}{\Gamma, y : \psi, \Gamma' \vdash x : \phi}$$

There are three cases: $x \in dom(\Gamma)$, $x \in dom(\Gamma')$, or $x = y$. If $x \in dom(\Gamma)$, then by ΓOk we know $y \notin FV(\phi)$, so $[b/y]\phi \equiv \phi$ and the conclusion follows by Var. If $x \in dom(\Gamma')$ the conclusion follows directly by Var. In the case $x = y$, we use the second assumption together with a weakening lemma:

Lemma 6 (Weakening)**.** *If $\Gamma \vdash a : \phi$, $dom(\Gamma) \subset dom(\Gamma')$, and $\Gamma' Ok$, then $\Gamma, \Gamma' \vdash a : \phi$.*

Proof. Induction on $\Gamma \vdash a : \phi$. The only interesting cases are for Abs and Spec-Abs. There we have $\Gamma, x : \phi' \vdash a : \phi$ by assumption; by regularity, we get $\Gamma, x : \phi' Ok$, so then $\Gamma', x : \phi' Ok$ and the conclusion follows by IH. □

Lemma 7 (Free variables of typable terms)**.** *If $\Gamma \vdash a : \phi$, then $FV(a) \subset dom(\Gamma)$.*

The proof is a straightforward induction on the typing derivation.

We also need to note that by Lemma 7 $FV(b) \subset dom(\Gamma)$, and the substitution preserves well-scoping of contexts:

Lemma 8. *If $\Gamma, y : \psi, \Gamma' Ok$ and $FV(b) \subset dom(\Gamma)$, then $\Gamma, [b/y]\Gamma' Ok$.*

Proof. For each variable z in Γ', if the entire context looks like $\Gamma, y : \psi, \Gamma'', z : \phi, \Gamma'''$ we know that $FV([b/y]\phi) \subset FV(\phi) \cup FV(b) - \{y\}$ and $dom(\Gamma, [b/y]\Gamma'') = dom(\Gamma, y : \psi, \Gamma'') - \{y\}$, so $[b/y]\phi$ is still well-scoped. □

Case:

$$\frac{a \downarrow a'}{\Gamma, y:\psi, \Gamma' \vdash x:\phi \vdash \mathtt{join} : a = a'}$$

Immediate by the fact that substitution preserves joinability.

Case:

$$\frac{\Gamma, y:\psi, \Gamma' \vdash a''' : a' = a'' \quad \Gamma, y:\psi, \Gamma' \vdash a : [a'/x]\phi \quad x \notin dom(\Gamma, y:\psi, \Gamma')}{\Gamma, y:\psi, \Gamma' \vdash a : [a''/x]\phi}$$

First, rename x in ϕ so that $x \notin FV(b)$.

By IH we get $\Gamma, [b/y]\Gamma' \vdash [b/y]a''' : [b/y]a' = [b/y]a''$ and $\Gamma, [b/y]\Gamma' \vdash [b/y]a : [b/y][a'/x]\phi \equiv [[b/y]a'/x][b/y]\phi$. Now by `(conv)` we conclude $\Gamma, [b/y]\Gamma' \vdash [b/y]a : [[b/y]a''/x]\phi \equiv [b/y][a''/x]\phi$ as required.

Case:

$$\frac{\Gamma, y:\psi, \Gamma', x:\phi' \vdash a:\phi \quad x \notin FV(a)}{\Gamma, y:\psi, \Gamma' \vdash a : \forall x:\phi'.\phi}$$

First, rename x in the derivation of $\Gamma, y:\psi, \Gamma', x:\phi' \vdash a:\phi$ so that $x \notin FV(a) \cup FV(b)$. This can be done without changing the depth.

By IH we get $\Gamma, [b/y]\Gamma', x:[b/y]\phi' \vdash [b/y]a : [b/y]\phi$. The conclusion follows by Spec-Abs.

Case:

$$\frac{\Gamma, y:\psi, \Gamma' \vdash a : \forall x:\phi'.\phi \quad \Gamma, y:\psi, \Gamma' \vdash a' : \phi'}{\Gamma, y:\psi, \Gamma' \vdash a : [a'/x]\phi}$$

Pick $x \neq y$ and $x \notin FV(b)$.

By IH we get $\Gamma, [b/y]\Gamma' \vdash [b/y]a : [b/y]\forall x:\phi'.\phi \equiv \forall x:[b/y]\phi'.[b/y]\phi$ and $\Gamma, [b/y]\Gamma' \vdash [b/y]a' : [b/y]\phi'$. Then by Spec-App, $\Gamma, [b/y]\Gamma' \vdash [b/y]a : [[b/y]a'/x][b/y]\phi \equiv [b/y][a'/x]\phi$ as required.

Case:

$$\frac{\Gamma, y:\psi, \Gamma' \vdash a:\phi}{\Gamma, y:\psi, \Gamma', x:\phi' \vdash \lambda x.a : \Pi x:\phi'.\phi}$$

Similar to Spec-abs.

Case:

$$\frac{\Gamma, y:\psi, \Gamma' \vdash a : \Pi x:\phi'.\phi \quad \Gamma, y:\psi, \Gamma' \vdash a' : \phi'}{\Gamma, y:\psi, \Gamma' \vdash (a\ a') : [a'/x]\phi}$$

Similar to Spec-App.

Case:

$$\frac{}{\Gamma \vdash 0 : \mathtt{nat}}$$

Immediate.

Case:

$$\frac{}{\Gamma \vdash \texttt{nil} : \langle \texttt{vec}\ \phi\ 0 \rangle}$$

Immediate.

Case:

$$\frac{\Gamma \vdash a : \texttt{nat}}{\Gamma \vdash (S\ a) : \texttt{nat}}$$

Immediate by IH.

Case:

$$\frac{\begin{array}{l}\Gamma, y_0 : \psi, \Gamma' \vdash a'' : \texttt{nat} \\ \Gamma, y_0 : \psi, \Gamma' \vdash a : [0/x]\phi \\ \Gamma, y_0 : \psi, \Gamma' \vdash a' : \Pi y : \texttt{nat}.\Pi u : [y/x]\phi.[(Sy)/x]\phi\end{array}}{\Gamma, y_0 : \psi, \Gamma' \vdash (R_{\texttt{nat}}\ a\ a'\ a'') : [a''/x]\phi}$$

First rename x in ϕ so that $x \notin FV(b) \cup \{z, y\}$.

IH gives

$$\Gamma, [b/y_0]\Gamma' \vdash a : [b/y_0][0/x]\phi \equiv [0/x][b/y_0]\phi$$

and

$$\Gamma, [b/y_0]\Gamma' \vdash a' : [b/y_0]\Pi y : \texttt{nat}.\Pi u : [y/x]\phi.[(Sy)/x]\phi \equiv \Pi y : \texttt{nat}.\Pi u : [y/x][b/y_0]\phi.[(Sy)/x][b/y_0]\phi$$

Then by Rnat,

$$\Gamma, [b/y_0]\Gamma' \vdash (R_{\texttt{nat}}\ a\ a'\ a'') : [[b/y_0]a''/x][b/y_0]\phi \equiv [b/y_0][a''/x]\phi$$

Case:

$$\frac{\begin{array}{l}\Gamma, y : \psi, \Gamma' \vdash a : \phi \\ \Gamma, y : \psi, \Gamma' \vdash a' : \langle \texttt{vec}\ \phi\ l \rangle\end{array}}{\Gamma, y : \psi, \Gamma' \vdash (\texttt{cons}\ a\ a') : \langle \texttt{vec}\ \phi\ (S\ l) \rangle}$$

Immediate by IH.

Case:

$$\frac{\begin{array}{l}\Gamma \vdash a'' : \langle \texttt{vec}\ \phi'\ l \rangle \\ \Gamma \vdash a : [0/y, \texttt{nil}/x]\phi \\ \Gamma \vdash a' : \Pi z : \phi'.\forall l : \texttt{nat}.\Pi v : \langle \texttt{vec}\ \phi'\ l \rangle.\Pi u : [l/y, v/x]\phi. \\ \qquad [(S\ l)/y, (\texttt{cons}\ z\ v)/x]\phi\end{array}}{\Gamma \vdash (R_{\texttt{vec}}\ a\ a'\ a'') : [l/y, a''/x]\phi}$$

Similar to Rnat case.

B Proof of Progress (Theorem 2)

The proof is by induction on $\Gamma \vdash a : \phi$.

Case:

$$\frac{\Gamma(x) \equiv \phi}{\Gamma \vdash x : \phi}$$

Impossible by $dom(\Gamma) \cap FV(v) = \emptyset$.

Case:

$$\frac{a \downarrow a'}{\Gamma \vdash \texttt{join} : a = a'}$$

`join` is a value, as required.

Case:

$$\frac{\Gamma \vdash a''' : a' = a'' \quad \Gamma \vdash a : [a'/x]\phi \quad x \notin dom(\Gamma)}{\Gamma \vdash a : [a''/x]\phi}$$

Directly by the IH for $\Gamma \vdash a : [a'/x]\phi$.

Case:

$$\frac{\Gamma, x : \phi' \vdash a : \phi \quad x \notin FV(a)}{\Gamma \vdash a : \forall x : \phi'.\phi}$$

The condition $x \notin FV(a)$ ensures that $dom(\Gamma, x : \phi') \cap FV(a) = \emptyset$, so we can apply the IH for $\Gamma, x : \phi' \vdash a : \phi$.

Case:

$$\frac{\Gamma \vdash a : \forall x : \phi'.\phi \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash a : [a'/x]\phi}$$

Directly by the IH for $\Gamma \vdash a : \forall x : \phi'.\phi$.
Case:

$$\frac{\Gamma, x : \phi' \vdash a : \phi}{\Gamma \vdash \lambda x.a : \Pi x : \phi'.\phi}$$

$\lambda x.a$ is a value as required.

Case:

$$\frac{\Gamma \vdash a : \Pi x : \phi'.\phi \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash (a\ a') : [a'/x]\phi}$$

By the IH for $\Gamma \vdash a : \Pi x : \phi'.\phi$, we know a either steps or is a value. If a steps, the entire expression $(a\ a')$ steps also, by $\rightsquigarrow$-congruence. If a is a value, by Lemma **??** $a = \lambda a.a_o$, so $(a\ a')$ steps by β.

Case:

$$\frac{}{\Gamma \vdash 0 : \texttt{nat}}$$

0 is a value as required.

Case:

$$\frac{}{\Gamma \vdash \texttt{nil} : \langle \texttt{vec}\ \phi\ 0 \rangle}$$

`nil` is a value as required.

Case:

$$\frac{\Gamma \vdash a : \texttt{nat}}{\Gamma \vdash (S\ a) : \texttt{nat}}$$

By the IH for $\Gamma \vdash a : \texttt{nat}$, we know a either steps or is a value; accordingly Sa steps or is a value.

Case:

$$\frac{\begin{array}{l}\Gamma \vdash a'' : \texttt{nat} \\ \Gamma \vdash a : [0/x]\phi \\ \Gamma \vdash a' : \Pi y : \texttt{nat}.\Pi u : [y/x]\phi.[(Sy)/x]\phi\end{array}}{\Gamma \vdash (R_{\texttt{nat}}\ a\ a'\ a'') : [a''/x]\phi}$$

By the IH for $\Gamma \vdash a'' : \texttt{nat}$ and Lemma **??**, we know that either a'' steps or $a'' = 0$ or $a'' = S\ v$. Then $(R_{\texttt{nat}}\ a\ a'\ a'')$ steps by congruence or one of the two reduction rules, respectively.

Case:

$$\frac{\begin{array}{l}\Gamma \vdash a : \phi \\ \Gamma \vdash a' : \langle \texttt{vec}\ \phi\ l \rangle\end{array}}{\Gamma \vdash (\texttt{cons}\ a\ a') : \langle \texttt{vec}\ \phi\ (S\ l) \rangle}$$

By the IH, a and a' either step or are values; accordingly $(\texttt{cons}\ a\ a')$ steps by congruence or is a value.

Case:

$$\frac{\begin{array}{l}\Gamma \vdash a'' : \langle \texttt{vec}\ \phi'\ l \rangle \\ \Gamma \vdash a : [0/y, \texttt{nil}/x]\phi \\ \Gamma \vdash a' : \Pi z : \phi'.\forall l : \texttt{nat}.\Pi v : \langle \texttt{vec}\ \phi'\ l \rangle.\Pi u : [l/y, v/x]\phi. \\ \qquad\quad [(S\ l)/y, (\texttt{cons}\ z\ v)/x]\phi \end{array}}{\Gamma \vdash (R_{\texttt{vec}}\ a\ a'\ a'') : [l/y, a''/x]\phi}$$

By the IH for $\Gamma \vdash a'' : \langle \texttt{vec}\ \phi'\ l \rangle$ and Lemma **??**, we know that either a'' steps or $a'' = \texttt{nil}$ or $a'' = (\texttt{cons}\ v\ v')$. Then $(R_{\texttt{vec}}\ a\ a'\ a'')$ steps by congruence or by one of the two reduction rules, respectively.

B.1 Proof of Canonical Forms (Lemma ??)

Induction on the typing judgement. The cases are

- **var** Impossible by $dom(\Gamma) \cap FV(v) = \emptyset$.
- **conv**

  $$\frac{\Gamma \vdash a''' : a' = a'' \quad \Gamma \vdash a : [a'/x]\phi \quad x \notin dom(\Gamma)}{\Gamma \vdash a : [a''/x]\phi}$$

  The types $[a''/x]\phi$ and $[a'/x]\phi$ have the same top-level structure, so the IH applies.
- **spec-abs**

  $$\frac{\Gamma, x : \phi' \vdash a : \phi \quad x \notin FV(a)}{\Gamma \vdash a : \forall x : \phi'.\phi}$$

  The condition $x \notin FV(a)$ ensures that $dom(\Gamma, x : \phi') \cup FV(v) = \emptyset$. Also, the type ϕ still has the required form. So the IH applies.
- **spec-app**

  $$\frac{\Gamma \vdash a : \forall x : \phi'.\phi \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash a : [a'/x]\phi}$$

  If the type $[a'/x]\phi$ has the required form, then $\forall x : \phi'.\phi$ has required form also, so the IH applies.
- **join,abs,zero,nil,succ,cons** The value in the conclusion has the required form.
- **app, Rnat, Rvec** The term in the conclusion is not a value.

C Proof of Critical Properties (Section 3.3)

C.1 More basic notation

We will define a term context to be a term a^* with a designated free variable $*$, which may be instantiated in a capture-avoiding way by a term a' using the notation $a^*[a']$.

Also, if S is a set of terms, then we will allow ourselves to write a term which has S inserted for the hole of some context a^*. For example, we may write $\lambda x.\{x, (x\ x)\}$ (here $a^* = \lambda x.*$). The meaning of this notation is the set of terms $\{a^*[a'] | a' \in S\}$. So in the example: $\{\lambda x.x,\ \lambda x.(x\ x)\}$.

Finally, if $a \in SN$, we define $\nu(a)$ to be some bound on the lengths of the reduction sequences from a.

C.2 Preliminary observation

We will not explicitly prove strong normalization or typability in the cases for **R-Join** below. This is because **R-Join** assumes $\Gamma \vdash a' : a_1 = a_2$ for some a', so we will always be able to show $\Gamma \vdash a : [a_2/x]\phi$ from $\Gamma \vdash a : [a_1/x]\phi$ using `(conv)`. Similarly, we will always have $a \in SN$, since it follows from $a \in [\![[a_1/x]\phi]\!]_\Gamma$.

C.3 Critical properties for `nat`

R-Pres holds using Type Preservation (Theorem 1), and the fact that $a \in SN \Rightarrow next(a) \subset SN$. For **R-Prog**, we have $next(a) \subset SN \Rightarrow a \in SN$, and $\Gamma \vdash a : \texttt{nat}$ by assumption (of **R-Prog**). We have **R-Join** because all instances $[a/x]\texttt{nat} \equiv \texttt{nat}$ have the same (trivial) interpretation.

C.4 Critical properties for $\langle$`vec` $\phi\ l\rangle$

C.4.1 Proof of R-Pres

Suppose $a \in [\![\langle \texttt{vec}\ \phi\ l\rangle]\!]_\Gamma$, and $a \rightsquigarrow a'$. We have $a' \in SN$ from $a \in SN$, and $\Gamma \vdash a' : \phi$ by Type Preservation. To prove the second conjunct for $[\![\cdot]\!]$ at `vec`-type – namely, $(a' \rightsquigarrow^* \texttt{nil} \Rightarrow l \sim_\Gamma 0)$ – assume $a' \rightsquigarrow^* \texttt{nil}$. From this and the assumption that $a \rightsquigarrow a'$, we have $a \rightsquigarrow^* \texttt{nil}$. So we can use the corresponding conjunct for a to conclude $l \sim_\Gamma 0$, as required. Similar reasoning applies for the third conjunct.

C.4.2 Proof of R-Prog

Suppose $next(a) \subset [\![\langle \texttt{vec}\ \phi\ l\rangle]\!]_\Gamma$, with a neutral and $\Gamma \vdash a : \phi$. We have $a \in SN$ for the same reason as for the `nat` case above. It suffices now to show the two conjuncts of the clause of $[\![\cdot]\!]$ for `vec`-types, for a. For the first, assume $a \rightsquigarrow^* \texttt{nil}$. Now since a is neutral, we cannot have $a \equiv \texttt{nil}$. So consider arbitrary $a' \in next(a)$. From $a \rightsquigarrow^* \texttt{nil}$ and $a \rightsquigarrow a'$, we obtain $a' \rightsquigarrow^* \texttt{nil}$ by confluence. We may then apply the corresponding second conjunct for a' to obtain the desired result for this conjunct. The second conjunct follows by similar reasoning.

C.4.3 Proof of R-Join

Suppose that $a_1 \sim_\Gamma a_2$, and consider arbitrary $a \in [\![[a_1/x]\langle \texttt{vec}\ \phi\ l\rangle]\!]_\Gamma$. We must show $a \in [\![[a_2/x]\langle \texttt{vec}\ \phi\ l\rangle]\!]_\Gamma$. It suffices to prove the two conjuncts for $[\![\cdot]\!]$ at the type involving a_2, assuming them for the type involving a_1. For the first conjunct, assume $a \rightsquigarrow^* \texttt{nil}$. We must show that for an arbitrary $\sigma \in [\![\Gamma]\!]$, we have $\sigma([a_2/x]l) \downarrow 0$. From the second conjunct for the a_1-type, we have $\sigma([a_1/x]l) \downarrow 0$. Instantiate our first assumption about a_1 and a_2, to obtain $\sigma a_1 \downarrow \sigma a_2$. Now we use the fact that joinability is closed under substitution of joinable terms (proof omitted), to obtain the desired result. Note that joinability is not closed under substitution of joinable terms for more specialized reduction strategies, such as call-by-value or call-by-name.

For the second conjunct, assume $a \rightsquigarrow^* (\texttt{cons}\ a'\ a'')$. From the corresponding second conjunct for the a_1-type, we obtain the following for some l':

- $a' \in [\![[a_1/x]\phi]\!]_\Gamma$
- $a'' \in [\![\langle \texttt{vec}\ [a_1/x]\phi\ l'\rangle]\!]_\Gamma$
- $\forall \sigma \in [\![\Gamma]\!].\ \sigma([a_1/x]l) \downarrow (S\ \sigma l')$

We use **R-Join** on the first formula to derive the similar statement involving a_2. The measure $(|\Gamma|, d(\phi), l(a))$ decreases, because the depth of the type decreases (and $|\Gamma|$ is unchanged). Now let z be a variable not in $dom(\Gamma)$ and not free in ϕ, a_1, or l'. Then the second formula is equivalent to $a'' \in [\![[a_1/z]\langle \texttt{vec}\ [z/x]\phi\ l'\rangle]\!]_\Gamma$, and by **R-Join** (where the measure decreases because the depth of the type is the same, but the quantity given by $l(\cdot)$ has decreased), we have $a'' \in [\![[a_2/z]\langle \texttt{vec}\ [z/x]\phi\ l'\rangle]\!]_\Gamma$, which is equivalent to the required $a'' \in [\![\langle \texttt{vec}\ [a_2/x]\phi\ l'\rangle]\!]_\Gamma$. Instantiating the third formula with an arbitrary $\sigma \in [\![\Gamma]\!]$, we have $\sigma([a_1/x]l) \downarrow (S\ l')$. We appeal as above to the closure of joinability under substitution of joinable terms, to obtain $\sigma([a_2/x]l) \downarrow (S\ l')$.

C.5 Critical properties for $\Pi x : \phi'.\phi$

C.5.1 Proof of R-Pres

Assume $a \in [\![\Pi x : \phi'.\phi]\!]_\Gamma$, and consider an arbitrary $a' \in [\![\phi']\!]^+_\Gamma$. By definition of $[\![\cdot]\!]$, we have $(a\ a') \in [\![[a'/x]\phi]\!]_\Gamma$. We also have

$$(next(a)\ a')\ \subset\ next(a\ a') \tag{1}$$

By **R-Pres** at type $[a'/x]\phi$ (where we have $d([a'/x]\phi) < d(\Pi x : \phi'.\phi)$, and so can apply the induction hypothesis), we obtain $(next(a\ a')) \subset [\![[a'/x]\phi]\!]_\Gamma$. By (1), this implies $(next(a)\ a') \subset [\![[a'/x]\phi]\!]_\Gamma$. We conclude this for all $a' \in [\![\phi']\!]^+_\Gamma$. Then by the definition of $[\![\cdot]\!]$, we obtain the desired $next(a) \subset [\![\Pi x : \phi'.\phi]\!]_\Gamma$, using also Type Preservation and the fact that $next(a) \subset SN$ (since $a \in SN$).

C.5.2 Proof of R-Prog

Suppose a is neutral with $\Gamma \vdash a : \Pi x : \phi'.\phi$. By assumption, we have

$$next(a) \subset [\![\Pi x : \phi'.\phi]\!]_\Gamma \tag{2}$$

It suffices, by the definition of $[\![\cdot]\!]$, to show that $a \in SN$ and for all $a' \in [\![\phi']\!]^+_\Gamma$, $(a\ a') \in [\![[a'/x]\phi]\!]_\Gamma$. We have $a \in SN$ from $next(a) \subset SN$, so we focus on the latter property. Consider arbitrary $a' \in [\![\phi']\!]^+_\Gamma$. Since a is neutral, $(a\ a')$ cannot be a β-redex. Since $a' \in [\![\phi']\!]_\Gamma$, we have $a' \in SN$ by **R-SN** at type ϕ' (where $d(\phi') < d(\Pi x : \phi'.\phi)$, so the induction hypothesis may be applied). So we may reason by inner induction on the number $\nu(a')$ to prove that for all $a' \in [\![\phi']\!]^+_\Gamma$, we have $(a\ a') \in [\![[a'/x]\phi]\!]_\Gamma$. By **R-Prog** at type $[a'/x]\phi$ (where $d([a'/x]\phi) < d(\Pi x : \phi'.\phi)$, so the induction hypothesis may be applied), it suffices to prove $next(a\ a') \subset [\![[a'/x]\phi]\!]_\Gamma$, since the term in question is neutral and since we have $\Gamma \vdash (a\ a') : [a'/x]\phi$. The possibilities for reduction are summarized by:

$$next(a\ a')\ \subset\ (next(a)\ a')\ \cup\ (a\ next(a'))$$

We have $(next(a)\ a') \in [\![[a'/x]\phi]\!]_\Gamma$ from (2), by the definition of $[\![\cdot]\!]$. For reducibility of the second set, consider arbitrary $a'' \in next(a')$. By our inner induction hypothesis, which we may apply because $a'' \in [\![\phi']\!]_\Gamma$ by **R-Pres** at type ϕ' (with smaller depth), we have $(a\ a'') \in [\![[a''/x]\phi]\!]_\Gamma$. Now we may apply **R-Join** at type ϕ (with smaller depth), using the obvious fact that $a' \rightsquigarrow a''$ implies the facts $a' \sim_\Gamma a''$ and $\Gamma \vdash \texttt{join} : a' = a''$ (required by **R-Join**). This yields $(a\ a'') \in [\![[a'/x]\phi]\!]_\Gamma$, as required by our inner induction.

C.5.3 Proof of R-Join

Suppose that $a_1 \sim_\Gamma a_2$, and consider arbitrary $a \in [\![[a_1/x]\Pi y : \phi'.\phi]\!]_\Gamma$. We must show $a \in [\![[a_2/x]\Pi y : \phi'.\phi]\!]_\Gamma$. It suffices to show $(a\ a') \in [\![[a'/y][a_2/x]\phi]\!]_\Gamma$ for an arbitrary $a' \in [\![[a_2/x]\phi']\!]^+_\Gamma$. We now wish to use **R-Join** at type ϕ' (with smaller depth), with the symmetric equality $a_2 \sim_\Gamma a_1$. Symmetry of $\sim_\Gamma$ is direct from its definition.

Using **R-Join** in this way with $a_2 \sim_\Gamma a_1$, we obtain $a' \in [\![[a_1/x]\phi']\!]_\Gamma$. We must further obtain $a' \in [\![[a_1/x]\phi']\!]_\Gamma^+$. So consider arbitrary $\sigma \in [\![\Gamma]\!]$. From closability of a' at the type involving a_2, we have $\sigma a' \in [\![\sigma([a_2/x]\phi')]\!]$. We must show $\sigma a' \in [\![\sigma([a_1/x]\phi')]\!]$. If Γ is empty, this formula is equivalent to $a' \in [\![[a_1/x]\phi']\!]$, which we already have. So suppose Γ is not empty. Then the formula is equivalent to $\sigma a' \in [\![[\sigma a_1/x](\sigma\phi')]\!]$, since $x \notin \mathit{ran}(\sigma)$. Notice that from our assumption that $a_1 \sim_\Gamma a_2$, we obtain $\sigma a_1 \sim \sigma a_2$. We may now use **R-Join**, where the length of the context has decreased, to conclude $\sigma a' \in [\![[\sigma a_1/x](\sigma\phi')]\!]$ from $\sigma a' \in [\![[\sigma a_2/x](\sigma\phi')]\!]$.

Since we have obtained $a' \in [\![[a_1/x]\phi']\!]_\Gamma^+$, we now get $(a\ a') \in [\![[a'/y][a_1/x]\phi]\!]_\Gamma$ by the assumption above of reducibility of a. Applying Lemma 7 to the fact that $\Gamma \vdash a' : [a_1/x]\phi'$ (which we have from $a' \in [\![[a_1/x]\phi']\!]_\Gamma$), and using the assumption that $x \notin \mathit{dom}(\Gamma)$, we obtain $x \notin \mathit{FV}(a')$. Since y is locally scoped, we may also assume that $y \notin \mathit{FV}(a_1)$ and $y \notin \mathit{FV}(a_2)$. This tells us that $[a'/y][a_1/x]\phi = [a_1/x][a'/y]\phi$ and also $[a'/y][a_2/x]\phi = [a_2/x][a'/y]\phi$. Using the first of these, we may conclude $(a\ a') \in [\![[a_1/x][a'/y]\phi]\!]_\Gamma$ from the similar fact we had just above. Using the second of these commutations of substitutions, and also **R-Join** at type $[a'/y]\phi$ (of smaller depth), we can conclude $(a\ a') \in [\![[a'/y][a_2/x]\phi]\!]_\Gamma$, as required.

C.6 Critical properties for $\forall x : \phi'.\phi$

The proofs here are simpler versions (particularly for **R-Prog**) of those for the previous case.

C.6.1 Proof of R-Pres

Assume $a \in [\![\forall x : \phi'.\phi]\!]_\Gamma$, and consider an arbitrary $a' \in [\![\phi']\!]_\Gamma^+$. By **R-Pres** at type $[a'/x]\phi$ (with smaller depth), we obtain $\mathit{next}(a) \subset [\![[a'/x]\phi]\!]_\Gamma$. We conclude this for all $a' \in [\![\phi']\!]_\Gamma^+$. Then by the definition of $[\![\cdot]\!]$, we get the required $\mathit{next}(a) \subset [\![\forall x : \phi'.\phi]\!]_\Gamma$, using also Type Preservation and the fact $\mathit{next}(a) \subset \mathit{SN}$ (from $a \in \mathit{SN}$).

C.6.2 Proof of R-Prog

Suppose a is neutral with $\Gamma \vdash a : \phi$. By assumption, we have

$$\mathit{next}(a) \subset [\![\forall x : \phi'.\phi]\!]_\Gamma$$

It suffices, by the definition of $[\![\cdot]\!]$, to show that $a \in \mathit{SN}$ and for all $a' \in [\![\phi']\!]_\Gamma^+$, $a \in [\![[a'/x]\phi]\!]_\Gamma$. We have $a \in \mathit{SN}$ from $\mathit{next}(a) \subset \mathit{SN}$, so we focus on the latter property. Consider arbitrary $a' \in [\![\phi']\!]_\Gamma^+$. By the definition of $[\![\cdot]\!]$ at $\forall$-type and our above assumption, we have $\mathit{next}(a) \subset [\![[a'/x]\phi]\!]_\Gamma$. So by **R-Prog** at type $[a'/x]\phi$ (with smaller depth), we have $a \in [\![[a'/x]\phi]\!]_\Gamma$, as required.

C.6.3 Proof of R-Join

Suppose that $a_1 \sim_\Gamma a_2$, and consider arbitrary $a \in [\![[a_1/x]\forall y : \phi'.\phi]\!]_\Gamma$. We must show $a \in [\![[a_2/x]\forall y : \phi'.\phi]\!]_\Gamma$. It suffices to show $a \in [\![[a'/y][a_2/x]\phi]\!]_\Gamma$ for an arbitrary $a' \in [\![[a_2/x]\phi']\!]_\Gamma^+$. By **R-Join** at type ϕ' (with smaller depth), and using the symmetric version of our assumption as above, we have $a' \in [\![[a_1/x]\phi']\!]_\Gamma$. We further obtain $a' \in [\![[a_1/x]\phi']\!]_\Gamma^+$ as in the case for **R-Prog** for Π-types. So we get $a \in [\![[a'/y][a_1/x]\phi]\!]_\Gamma$ by the assumption of reducibility of a. By similar reasoning as above, we may permute the substitutions in question. So we may apply **R-Join** at type $[a'/y]\phi$ (of smaller depth) to conclude $a \in [\![[a'/y][a_2/x]\phi]\!]_\Gamma$, as required.

C.7 Critical properties for $a_1 = a_2$

C.7.1 Proof of R-Pres

Consider arbitrary b with $a \rightsquigarrow b$. We have $b \in SN$ from $a \in SN$, and $\Gamma \vdash b : a_1 = a_2$ by Type Preservation. Now suppose $b \rightsquigarrow^* \texttt{join}$. Then we have $a \rightsquigarrow^* \texttt{join}$ and obtain $a_1 \sim_\Gamma a_2$ from $a \in [\![a_1 = a_2]\!]_\Gamma$.

C.7.2 Proof of R-Prog

We have $a \in SN$ from $next(a) \subset SN$ as in other cases above. Suppose that $a \rightsquigarrow^* \texttt{join}$. Since a is neutral, we cannot have $a \equiv \texttt{join}$. So we must have $a \rightsquigarrow b$. Then we get $b \rightsquigarrow^* \texttt{join}$ by confluence, and we can use the assumption that $b \in [\![a_1 = a_2]\!]_\Gamma$ to obtain $a_1 \sim_\Gamma a_2$ as required.

C.7.3 Proof of R-Join

Assume $a_1' \sim_\Gamma a_2'$, and assume $a \rightsquigarrow^* \texttt{join}$. Then we have $[a_1'/x]a_1 \sim_\Gamma [a_1'/x]a_2$ from $a \in [\![[a_1'x]a_1 = [a_1'/x]a_2]\!]_\Gamma$. Consider arbitrary $\sigma \in [\![\Gamma]\!]$. Instantiating our two assumptions of joinability under all ground instances with this σ, we obtain:

- $\sigma a_1' \downarrow \sigma a_2'$
- $(\sigma([a_1'/x]a_1)) \downarrow (\sigma([a_1'/x]a_2))$

The desired result (namely, $(\sigma([a_2'/x]a_1)) \downarrow (\sigma([a_2'/x]a_2)))$ now follows from closure of joinability under substitution of joinable terms.

D Proof of Soundness (Theorem 3)

D.1 The Closability Lemma

We have carefully crafted our notions of closable terms and closable substitutions to allow the following two lemmas to be proved. The first expresses the basic desired property of closable substitutions, and the second shows that under the conditions of the Soundness Theorem, the term σa is closable which Soundness tells us is in the interpretation of $\sigma\phi$ with context Δ.

Lemma 9 (Composing Substitutions)**.** *Suppose* $\sigma \in [\![\Gamma]\!]_\Delta$ *and* $\sigma' \in [\![\Delta]\!]$. *Then* $\sigma' \circ \sigma \in [\![\Gamma]\!]$.

The proof is by induction on the structure of the derivation of $\sigma \in [\![\Gamma]\!]_\Delta$. The base case holds trivially, noting that $\sigma' \circ \emptyset = \emptyset$. For the step case, we have

$$\frac{a \in [\![\sigma''\phi]\!]^+_\Delta \quad \sigma'' \in [\![\Gamma']\!]_\Delta}{\sigma'' \cup \{(x,a)\} \in [\![\Gamma', x : \phi]\!]_\Delta}$$

Now we obtain $\sigma'(\sigma''a) \in [\![\sigma'(\sigma''\phi)]\!]$, by the definition of closability of a. This implies $\sigma'(\sigma''a) \in [\![\sigma'(\sigma''\phi)]\!]^+$, since the definitions of $[\![\cdot]\!]$ and $[\![\cdot]\!]^+$ coincide when the context is empty. By the induction hypothesis we have $\sigma' \circ \sigma'' \in [\![\Gamma']\!]_\Delta$. So we may reapply the rule to obtain the desired $(\sigma' \circ \sigma'') \cup \{(x, \sigma'a)\} \in [\![\Gamma', x : \phi]\!]$.

Lemma 10 (Closability)**.** *Suppose the following main assumption is true: for any* $\Delta\, Ok$ *and* $\sigma \in [\![\Gamma]\!]_\Delta$, *we have* $(\sigma a) \in [\![\sigma\phi]\!]_\Delta$. *In this case, for any such* Δ *and* σ, *we also have* $(\sigma a) \in [\![\sigma\phi]\!]^+_\Delta$.

Assume an arbitrary $\sigma' \in [\![\Delta]\!]$. We must show $\sigma'(\sigma a) \in [\![\sigma'(\sigma\phi)]\!]$. By Composing Substitutions (Lemma 9), we have $\sigma' \circ \sigma \in [\![\Gamma]\!]$. So we may instantiate the main assumption with $\sigma' \circ \sigma$ to obtain the the required formula.

D.2 The Proof

The proof of the Soundness Theorem is by induction on the structure of the assumed typing derivation. We consider all cases, and implicitly start each by assuming an arbitrary $\sigma \in [\![\Gamma]\!]_\Delta$. We often will use this σ to instantiate universal formulas obtained by application of our induction hypothesis, without explicitly noting that we are instantiating the induction hypothesis. If σ is a substitution, we will write $\sigma[a'/x]$ for the substitution that extends σ by mapping x to a'.

Case:

$$\frac{\Gamma(x) \equiv \phi}{\Gamma \vdash x : \phi}$$

We prove $\sigma x \in [\![\sigma\Gamma(x)]\!]_\Delta^+$ by inner induction on the structure of $\sigma \in [\![\Gamma]\!]_\Delta$. The base case cannot arise, since we have $\Gamma(x)$ defined. For the step case, we have:

$$\frac{a \in [\![\sigma'\phi]\!]_\Delta^+ \quad \sigma' \in [\![\Gamma']\!]_\Delta}{\sigma' \cup \{(y,a)\} \in [\![\Gamma', y : \phi]\!]_\Delta}$$

If $x \equiv y$, then we have $\sigma x \in [\![\sigma'\Gamma(x)]\!]_\Delta$ from the first premise. We just need to show $\sigma\Gamma(x) \equiv \sigma'\Gamma(x)$. But this follows from the fact that $x \notin FV(\phi)$ (by Γ *Ok*). If $x \not\equiv y$, then by the inner induction hypothesis we have $\sigma' x \in [\![\sigma'\Gamma'(x)]\!]_\Delta$. We must show that this implies the desired $\sigma x \in [\![\sigma\Gamma(x)]\!]_\Delta$. We certainly have $\sigma' x \equiv \sigma x$, and $\Gamma'(x) \equiv \Gamma(x)$. So it suffices to show that $\sigma'\Gamma(x) \equiv \sigma\Gamma(x)$. But $\Gamma(x)$ cannot contain x, so this holds.

Case:

$$\frac{a \downarrow a'}{\Gamma \vdash \mathtt{join} : a = a'}$$

If $a \downarrow a'$, we certainly also have $\sigma a \downarrow \sigma a'$, since joinability is closed under substitution. This gives us $\Delta \vdash \mathtt{join} : \sigma a = \sigma a'$. Again by closure of joinability under substitution, we have $\sigma a \sim_\Delta \sigma a$, since for any $\sigma' \in [\![\Delta]\!]$, we certainly have $\sigma'(\sigma a) \downarrow \sigma'(\sigma a')$. We obviously have $\mathtt{join} \in SN$, so we conclude $\mathtt{join} \in [\![\sigma(a_1 = a_2)]\!]_\Delta$.

Case:

$$\frac{\Gamma \vdash a''' : a' = a'' \quad \Gamma \vdash a : [a'/x]\phi \quad x \notin dom(\Gamma)}{\Gamma \vdash a : [a''/x]\phi}$$

The required conclusion follows by **R-Join** from $\sigma a \in [\![[\sigma a'/x](\sigma\phi)]\!]_\Delta$, which we have from the induction hypothesis for the second premise. To enable this use of **R-Join**, we need $\Delta \vdash \sigma a''' : \sigma a' = \sigma a''$ and $\sigma a' \sim_\Delta \sigma a''$. The former we obtain from $\sigma a''' \in [\![\sigma a' = \sigma a'']\!]_\Delta$, which we have by the induction hypothesis for the first premise. The latter we obtain as follows. Consider an arbitrary $\sigma' \in [\![\Delta]\!]$. From this and the fact that $\sigma \in [\![\Gamma]\!]_\Delta$, we have $\sigma' \circ \sigma \in [\![\Gamma]\!]$ by Composing Substitutions (Lemma 9).

Since $\sigma' \circ \sigma \in [\![\Gamma]\!]$, we can use it to instantiate the induction hypothesis for the first premise. This gives us $\sigma'(\sigma a''') \in [\![\sigma'(\sigma a') = \sigma'(\sigma a'')]\!]$, which implies $\cdot \vdash \sigma'(\sigma a''') : \sigma'(\sigma a') = \sigma'(\sigma a'')$. So consider the unique normal form n of $\sigma'(\sigma a''')$, which exists by confluence and **R-SN**. We have $n \in [\![\sigma'(\sigma a') = \sigma'(\sigma a'')]\!]$ by repeated application of **R-Pres**. This implies $n : \sigma'(\sigma a') = \sigma'(\sigma a'')$. By Progress, this n

must be a value. We may now apply Canonical Forms (Lemma **??**), to conclude that $n \equiv \texttt{join}$. Now by the definition of $[\![\sigma'(\sigma a') = \sigma'(\sigma a'')]\!]$, we have $\sigma'(\sigma a') \downarrow \sigma'(\sigma a'')$, as required. We assumed an arbitrary $\sigma' \in [\![\Delta]\!]$, so we may conclude that $\sigma'(\sigma a') \downarrow \sigma'(\sigma a'')$ holds for all such σ'. This is sufficient for $\sigma a' \sim_\Delta \sigma a''$.

Case:

$$\frac{\Gamma, x : \phi' \vdash a : \phi \quad x \notin FV(a)}{\Gamma \vdash a : \forall x : \phi'.\phi}$$

From the induction hypothesis, we infer the following, for any $\sigma' \in [\![\Gamma, x : \phi']\!]_{\Delta'}$, for any $\Delta' \subset \sigma(\Gamma, x : \phi')$:

$$\forall a' \in [\![\sigma\phi']\!]^+_\Delta.\ (\sigma[a'/x])a \in [\![(\sigma[a'/x])\phi]\!]_\Delta \tag{3}$$

Our first need is to prove $a \in SN$. For this, we instantiate (3) with $\Delta' \equiv \Delta, x : \sigma\phi'$; $\sigma' \equiv \sigma[x/x]$; and $a' \equiv x$. Note that it is at precisely this point that we critically need open substitutions in the statement of Soundness. To show that this instantiation is legal, we must, of course, prove that $\sigma' \in [\![\Gamma]\!]_{\Delta'}$. For this, we need two things. First, we need to know that $x \in [\![\sigma\phi']\!]^+_{\Delta,x:\sigma\phi'}$. This follows because $x \in [\![\sigma\phi']\!]_{\Delta,x:\sigma\phi'}$ by **R-Prog** (since $next(a) = \emptyset$); and further, for any $\sigma' \in [\![\Delta, x : \sigma\phi']\!]$, we derive from that same fact the formula $\sigma' x \in [\![\sigma'(\sigma\phi')]\!]$, which we require for closability of x. Now we use the following lemma (proof in Section D.3 below) to finish our proof of the intermediate fact $\sigma' \in [\![\Gamma, x : \phi']\!]_{\Delta'}$:

Lemma 11 (Weakening Substitutions)**.** *If $\sigma \in [\![\Gamma]\!]_\Delta$ and $\Delta, y : \phi'$ Ok, then $\sigma \in [\![\Gamma]\!]_{\Delta,y:\phi'}$.*

Using this intermediate fact $\sigma' \in [\![\Gamma, x : \phi']\!]_{\Delta'}$, we may indeed instantiate (4) above with σ', Δ' and x for a', as mentioned. This gives us $\sigma a \in [\![\sigma\phi]\!]_{\Delta'}$. By **R-SN**, we then obtain $\sigma a \in [\![\sigma\phi]\!]_{\Delta,x:\sigma\phi'}$. From this, we obtain $\sigma a \in SN$ and $\Delta \vdash \lambda x.a : \Pi x : \sigma\phi'.\sigma\phi$, which we need to show $\sigma a \in [\![\Pi x : \sigma\phi'.\sigma\phi]\!]_\Delta$.

To complete this case, it suffices to consider arbitrary $a' \in [\![\sigma\phi']\!]^+_\Delta$, and show $\sigma a \in [\![[a'/x]\sigma\phi]\!]_\Delta$. Instantiating (3) with a', we obtain $(\sigma[a'/x])a \in [\![(\sigma[a'/x])\phi]\!]_\Delta$. This is equivalent to the goal, thanks to the following facts about the substitutions in question. Since $x \notin FV(a)$, we have $(\sigma[a'/x])a \equiv \sigma a$. Also, we have $x \notin ran(\sigma)$ by the following lemma. So we get the desired $\sigma a \in [\![[a'/x]\sigma\phi]\!]_\Delta$.

Lemma 12 (Basic Property of Substitutions)**.** $\sigma \in [\![\Gamma]\!]_\Delta \wedge \Gamma Ok \Rightarrow \sigma(x) \in [\![\sigma\Gamma(x)]\!]^+_\Delta$

The proof is by induction on the structure of the assumed derivation.

Case:

$$\frac{\Gamma \vdash a : \forall x : \phi'.\phi \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash a : [a'/x]\phi}$$

This follows immediately from induction hypothesis, and the definition of $[\![\cdot]\!]$ for $\forall$-types (here, the type $\forall x : \sigma\phi'.\sigma\phi$), using the fact that various substitutions involved commute, as in cases above. We critically use Closability (Lemma 10), to get $\sigma a' \in [\![\sigma\phi']\!]^+_\Delta$ from $\sigma a' \in [\![\sigma\phi']\!]_\Delta$.

Case:

$$\frac{\Gamma, x : \phi' \vdash a : \phi}{\Gamma \vdash \lambda x.a : \Pi x : \phi'.\phi}$$

We begin just as for the `(spec-abs)` case. From the induction hypothesis, we infer the following, for any $\sigma' \in [\![\Gamma, x:\phi']\!]_{\Delta'}$, for any $\Delta' \subset \sigma(\Gamma, x:\phi')$:

$$\forall a' \in [\![\sigma'\phi']\!]^+_{\Delta'}.\ (\sigma'[a'/x])a \in [\![(\sigma'[a'/x])\phi]\!]_{\Delta'} \tag{4}$$

By the same reasoning as for the `(spec-abs)` case, we obtain $\sigma a \in SN$ and $\Delta \vdash \lambda x.a : \Pi x:\sigma\phi'.\sigma\phi$.

Now by the definition of $[\![\cdot]\!]$, it suffices to prove that for all $a' \in [\![\sigma\phi']\!]^+_{\Delta}$, we have $((\lambda x.(\sigma a))\ a') \in [\![[a'/x](\sigma\phi)]\!]_{\Delta}$. We prove that (4) implies this, by inner induction on $\nu(\sigma a)+\nu(a')$, which is defined by **R-SN** (for σa and a'). By **R-Prog**, it suffices to prove $next((\lambda x.(\sigma a))\ a') \subset [\![[a'/x](\sigma\phi)]\!]_{\Delta}$, since the term in question (i.e., $((\lambda x.(\sigma a))\ a')$) is neutral and appropriately typable since σa is. In more detail, since we have $\sigma a \in [\![\sigma\phi]\!]_{\Delta,x:\sigma\phi'}$, we obtain $\Delta, x:\sigma\phi' \vdash \sigma a : \sigma\phi$. Then we apply the typing rule for λ-abstractions to obtain $\Delta \vdash \lambda x.\sigma a : \Pi x:\sigma\phi'.\sigma\phi$, and we conclude the typing proof with the application rule on this fact and $\Delta \vdash a' : \sigma\phi'$.

Now the possibilities for reduction of the term in question are summarized by:

$$next((\lambda x.\sigma a)\ a')\ \subset\ ((\lambda x.next(\sigma a))\ a')\ \cup\ ((\lambda x.\sigma a)\ next(a'))\ \cup\ \{[a'/x]\sigma a\}$$

We have $((\lambda x.next(\sigma a))\ a') \subset [\![[a'/x](\sigma\phi)]\!]_{\Delta}$ by the inner induction hypothesis, using **R-Pres** to conclude $next(\sigma a) \subset [\![\sigma\phi']\!]_{\Delta}$. For the set $((\lambda x.\sigma a)\ next(a'))$, we use the inner induction hypothesis to conclude that for all $a'' \in next(a')$, we have $((\lambda x.\sigma a)\ a'') \in [\![[a''/x](\sigma\phi)]\!]_{\Delta}$. Applying the induction hypothesis here requires the fact that $a'' \in [\![\sigma\phi']\!]_{\Delta}$, which follows by **R-Pres**. Now we may apply **R-Join** with $a' \sim_{\Delta} a''$ and $\Delta \vdash \texttt{join} : a' = a''$ (which follow from $a' \rightsquigarrow a''$), to obtain $((\lambda x.\sigma a)\ a'') \in [\![[a'/x](\sigma\phi)]\!]_{\Delta}$. This implies $((\lambda x.\sigma a)\ next(a')) \subset [\![[a'/x](\sigma\phi)]\!]_{\Delta}$, as required.

Finally, we wish to conclude $[a'/x]\sigma a \in [\![[a'/x](\sigma\phi')]\!]_{\Delta}$ by instantiating (4) above with $\Delta' \equiv \Delta$; $\sigma' \equiv \sigma$; and $a' \equiv a'$. We have the required $a' \in [\![\sigma\phi']\!]^+_{\Delta}$, of course. But we also need the fact that $(\sigma[a'/x])\phi' = [a'/x](\sigma\phi')$. This holds because $x \notin ran(\sigma)$ (by Lemma 12, as in an earlier case). So we conclude the desired $[a'/x]\sigma a \in [\![[a'/x](\sigma\phi')]\!]_{\Delta}$.

Case:

$$\frac{\Gamma \vdash a : \Pi x:\phi'.\phi \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash (a\ a') : [a'/x]\phi}$$

By the induction hypothesis, we have $\sigma a \in [\![\Pi x:\sigma\phi'.\sigma\phi]\!]_{\Delta}$ and $\sigma a' \in [\![\sigma\phi']\!]_{\Delta}$. By Closability (Lemma 10), we then get $\sigma a' \in [\![\sigma\phi']\!]^+_{\Delta}$. Then using the definition of $[\![\cdot]\!]$ at Π-type, we directly obtain $((\sigma a)\ (\sigma a')) \in [\![[\sigma a'/x]\sigma\phi]\!]_{\Gamma}$. Since $x \notin ran(\sigma)$ (by Lemma 12), we get from this the desired conclusion, namely $\sigma(a\ a') \in [\![\sigma([a'/x]\phi)]\!]_{\Gamma}$.

Case:

$$\frac{}{\Gamma \vdash 0 : \texttt{nat}}$$

Since 0 is a normal form, we have $0 \in SN$ and $\Delta \vdash 0 : \texttt{nat}$, which suffices for this case.

Case:

$$\frac{}{\Gamma \vdash \texttt{nil} : \langle \texttt{vec}\ \phi\ 0\rangle}$$

Since $\texttt{nil}$ is a normal form, we have $\texttt{nil} \in SN$, and of course, $\Delta \vdash \texttt{nil} : \langle \texttt{vec}\ \sigma\phi\ 0 \rangle$. For the second conjunct of the definition of $[\![\cdot]\!]$ at $\texttt{vec}$-type, we have $0 \downarrow 0$. For the third conjunct, assume $\texttt{nil} \rightsquigarrow^* (\texttt{cons}\ a\ a')$ for some a and a'. This is easily shown to be impossible, since reduction cannot possibly turn $\texttt{nil}$ into a $\texttt{cons}$-term.

Case:

$$\frac{\Gamma \vdash a : \texttt{nat}}{\Gamma \vdash (S\ a) : \texttt{nat}}$$

By the induction hypothesis, we have $\sigma a \in [\![\texttt{nat}]\!]_\Delta$, which is equivalent to the conjunction of $\sigma a \in SN$ and $\Delta \vdash \sigma a : \texttt{nat}$. This implies $(S\ \sigma a) \in SN$ and $\Delta \vdash (S\ \sigma a) : \texttt{nat}$, which suffices.

Case:

$$\frac{\begin{array}{l}\Gamma \vdash a'' : \texttt{nat} \\ \Gamma \vdash a : [0/x]\phi \\ \Gamma \vdash a' : \Pi y : \texttt{nat}.\Pi u : [y/x]\phi.[(Sy)/x]\phi\end{array}}{\Gamma \vdash (R_{\texttt{nat}}\ a\ a'\ a'') : [a''/x]\phi}$$

By the induction hypothesis, we have

- $\sigma a'' \in [\![\texttt{nat}]\!]_\Delta$
- $\sigma a \in [\![\sigma[0/x]\phi]\!]_\Delta$
- $\sigma a' \in [\![\Pi y : \texttt{nat}.\Pi u : \sigma([y/x]\phi).\ \sigma([(Sy)/x]\phi)]\!]_\Delta$

We will prove that for any $b \in [\![\texttt{nat}]\!]_\Delta$, and assuming the second two of these facts, we have $(R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ b) \in [\![[b/x]\sigma\phi]\!]_\Delta$. The proof is by inner induction on the measure $\nu(\sigma a) + \nu(\sigma a') + \nu(b) + l(b)$. Our measure is defined, since all the terms involved are reducible and hence strongly normalizing by **R-SN**. By **R-Prog**, it suffices to prove $next(R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ b) \subset [\![[b/x]\sigma\phi]\!]_\Delta$, since the term in question is neutral and appropriately typable. The possibilities for reduction are summarized by:

$$\begin{array}{rcl}(R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ b) & \subset & (R_{\texttt{nat}}\ next(\sigma a)\ (\sigma a')\ b) \cup \\ & & (R_{\texttt{nat}}\ (\sigma a)\ next(\sigma a')\ b) \cup \\ & & (R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ nextb) \cup \\ & & \{(\sigma a) \mid b \equiv 0\} \cup \\ & & \{((\sigma a')\ b'\ (R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ b')) \mid b \equiv (S\ b')\}\end{array}$$

The first three cases are for when the reduction is due to reduction in a subterm. The second two are for when the term in question is itself a redex. For the first two cases, we use the inner induction hypothesis and **R-Pres**. For the third, we do the same, except also apply **R-Join** with $b \sim_\Delta b'$ for $b' \in next(b)$. This ensures that we have $(R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ next(b)) \subset [\![[b/x]\sigma\phi]\!]_\Delta$ (the critical point being that we have b in the type, and not some $b' \in next(b)$). The fourth case follows by our assumption that $\sigma a \in [\![[0/x]\phi]\!]_\Delta$ (note that in this case that the type in question is equivalent to the desired $[b/x]\phi$). For the fifth case, we have $(R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ b') \in [\![[b'/x]\sigma\phi]\!]_\Gamma$ by the inner induction hypothesis, using the fact that $b \in SN$ and $b = (S\ b')$ implies $b' \in SN$; and this then implies $b' \in [\![\texttt{nat}]\!]_\Delta$ by definition of $[\![\cdot]\!]$. Note that we obtain $\Delta \vdash b' : \texttt{nat}$ from $\Delta \vdash (S\ b') : \texttt{nat}$, by applying Simplifying Inversion (Lemma 4 above). By the definition of $[\![\cdot]\!]$ at Π-type and our hypothesis that $\sigma a'$ is reducible at the appropriate Π-type, we have that the given term is in the set $[\![[(S\ b')/x]\phi]\!]_\Delta$, which is equal to the desired $[\![[b/x]\phi]\!]_\Gamma$.

Case:

$$\frac{\begin{array}{l}\Gamma \vdash a : \phi \\ \Gamma \vdash a' : \langle \texttt{vec}\ \phi\ l \rangle\end{array}}{\Gamma \vdash (\texttt{cons}\ a\ a') : \langle \texttt{vec}\ \phi\ (S\ l) \rangle}$$

By the induction hypothesis, we have $\sigma a \in [\![\phi]\!]_\Delta$ and $\sigma a' \in [\![\langle \texttt{vec}\ \sigma\phi\ \sigma l \rangle]\!]_\Gamma$. By **R-SN**, these facts imply $\sigma a \in SN$ and $\sigma a' \in SN$, respectively, and hence $\sigma(\texttt{cons}\ a\ a') \in SN$. We also have $\Delta \vdash \sigma(\texttt{cons}\ a\ a') : \langle \texttt{vec}\ \sigma\phi\ (S\ \sigma l) \rangle$. We must show the conjuncts of the definition of $[\![\cdot]\!]$ at $\texttt{vec}$-type to conclude $(\texttt{cons}\ (\sigma a)\ (\sigma a')) \in [\![\langle \texttt{vec}\ \phi\ (S\ l) \rangle]\!]_\Delta$. The second conjunct is vacuously true, since we cannot have $(\texttt{cons}\ a\ a') \leadsto^* \texttt{nil}$. The third conjunct follows directly from our assumptions.

Case:

$$\frac{\begin{array}{l}\Gamma \vdash a'' : \langle \texttt{vec}\ \phi'\ l \rangle \\ \Gamma \vdash a : [0/y, \texttt{nil}/x]\phi \\ \Gamma \vdash a' : \Pi z : \phi'.\forall l : \texttt{nat}.\Pi v : \langle \texttt{vec}\ \phi'\ l \rangle.\Pi u : [l/y, v/x]\phi. \\ \qquad\quad [(S\ l)/y, (\texttt{cons}\ z\ v)/x]\phi\end{array}}{\Gamma \vdash (R_{\texttt{vec}}\ a\ a'\ a'') : [l/y, a''/x]\phi}$$

This case is similar to that for $R_{\texttt{nat}}$ above, although it is for this case that we have the various clauses of the definition of $[\![\cdot]\!]$ at $\texttt{vec}$-type. By the induction hypothesis, we have

- $\sigma a'' \in [\![\sigma\langle \texttt{vec}\ \phi'\ l \rangle]\!]_\Delta$
- $\sigma a \in [\![\sigma[0/y, \texttt{nil}/x]\phi]\!]_\Delta$
- $\sigma a' \in [\![\sigma\Pi z : \phi'.\forall l : \texttt{nat}.\Pi v : \langle \texttt{vec}\ \phi'\ l \rangle.\Pi u : [l/y, v/x]\phi.[(S\ l)/y, (\texttt{cons}\ z\ v)/x]\phi]\!]_\Delta$

It is sufficient to prove that for any l, for any $b \in [\![\langle \texttt{vec}\ \phi'\ l \rangle]\!]_\Delta$, and assuming the second two of these facts, we have $(R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ b) \in [\![[l/y, b/x]\sigma\phi]\!]_\Delta$. The proof is by inner induction on the measure $\nu(\sigma a) + \nu(\sigma a') + \nu(b) + l(b)$. As above, this measure is defined, by **R-SN**. By **R-Prog**, it suffices to prove $next(R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ b) \subset [\![[l/y, b/x]\sigma\phi]\!]_\Delta$, since the term in question is neutral and appropriately typable. The possibilities for reduction are summarized by:

$$\begin{array}{rcl}(R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ b) & \subset & (R_{\texttt{vec}}\ next(\sigma a)\ (\sigma a')\ b) \cup \\ & & (R_{\texttt{vec}}\ (\sigma a)\ next(\sigma a')\ b) \cup \\ & & (R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ next b) \cup \\ & & \{(\sigma a) \mid b \equiv \texttt{nil}\} \cup \\ & & \{((\sigma a')\ b'\ b''\ (R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ b'')) \mid b \equiv (\texttt{cons}\ b'\ b'')\}\end{array}$$

The first three cases are for when the reduction is due to reduction in a subterm. The second two are for when the term in question is itself a redex. For the first two cases, we use the inner induction hypothesis and **R-Pres**. For the third, we also apply **R-Join** as in the $R_{\texttt{nat}}$ case above, to ensure that we have $(R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ next(b)) \subset [\![[b/x]\sigma\phi]\!]_\Gamma$. The fourth case follows by our assumption that $\sigma a \in [\![[0/y, \texttt{nil}/x]\phi]\!]_\Gamma$. By the definition of $[\![\cdot]\!]$ at $\texttt{vec}$-type, we have $l \sim_\Gamma 0$; so we can apply **R-Join** and the fact that $b = \texttt{nil}$ to obtain $a \in [\![[l/y, b/x]\phi]\!]_\Gamma$, as required.

We now consider the fifth case. First, since $b = (\texttt{cons}\ b'\ b'')$ and $b \in [\![\langle \texttt{vec}\ \phi'\ l \rangle]\!]_\Gamma$, the definition of $[\![\cdot]\!]$ at $\texttt{vec}$-type gives us the following facts for some l':

- $b' \in [\![\phi']\!]_\Delta$
- $b'' \in [\![\langle \texttt{vec}\ \phi'\ l' \rangle]\!]_\Delta$
- $l \sim_\Gamma (S\ l')$

We next apply the inner induction hypothesis to obtain $(R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ b'') \in [\![[l'/y, b''/x]\sigma\phi]\!]_\Delta$; this is legal, since the measure has decreased (in particular, $l(b'') < l(b)$). With this obtained, we use the definition of $[\![\cdot]\!]$ at Π-type and our hypothesis that $\sigma a'$ is reducible at the appropriate Π-type. So we obtain the fact that the given term is in the set $[\![[(S\ l'))/y, (\texttt{cons}\ b'\ b'')/x]\phi]\!]_\Delta$. We can then use **R-Join** with the fourth assumed formula above, and the fact that $b = (\texttt{cons}\ b'\ b'')$, to get that the term is in the desired $[\![[l/y, b/x]\phi]\!]_\Delta$.

D.3 Proof of Weakening Substitutions (Lemma 11)

The proof is by induction on the structure of the assumed derivation of $\sigma \in [\![\Gamma]\!]_\Delta$. The base case is trivial. For the step case, we have:

$$\frac{a \in [\![\sigma'\phi]\!]^+_\Delta \quad \sigma' \in [\![\Gamma']\!]_\Delta}{\sigma' \cup \{(x,a)\} \in [\![\Gamma', x : \phi]\!]_\Delta}$$

The induction hypothesis gives us $\sigma' \in [\![\Gamma']\!]_{\Delta, y:\phi'}$. We just need $a \in [\![\sigma'\phi]\!]^+_{\Delta, y:\phi'}$, and we can reapply the rule to get the desired result. For this, we use the following lemma:

Lemma 13 (Weakening for Closable Terms)**.** *Suppose $\Delta, y : \phi'$ Ok. Then $a \in [\![\phi]\!]^+_\Delta$ implies $a \in [\![\phi]\!]^+_{\Delta, y:\phi'}$.*

Now we apply the following lemma to complete the proof, noting that the variable y of interest here is not in the free variables of a or ϕ, and so the assumption implies the required universal formula:

Lemma 14 (Weakening-Strengthening for Interpretations)**.** *Suppose $a \in [\![\phi]\!]^+_\Delta$. Then we have $a \in [\![\phi]\!]^+_{\Delta, y:\phi'}$ iff for all $a' \in [\![\phi']\!]^+_\Delta$, we have $[a'/y]a \in [\![[a'/y]\phi]\!]^+_\Delta$.*

The proof makes frequent use of the following lemma, which we prove briefly first:

Lemma 15 (Weakening-Strengthening for Ground Joinability)**.** *Suppose $(\Delta, y : \phi')$ Ok. Then we have $a_1 \sim_{\Delta, y:\phi'} a_2$ iff for all $a' \in [\![\phi']\!]_\Delta$, we have $[a'/y]a_1 \sim_\Delta [a'/y]a_2$.*

First, suppose $a_1 \sim_{\Delta, y:\phi'} a_2$, and consider arbitrary $\sigma' \in [\![\Delta]\!]$ and $a' \in [\![\sigma'\phi']\!]$. Then we have $(\sigma' \circ [a'/y]) \in [\![\Delta, y : \phi']\!]$ by Composing Substitutions (Lemma 9). We can then use $a_1 \sim_{\Delta, y:\phi'} a_2$ to get $(\sigma'([a'/y]a_1)) \downarrow (\sigma'([a'/y]a_2$, as required.

Second, suppose that for all $a' \in [\![\phi']\!]_\Delta$, we have $[a'/y]a_1 \sim_\Delta [a'/y]a_2$, and show $a_1 \sim_{\Delta, y:\phi'} a_2$. Assume arbitrary $\sigma \in [\![\Delta, y : \phi']\!]$. Then for some σ' and $a' \in [\![\sigma'\phi]\!]$, we have $\sigma \equiv \sigma'[a'/y]$. Instantiating our assumption with this a' and then σ', we obtain the desired conclusion.

We now turn to the main proof for Weakening-Strengthening, which is by induction on $(|\Gamma|, d(\phi), l(a))$. In all cases, the typing statement in question follows by either Weakening (Lemma 6) or Substitution (Lemma 5), so we omit consideration of typing below. Strong normalization of the substitution instances follows from the definition of closability (and the assumption that a is a closable term in context $\Delta, y : \phi'$).

Case: $\phi \equiv \texttt{nat}$.

This case is trivial.

Case: $\phi \equiv \langle \texttt{vec}\ \phi\ l \rangle$.

These cases follow easily by Weakening-Strengthening for Ground Joinability (Lemma 15) and the induction hypothesis.

Case: $\phi \equiv \Pi x:\psi.\psi'$.

First, assume $a \in [\![\phi]\!]^+_{\Delta,y:\phi'}$, and show $[a'/y]a \in [\![[a'/y]\phi]\!]^+_\Delta$ for an arbitrary $a' \in [\![\phi']\!]^+_\Delta$. It suffices to consider arbitrary $a'' \in [\![[a'/y]\psi]\!]^+_\Delta$, and show $(([a'/y]a)\ a'') \in [\![[a''/x][a'/y]\psi']\!]^+_\Delta$. Since $y \notin FV(a'')$, we certainly have $[\hat{a}/y]a'' \in [\![[\hat{a}/y][a'/y]\psi]\!]^+_\Delta$ for all $\hat{a} \in [\![\phi']\!]^+_\Delta$. So we may apply the induction hypothesis to conclude $a'' \in [\![[a'/y]\psi]\!]^+_{\Delta,y:\phi'}$. Now we may use our assumption of reducibility of a in context $\Delta, y:\phi'$ to conclude $(a\ a'') \in [\![[a''/y]\psi']\!]_{\Delta,y:\phi'}$. To obtain $(a\ a'') \in [\![[a''/y]\psi']\!]^+_{\Delta,y:\phi'}$ from this, assume an arbitrary partition $(\Delta_1,\Delta_2) \equiv (\Delta, y:\phi')$, and arbitrary $\sigma \in [\![\Delta_2]\!]_{\Delta_1}$. We must show $\sigma(a\ a'') \in [\![\sigma[a''/y]\psi']\!]_{\Delta_1}$. If Δ_2 is empty, this statement is equivalent to the fact $(a\ a'') \in [\![[a''/y]\psi']\!]_{\Delta,y:\phi'}$, which we already have. So suppose Δ_2 ends in $y:\phi'$. Then $y \notin ran(\sigma)$. Also, $y \notin FV(a'')$, so our current goal formula is equivalent to $((\sigma a)\ a'') \in [\![[a''/y](\sigma\psi')]\!]_{\Delta_1}$. Instantiating our assumption of closability of a with $\sigma|_{dom(\Delta_1)}$, we obtain $\sigma a \in [\![\sigma\phi]\!]_{\Delta_1}$. This is then sufficient for the desired conclusion, since we easily obtain $\sigma a'' \in [\![\sigma[a'/y]\psi]\!]^+_{\Delta_1}$ from our assumption of closability of a'' in context Δ. Having obtained $(a\ a'')$ closable, we may now apply the induction hypothesis again, to obtain $(([a'/y]a)\ a'') \in [\![[a'/y][a''/x]\psi']\!]_\Delta$, noting again that $y \notin FV(a'')$. Closability of a and a'' again imply closability of this final term.

Now assume that for all $a' \in [\![\phi]\!]^+_\Delta$, we have $[a'/y]a \in [\![[a'/y]\phi]\!]^+_\Delta$; and show $a \in [\![\phi]\!]^+_{\Delta,y:\phi'}$. It suffices to consider arbitrary $a'' \in [\![\psi]\!]^+_{\Delta,y:\phi'}$, and show $(a\ a'') \in [\![[a''/x]\psi']\!]_{\Delta,y:\phi'}$. By the induction hypothesis, we have $[a'/y]a'' \in [\![[a'/y]\psi]\!]^+_\Delta$ for any $a' \in [\![\phi']\!]_\Delta$. Consider arbitrary such a'. We have $([a'/y]a\ [a'/y]a'') \in [\![[[a'/y]a''/x][a'/y]\psi']\!]_\Delta$ by reducibility of $[a'/y]a$ in context Δ. Closability of a and a'' in context $(\Delta, y:\phi')$ again imply closability of this term. This is true for any a', so we may apply the induction hypothesis again to conclude $(a\ a'') \in [\![[a''/x]\psi']\!]_\Delta$, and again obtain closability as above, for the required conclusion.

Case: $\phi \equiv \forall x:\psi.\psi'$.

This case is very similar to the previous one, so we omit it.

Case: $a_1 = a_2$.

This follows by Weakening-Strengthening for Ground Joinability (Lemma 15).

E Proof of Corollaries of Theorem 3

E.1 Proof of Strong Normalization (Corollary 5)

By Soundness for Interpretations, we have $\sigma a \in [\![\sigma\phi]\!]_\Delta$ for all Δ and σ with $\Delta \subset \sigma\Gamma$ and $\sigma \in [\![\Gamma]\!]_\Delta$. We instantiate this by taking Γ for Δ and the identity substitution id on $dom(\Gamma)$ for σ. We have $id \in [\![\Gamma]\!]_\Gamma$, since for all $x \in dom(\Gamma)$, we have $x \in [\![\Gamma(x)]\!]^+_\Gamma$ by **R-Prog** and the fact that if $\sigma' \in [\![\Gamma]\!]$, then by that assumption, we get $\sigma' x \in [\![\sigma'\Gamma(x)]\!]$, which is needed for closability of x. This instantiation yields $a \in [\![\phi]\!]_\Gamma$, which implies $a \in SN$ by **R-SN**.

$$\frac{}{\phi \longmapsto \phi} \qquad \text{when } \phi \text{ is } \alpha, \texttt{nat}, \text{or } a_1 = a_2$$

$$\frac{\phi \longmapsto \phi'}{\langle \texttt{vec}\ \phi\ l\rangle \longmapsto \langle \texttt{vec}\ \phi'\ l\rangle} \qquad \frac{\begin{array}{c}\phi_1 \longmapsto \phi_1' \\ \phi_2 \longmapsto \phi_2'\end{array}}{\Pi x:\phi_1.\phi_2 \longmapsto \Pi x:\phi_1'.\phi_2'} \qquad \frac{\begin{array}{c}\phi_1 \longmapsto \phi_1' \\ \phi_2 \longmapsto \phi_2'\end{array}}{\forall x:\phi_1.\phi_2 \longmapsto \forall x:\phi_1'.\phi_2'}$$

$$\frac{\begin{array}{c}\phi_1 \longmapsto \phi_1' \\ \phi_2 \longmapsto \phi_2'\end{array}}{R\ b\ \phi_1\ (\alpha.\phi_2) \longmapsto (R\ b\ \phi_1'\ (\alpha.\phi_2'))}\ \texttt{rwrR}-\texttt{cong} \qquad \frac{\begin{array}{c}b \leadsto^* (S\ n) \\ \phi_1 \longmapsto \phi_1' \\ \phi_2 \longmapsto \phi_2'\end{array}}{R\ b\ \phi_1\ (\alpha.\phi_2) \longmapsto [R\ n\ \phi_1'\ (\alpha.\phi_2')/\alpha]\phi_2'}\ \texttt{rwrR}-\texttt{beta2}$$

$$\frac{\begin{array}{c}b \leadsto^* 0 \\ \phi_1 \longmapsto \phi_1'\end{array}}{R\ b\ \phi_1\ (\alpha.\phi_2) \longmapsto \phi_1'}\ \texttt{rwrR}-\texttt{beta1}$$

$$\begin{aligned}
\alpha^* &= \alpha \\
\texttt{nat}^* &= \texttt{nat} \\
(a_1 = a_2)^* &= (a_1 = a_2) \\
\langle \texttt{vec}\ \phi\ l\rangle^* &= \langle \texttt{vec}\ \phi^*\ l\rangle \\
(\Pi x:\phi_1.\phi_2)^* &= \Pi x:\phi_1^*.\phi_2^* \\
(\forall x:\phi_1.\phi_2)^* &= \forall x:\phi_1^*.\phi_2^* \\
(R\ b\ \phi_1\ (\alpha.\phi_2))^* &= \begin{cases} [R\ n\ \phi_1^*\ (\alpha.\phi_2^*)/\alpha]\phi_2^* & \text{if } b \leadsto^* (S\ n) \text{ for some } n \\ \phi_1^* & \text{if } b \leadsto^* 0 \\ (R\ b\ \phi_1^*\ (\alpha.\phi_2^*)) & \text{otherwise} \end{cases}
\end{aligned}$$

Figure 9: Full definition of $\phi \longmapsto \phi'$, and complete development ϕ^*.

E.2 Proof of Equational Soundness (Theorem 3)

By Type Preservation, Progress, and Canonical Forms, we obtain $a \leadsto^*$ `join`. By Inversion, the only possible derivations are `(conv)` inferences starting with a `join`-introduction. This implies $b_1 \downarrow b_2$, because joinability is closed under substitution of joinable terms. An easy corollary is:

F Proofs for section 5 (Large Eliminations)

F.1 Proof of lemma 1 (Existence and uniqueness of $\triangledown\phi$)

We define type rewriting (unfolding of recursive types) and complete development as in figure 9. Note that the function $(\cdot)^*$ is well-defined by confluence of $\leadsto^*$.

Showing that $\longmapsto^*$ is confluent is routine[10]:

Lemma 16. *If $\phi \longmapsto \phi'$ and $\psi \longmapsto \psi'$, then $[\psi/\alpha]\phi \longmapsto [\psi'/\alpha]\phi'$*

Proof. Easy induction. □

Lemma 17. *For all types ϕ, we have $\phi \longmapsto \phi$ and $\phi \longmapsto \phi^*$.*

Proof. Easy inductions. $\square$

Lemma 18 (Single-step diamond property)**.** *If $\phi \longmapsto \phi'$, then $\phi' \longmapsto \phi^*$.*

Proof. By induction on structure of ϕ.

- If ϕ is α, $\mathtt{nat}$, or $(a_1 = a_2)$, then $phi' = \phi^* = \phi$.
- If ϕ is $\langle \mathtt{vec}\ \phi\ l \rangle$, $(\Pi x : \phi_1 . \phi_2)$, or $(\forall x : \phi_1 . \phi_2)$, this follows directly by IH and lemma 17.
- If ϕ is $R\ b\ \phi_1\ (\alpha.\phi_2)$, there are three cases, namely $b \not\rightsquigarrow^* n$, $b \rightsquigarrow^* (S\ n')$, or $b \rightsquigarrow^* 0$. If $b \not\rightsquigarrow^* n$, the only rule that could have applied is `rwrR-cong`, so we have $\phi' = (R\ b\ \phi_1'\ (\alpha.\phi_2'))$. By IH $\phi_1' \rightsquigarrow \phi_1^*$ and $\phi_2 \rightsquigarrow \phi_2^*$, so we conclude by reapplying `rwrR-cong`.

  If $b \rightsquigarrow^* (S\ n)$, $\phi^* = [R\ n\ \phi_1^*\ (\alpha.\phi_2^*)/\alpha]\phi_2^*$. By confluence of $\rightsquigarrow^*$ we know $b \not\rightsquigarrow^* 0$, so the two rules that could have applied are `rwrR-cong` and `rwr-beta2`. If the former, $\phi' = [R\ n\ \phi_1'\ (\alpha.\phi_2')/\alpha]\phi_2'$ and by IH $\phi_1' \longmapsto \phi_1^*$ and $\phi_2 \longmapsto \phi_2^*$, so by `rwrR-cong` and lemma 16, $\phi' \longmapsto \phi^*$. If the latter, we have $\phi' = (R\ b\ \phi_1'\ (\alpha.\phi_2'))$ and $a \rightsquigarrow^* (S\ n)$. By IH $\phi_1' \longmapsto \phi_1^*$ and $\phi_2 \longmapsto \phi_2^*$ and we get $\phi' \longmapsto \phi^*$ by `rwrR-beta2`.

  If $b \rightsquigarrow^* 0$, $\phi^* = \phi_1^*$. By confluence $b \not\rightsquigarrow^* (S\ n)$, so the two rules that could have applied are `rwrR-cong` and `rwr-beta1`. In the first case the desired conclusion follows by IH and `rwr-beta1`, in the second directly by IH.

$\square$

Corollary 7. *$\longmapsto^*$ is confluent.*

Lemma 19 (Weak normalization of types)**.** *For every type ϕ there exists a ϕ' such that $\phi \longmapsto^* \phi'$ and $\phi' \not\longmapsto$.*

Proof. Define a *redex* to be a subexpression of ϕ of the form $(R\ b\ \phi_1\ (\alpha.\phi_2))$ where $b \rightsquigarrow_v n$ for some n. We claim that the following *inner-most first* reduction strategy will always terminate: in each iteration, pick a redex such that ϕ_1 (if $n = 0$) or ϕ_2 (if $n = S\ n'$) does not contain any redexes, and reduce that. To see that this process terminates, assign to each ϕ as termination measure the multiset of all redexes in ϕ under the multiset ordering[**?**], where the individual redexes are ordered by n. Each step deletes one redex from the multiset, and adds only redexes where n is decreased. $\square$

It follows that every type ϕ has a unique normal form, which we will write $\nabla\phi$.

F.2 Proof of Lemma 2 (Properties of unfold)

Lemma 20. *If $\phi \longmapsto \phi'$, then $\sigma\phi \longmapsto \sigma\phi'$.*

Proof. Induction on $\phi \longmapsto \phi'$. The congruence rules follow immediately by IH, so the only interesting case is

$$\frac{\begin{array}{c} b \rightsquigarrow^* (S\ n) \\ \phi_1 \longmapsto \phi_1' \\ \phi_2 \longmapsto \phi_2' \end{array}}{R\ b\ \phi_1\ (\alpha.\phi_2) \longmapsto [R\ n\ \phi_1'\ (\alpha.\phi_2')/\alpha]\phi_2'}$$

By substitution for $\rightsquigarrow$ we have $\sigma b \rightsquigarrow^* \sigma(S\ n) = (S\ n)$. The conclusion then follows directly by IH. $\square$

Lemma 21 (Properties of type unfolding).

- *If $a \rightsquigarrow^* n$, then $\triangledown(R\ a\ \phi\ (\alpha.\phi')) = \triangledown(R\ n\ \phi\ (\alpha.\phi'))$.*
- *If $\triangledown\phi_1 = \triangledown\phi_2$, then $\triangledown[\phi_1/\alpha]\psi = \triangledown[\phi_2/\alpha]\psi$.*
- *$\triangledown\langle \texttt{vec}\ \phi\ a\rangle = \langle \texttt{vec}\ \phi\ \triangledown a\rangle$, $\triangledown(\forall x : \phi'.\phi) = \forall x : \triangledown\phi'.\triangledown\phi$, and $\triangledown(\Pi x : \phi'.\phi) = \Pi x : \triangledown\phi'.\triangledown\phi$.*
- *For all σ, ϕ, we have $\triangledown\sigma\triangledown\phi = \triangledown\sigma\phi$.*

Proof. The first property follows by a case-split on whether $n = 0$ or $n = S\ n'$. The second property follows since by lemma 16, the types $[\phi_1/\alpha]\psi$ and $[\phi_2/\alpha]\psi$ both reduce to the common intermediate type $[\triangledown\phi_1/\alpha]\psi$. For the third property, note that by the congruence rules $\langle \texttt{vec}\ \phi\ a\rangle \longmapsto^* \langle \texttt{vec}\ \triangledown\phi\ a\rangle$, and by inspection of the definition of $\longmapsto$, $\langle \texttt{vec}\ \triangledown\phi\ a\rangle \not\longmapsto$.

For the fourth property, by lemma 20 and induction on $\phi \longmapsto^* \triangledown\phi$ we get $\sigma\phi \longmapsto^* \sigma\triangledown\phi$. But we also know $\sigma\phi \longmapsto^* \triangledown\sigma\phi$, and $\triangledown\sigma\phi$ is normal, so by confluence $\sigma\triangledown\phi \longmapsto^* \triangledown\sigma\phi$. Therefore $\triangledown\sigma\triangledown\phi = \triangledown\sigma\phi$. □

Lemma 22. *If $a_1 \downarrow a_2$, then for any ϕ there exists a ψ such that $\triangledown[a_1/x]\phi = [a_1/x]\psi$ and $\triangledown[a_2/x]\phi = [a_2/x]\psi$.*

Proof. Intuitively, we create ψ by applying the substitution to the b in all subexpressions $(R\ b\ \phi_1\ (\alpha.\phi_2))$ in ϕ wherever that would create a redex, and then normalize the resulting type.

Formally, it suffices to prove that if $a_1 \downarrow a_2$ and $[a_1/x]\phi \longmapsto \psi$, then there exists a ϕ' such that $\psi = [a_1/x]\phi$ and $[a_2/x]\phi \longmapsto [a_2/x]\phi'$. This is proved by induction on $[a_1/x]\phi \longmapsto \psi$. The proof relies on that fact that if $[a_1/x]b \rightsquigarrow^* n$ and $a_1 \downarrow a_2$, then $[a_2/x]b \rightsquigarrow^* n$. □

F.3 Proof of Lemma 24 (Characterization of $[\![\ ]\!]$)

Lemma 23 (Interpretation and unfolding).

- *For all ϕ, $[\![\phi]\!] = [\![\triangledown\phi]\!]$*
- *For all σ and ϕ, $[\![\triangledown\sigma\phi]\!] = [\![\sigma\triangledown\phi]\!]$.*

Proof. The first property follows by a case-split on whether ϕ steps. If ϕ does not step, $\triangledown\phi = \phi$. If it does step, $[\![\phi]\!] = [\![\triangledown\phi]\!]$ by the definition of $[\![\ ]\!]$.

For the second property, note that by the first property $[\![\sigma\triangledown\phi]\!] = [\![\triangledown\sigma\triangledown\phi]\!]$. But by lemma 21, $\triangledown\sigma\triangledown\phi = \triangledown\sigma\phi$. □

Lemma 24 (Interpretation of value types). *For all value types ϕ (not just types such that $\phi \not\longmapsto$), the equivalences in figure 8 hold.*

Proof. If ϕ is $\texttt{nat}$ or $a_1 = a_2$, this is immediate. If $\phi = \Pi x : \phi_1.\phi_2$ we get,

$$
\begin{array}{rlr}
a \in [\![\Pi x : \phi_1.\phi_2]\!] & \Leftrightarrow a \in [\![\triangledown\Pi x : \phi_1.\phi_2]\!] & \text{lemma 23} \\
& \Leftrightarrow a \in [\![\Pi x : \triangledown\phi_1.\triangledown\phi_2]\!] & \text{lemma 21} \\
& \Leftrightarrow \begin{array}{l}\exists a'.a \rightsquigarrow_v^* (\lambda x.a') \\ \wedge\ \forall a' \in [\![\triangledown\phi']\!].\ (a\ a') \in [\![[a'/x]\triangledown\phi]\!]\end{array} & \text{Since } (\Pi x : \triangledown\phi_1.\triangledown\phi_2) \not\longmapsto. \\
& \Leftrightarrow \begin{array}{l}\exists a'.a \rightsquigarrow_v^* (\lambda x.a') \\ \wedge\ \forall a' \in [\![\triangledown\phi']\!].\ (a\ a') \in [\![\triangledown[a'/x]\phi]\!]\end{array} & \text{lemma 23} \\
& \Leftrightarrow \begin{array}{l}\exists a'.a \rightsquigarrow_v^* (\lambda x.a') \\ \wedge\ \forall a' \in [\![\phi']\!].\ (a\ a') \in [\![[a'/x]\phi]\!]\end{array} & \text{lemma 23}
\end{array}
$$

The cases $\phi = \forall x : \phi_1.\phi_2$ and $\phi = \langle \texttt{vec}\ \phi_1\ l\rangle$ are similar. □

F.4 Proof of Critical Properties

R-Canon. If $a \in [\![\phi]\!]$, then $a \leadsto_v^* v$ for some v. Furthermore, if ϕ is a value type (i.e. $\mathtt{nat}$, Π, $\forall$, $=$, or $\mathtt{vec}$), then v is the corresponding introduction form.

Proof. Immediate from lemma 24 and the definition of $[\![\ ]\!]$. □

R-Pres. If $a \in [\![\phi]\!]$ and $a \leadsto_v a'$, then $a' \in [\![\phi]\!]$.

Proof. Using that $[\![\phi]\!] = [\![\triangledown\phi]\!]$ we can assume without loss of generality that $\phi \not\longmapsto$. We proceed by induction on the depth of ϕ.

The clauses of the form $a \leadsto_v^*$ are all proven in the same way: for instance if $a \leadsto_v^* n$ and $a \leadsto_v a'$, then $a' \leadsto_v^* n$ by determinacy of $\leadsto_v$. This takes care of all cases except Π and $\forall$.

For $\Pi y : \phi'.\phi$, we also need to show $\forall a'' \in [\![\phi']\!].\ (a'\ a'') \in [\![[a''/x]\phi]\!]$. Let $a'' \in [\![\phi']\!]$. By assumption we know $(aa'') \in [\![[a''/x]\phi]\!]$. But $(aa'') \leadsto_v (a'a'')$, so by the IH at the type $[a''/x]\phi$ (which is of lower depth) $(a'a'') \in [\![[a''/x]\phi]\!]$ as required.

The case $\forall y : \phi'.\phi$ is similar to the above case: we need to show $(a') \in [\![[a''/x]\phi]\!]$ and use that $(a) \leadsto_v (a')$. □

R-Prog. If $a \leadsto_v a'$, and $a' \in [\![\phi]\!]$, then $a \in [\![\phi]\!]$.

Proof. Using that $[\![\phi]\!] = [\![\triangledown\phi]\!]$ we can assume without loss of generality that $\phi \not\longmapsto$. We proceed by induction on the depth of ϕ.

The clauses of the form $a \leadsto_v^* v$ are all proven in the same way: for instance if $a' \leadsto_v^* n$ and $a \leadsto_v a'$, then $a' \leadsto_v^* n$. This takes care of all cases except Π and $\forall$.

For $\Pi y : \phi'.\phi$, we also need to show $\forall a'' \in [\![\phi']\!].\ (a\ a'') \in [\![[a''/x]\phi]\!]$. Let $a'' \in [\![\phi']\!]$. By assumption $(a'a'') \in [\![[a''/x]\phi]\!]$. But $(aa'') \leadsto_v (a'a'')$, so by IH at the type $[a''/x]\phi$ (which is of lower depth), $(aa'') \in [\![[a''/x]\phi]\!]$ as required.

The case $\forall y : \phi'.\phi$ is similar to the above case: we need to show $(a) \in [\![[a''/x]\phi]\!]$ and use $(a) \leadsto_v (a')$. □

R-Join. If $a_1 \downarrow a_2$, then $a \in [\![[a_1/x]\phi]\!]$ implies $a \in [\![[a_2/x]\phi]\!]$.

Proof. Using $[\![\phi]\!] = [\![\triangledown\phi]\!]$ and lemma 22, we can assume without loss of generality that $[a_1/x]\phi \not\longmapsto$. We proceed by induction on the depth of ϕ.

- $\mathtt{nat}$. Trivially true since $[a_1/x]\mathtt{nat} = [a_2/x]\mathtt{nat} = \mathtt{nat}$.
- $\langle \mathtt{vec}\ \phi l\rangle$. By assumption $a \in [\![\langle \mathtt{vec}\ \phi l]\!]$, so either $a \leadsto_v^* \mathtt{nil}$ and $[a_1/x]l \leadsto^* 0$, or $a \leadsto_v^* (\mathtt{cons}\ v\ v')$ and $[a_1/x]l \leadsto^* (S\ n)$ with $v \in [\![[a_1/x]\phi]\!]$ and $v' \in [\![\langle\ [a_1/x]\phi\ n\rangle]\!]$.

  In the first case, note that joinability implies $[a_2/x]l \leadsto^* 0$. In the second case, joinability gives $[a_2/x] \leadsto^* (S\ n)$, and the IH gives $v \in [\![[a_2/x]\phi]\!]$ and $v' \in [\![\langle \mathtt{vec}\ [a_2/x]\phi\ n\rangle]\!]$.
- $\Pi y : \phi'.\phi$. The first conjunct is the same for both $[a_1/x]\phi$ and $[a_2/x]\phi$. For the second conjunct, let $a' \in [\![[a_2/x]\phi']\!]$. By IH $a' \in [\![[a_1/x]\phi']\!]$, so $(a\ a') \in [\![[a'/y][a_1/x]\phi]\!]$. Since y was a bound variable we can choose it such that $a' \notin \mathrm{FV}(a_1) \cup \{x\}$, so $[a'/y][a_1/x]\phi = [a_1/x][a'/y]\phi$. By IH applied to $[a'/y]\phi$ (which is of lower depth), $(aa') \in [\![[a_2/x][a'/y]\phi]\!]$ as required.
- $\forall y : \phi'.\phi$. Similar to the previous case.
- $b_1 = b_2$. We need to show that $[a_1/x]b_1 \downarrow [a_1/x]b_2$ implies $[a_2/x]b_1 \downarrow [a_2/x]b_2$, which is true.

- $(R\ b\ \phi_1\ (\alpha.\phi_2))$. Vacuously true.

$\square$

F.5 Proof of Theorem 6 (Fundamental Lemma for Large Eliminations version of $[\![\ ]\!]$)

Case:

$$\frac{\Gamma(x) \equiv \phi}{\Gamma \vdash x : \phi}$$

Immediate by $\sigma \in [\![\Gamma]\!]$.

Case:

$$\frac{\Gamma \vdash a : \phi \quad \Gamma \vdash a' : \phi' \quad a \downarrow a'}{\Gamma \vdash \texttt{join} : a = a'}$$

$\texttt{join}$ is a value of the right form. We get $\sigma a \downarrow \sigma a'$, since joinability is closed under substitution. We get $\exists v_i.\sigma a_i \leadsto_v^* v_i$ by IH and **R-Canon**.

Case:

$$\frac{\Gamma \vdash a''' : a' = a'' \quad \Gamma \vdash a : [a'/x]\phi \quad x \notin dom(\Gamma)}{\Gamma \vdash a : [a''/x]\phi}$$

By the IH from the second premise we have $\sigma a \in [\![[\sigma a'/x](\sigma\phi)]\!]$. By the IH from the first premise we have $\sigma a''' \in [\![\sigma(a' = a'')]\!]$, so $\sigma a' \downarrow \sigma a''$. So by **R-Join**, $\sigma a \in [\![[\sigma a''/x]/\psi]\!] = [\![[\sigma a''/x](\sigma\phi)]\!]$.

Case:

$$\frac{\Gamma, x : \phi' \vdash a : \phi \quad x \notin FV(a)}{\Gamma \vdash (\lambda a) : \forall x : \phi'.\phi}$$

(λa) is a value of the right form. We must show $(\lambda a) \in [\![\forall x : \sigma\phi'.\sigma\phi]\!]$.

Consider some $a' \in [\![\sigma\phi']\!]$. By **R-Prog**, it suffices to show $a \in [\![[a'/x]\sigma\phi]\!]$, since $((\lambda a)) \leadsto_v a$. Let $\sigma' = \sigma \cup \{(x, a')\}$. Then $\sigma' \in [\![\Gamma, x : \phi']\!]$, so by IH we have $\sigma' a \in [\![\sigma'\phi]\!]$, that is $\sigma a \in [\![[a'/x]\sigma'\phi]\!]$

Case:

$$\frac{\Gamma \vdash a : \forall x : \phi'.\phi \quad \Gamma \vdash a' : \phi'}{\Gamma \vdash (a) : [a'/x]\phi}$$

This follows immediately from induction hypothesis, and the characterization of $[\![\cdot]\!]$ for $\forall$-types (lemma 24).

Case:

$$\frac{\Gamma, x : \phi' \vdash a : \phi}{\Gamma \vdash \lambda x.a : \Pi x : \phi'.\phi}$$

$\lambda x.a$ is a value of the right form. We must show $(\lambda x.a) \in [\![\Pi x : \sigma\phi'.\sigma\phi]\!]$.

Consider some $a' \in [\![\sigma\phi']\!]$. We must show $(\lambda x.\sigma a)a' \in [\![[a'/x]\sigma\phi]\!]$. By **R-Prog** it suffices to show that it steps to a term in $[\![[a'/x]\sigma\phi]\!]$.

By **R-Canon**, $a' \rightsquigarrow v'$ for some v'. We proceed by the number of steps a' takes to normalize. In the base case a' is already a value. Then $(\lambda x.\sigma a)a' \rightsquigarrow_v [a'/x]\sigma a = \sigma' a$ where $\sigma' = \sigma \cup \{(x, a')\}$. $\sigma' \in [\![\Gamma, x : \phi']\!]$, so by IH $\sigma' a \in [\![\sigma'\phi]\!]$.

In the step case, $a' \rightsquigarrow_v a''$ for some a'', so $(\lambda x.\sigma a)a' \rightsquigarrow_v (\lambda x.\sigma a)a''$. By **R-Pres**, $a'' \in [\![\sigma\phi']\!]$, so the inner IH applies and $(\lambda x.\sigma a)a'' \in [\![[a''/x]\sigma\phi']\!]$. But $a' \rightsquigarrow_v a''$, so $a \rightsquigarrow a''$, so $a' \downarrow a''$, so **R-Join** applies and $(\lambda x.\sigma a)a'' \in [\![[a'/x]\sigma\phi']\!]$ as required.

Case:

$$\frac{\Gamma \vdash a : \Pi x : \phi'.\phi \qquad \Gamma \vdash a' : \phi'}{\Gamma \vdash (a\ a') : [a'/x]\phi}$$

This follows immediately from induction hypothesis, and the characterization of $[\![\cdot]\!]$ for Π-types (lemma 24).

Case:

$$\frac{}{\Gamma \vdash 0 : \mathtt{nat}}$$

0 is a value of the right form.

Case:

$$\frac{\Gamma \vdash a : \mathtt{nat}}{\Gamma \vdash (S\ a) : \mathtt{nat}}$$

By the induction hypothesis, we have $\sigma a \in [\![\mathtt{nat}]\!]$, so by **R-Canon** $\sigma a \rightsquigarrow_v^* n$. Then $(S\ \sigma a) \rightsquigarrow_v^* (Sn)$, which is a value of the right form.

Case:

$$\frac{\begin{array}{l}\Gamma \vdash a'' : \mathtt{nat} \\ \Gamma \vdash a : [0/x]\phi \\ \Gamma \vdash a' : \Pi y : \mathtt{nat}.\Pi u : [y/x]\phi.[(Sy)/x]\phi\end{array}}{\Gamma \vdash (R_{\mathtt{nat}}\ a\ a'\ a'') : [a''/x]\phi}$$

By the induction hypothesis, we have

- $\sigma a'' \in [\![\mathtt{nat}]\!]$
- $\sigma a \in [\![\sigma[0/x]\phi]\!]$
- $\sigma a' \in [\![\Pi y : \mathtt{nat}.\Pi u : \sigma([y/x]\phi).\ \sigma([(Sy)/x]\phi)]\!]$

We will prove that for any $b \in [\![\texttt{nat}]\!]$, and assuming the second two of these facts, we have $(R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ b) \in [\![[b/x]\sigma\phi]\!]$. The proof is by inner induction on the measure $\nu(\sigma a) + \nu(\sigma a') + \nu(b) + l(b)$. Our measure is defined, since all the terms involved are normalizing by **R-Canon**.

By **R-Prog**, it suffices to prove that $R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ b$ steps to a term in $[\![[b/x]\sigma\phi]\!]$. The terms (σa), $(\sigma a')$, and b are all in $[\![\cdot]\!]$, so by **R-Canon** each of them either steps or is a value. By considering the cases, one of the following must be the case:

$$\begin{array}{llll}
R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ b & \rightsquigarrow_v & R_{\texttt{nat}}\ c\ (\sigma a')\ b & \text{where } (\sigma a) \rightsquigarrow_v c \\
R_{\texttt{nat}}\ v\ (\sigma a')\ b & \rightsquigarrow_v & R_{\texttt{nat}}\ v\ c'\ b & \text{where } (\sigma a') \rightsquigarrow_v c' \\
R_{\texttt{nat}}\ v\ v'\ b & \rightsquigarrow_v & R_{\texttt{nat}}\ v\ v'\ c & \text{where } b \rightsquigarrow_v c \\
R_{\texttt{nat}}\ v\ v'\ 0 & \rightsquigarrow_v & v & \\
R_{\texttt{nat}}\ v\ v'\ (S\ n) & \rightsquigarrow_v & v'\ n\ (R_{\texttt{nat}}\ v\ v'\ n) &
\end{array}$$

The first three cases are for when the reduction is due to reduction in a subterm. The second two are for when the term in question is itself a redex. For the first two cases, we use the inner induction hypothesis and **R-Pres**. For the third, we do the same, except also apply **R-Join** with $b \downarrow c$. This ensures that we have $(R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ b) \in [\![[b/x]\sigma\phi]\!]$ (the critical point being that we have b in the type, and not c) The fourth case follows by our assumption that $\sigma a \in [\![[0/x]\phi]\!]$ (note that in this case that the type in question is equivalent to the desired $[b/x]\phi$). For the fifth case, we have $(R_{\texttt{nat}}\ (\sigma a)\ (\sigma a')\ n) \in [\![[n/x]\sigma\phi]\!]$ by the inner induction hypothesis. Since n is a number we trivially have $n \in [\![\texttt{nat}]\!]$. By the definition of $[\![\cdot]\!]$ at Π-type and our hypothesis that $\sigma a'$ is reducible at the appropriate Π-type, we have that the given term is in the set $[\![[(S\ n)/x]\phi]\!]$, which is equal to the desired $[\![[b/x]\phi]\!]_\Gamma$.

Case:

$$\frac{}{\Gamma \vdash \texttt{nil} : \langle \texttt{vec}\ \phi\ 0 \rangle}$$

$\texttt{nil}$ and 0 are values of the right form.

Case:

$$\frac{\begin{array}{l}\Gamma \vdash a : \phi \\ \Gamma \vdash a' : \langle \texttt{vec}\ \phi\ l \rangle\end{array}}{\Gamma \vdash (\texttt{cons}\ a\ a') : \langle \texttt{vec}\ \phi\ (S\ l) \rangle}$$

We prove the second disjunct of $\sigma(\texttt{cons}\ a\ a') \in [\![\sigma\langle \texttt{vec}\ \phi\ (S\ l)\rangle]\!]$. By IH and **R-Canon**, we know σa and $\sigma a'$ reduce to some values v and v'. Then $\sigma(\texttt{cons}\ a\ a') \rightsquigarrow_v^* (\texttt{cons}\ v\ v')$ as required. Similarly from the IH we know $\sigma l \rightsquigarrow_v^* n$, so $\sigma(S\ l) \rightsquigarrow_v^* (S\ n)$ as required.

Case:

$$\frac{\begin{array}{l}\Gamma \vdash a'' : \langle \texttt{vec}\ \phi'\ l \rangle \\ \Gamma \vdash a : [0/y, \texttt{nil}/x]\phi \\ \Gamma \vdash a' : \Pi z : \phi'.\forall l : \texttt{nat}.\Pi v : \langle \texttt{vec}\ \phi'\ l \rangle.\Pi u : [l/y, v/x]\phi. \\ \qquad [(S\ l)/y, (\texttt{cons}\ z\ v)/x]\phi\end{array}}{\Gamma \vdash (R_{\texttt{vec}}\ a\ a'\ a'') : [l/y, a''/x]\phi}$$

This case is similar to that for $R_{\texttt{nat}}$ above. By the induction hypothesis, we have

- $\sigma a'' \in [\![\sigma\langle \texttt{vec}\ \phi'\ l\rangle]\!]$
- $\sigma a \in [\![\sigma[0/y, \texttt{nil}/x]\phi]\!]$
- $\sigma a' \in [\![\sigma \Pi z : \phi'.\forall l : \texttt{nat}.\Pi v : \langle \texttt{vec}\ \phi'\ l\rangle.\Pi u : [l/y, v/x]\phi.[(S\ l)/y, (\texttt{cons}\ z\ v)/x]\phi]\!]$

It is sufficient to prove that for any l, for any $b \in [\![\langle \texttt{vec}\ \phi'\ l\rangle]\!]$, and assuming the second two of these facts, we have $(R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ b) \in [\![[l/y, b/x]\sigma\phi]\!]$. The proof is by inner induction on the measure $\nu(\sigma a) + \nu(\sigma a') + \nu(b) + l(b)$. As above, this measure is defined, by **R-Canon**.

By **R-Prog**, it suffices to prove that $R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ b$ steps to a term in $[\![[l/y, b/x]\sigma\phi]\!]$. The terms (σa), $(\sigma a')$, and b are all in $[\![\cdot]\!]$, so by **R-Canon** each of them either steps or is a value. By considering the cases, one of the following must be the case:

$$\begin{array}{llll}
R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ b & \rightsquigarrow_v & R_{\texttt{vec}}\ c\ (\sigma a')\ b & \text{where } (\sigma a) \rightsquigarrow_v c \\
R_{\texttt{vec}}\ v\ (\sigma a')\ b & \rightsquigarrow_v & R_{\texttt{vec}}\ v\ c'\ b & \text{where } (\sigma a') \rightsquigarrow_v c' \\
R_{\texttt{vec}}\ v\ v'\ b & \rightsquigarrow_v & R_{\texttt{vec}}\ v\ v'\ c & \text{where } b \rightsquigarrow_v c \\
R_{\texttt{vec}}\ v\ v'\ \texttt{nil} & \rightsquigarrow_v & v & \\
R_{\texttt{vec}}\ v\ v'\ (\texttt{cons}\ u\ u') & \rightsquigarrow_v & v'\ u\ u'\ (R_{\texttt{vec}}\ v\ v'\ u') &
\end{array}$$

The first three cases are for when the reduction is due to reduction in a subterm. The second two are for when the term in question is itself a redex. For the first two cases, we use the inner induction hypothesis and **R-Pres**. For the third, we also apply **R-Join** as in the $R_{\texttt{nat}}$ case above, to ensure that we have $(R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ c) \in [\![[l/y, b/x]\sigma\phi]\!]$. The fourth case follows by our assumption that $\sigma a \in [\![[0/y, \texttt{nil}/x]\phi]\!]$. By the definition of $[\![\cdot]\!]$ at $\texttt{vec}$-type, we must have $l \rightsquigarrow_v^* 0$; so we can apply **R-Join** and the fact that $b = \texttt{nil}$ to obtain $a \in [\![[l/y, b/x]\phi]\!]$, as required.

For the fifth case, we know by assumption that $(\texttt{cons}\ u\ u') \in [\![\sigma\langle \texttt{vec}\ \phi\ l\rangle]\!]$. By the definition of $[\![\cdot]\!]$ that means that $u \in [\![\phi]\!]$, $\sigma l \rightsquigarrow_v^* (S\ n)$, and $u' \in [\![\langle \texttt{vec}\ \phi\ n\rangle]\!]$.

Then we have $(R_{\texttt{vec}}\ (\sigma a)\ (\sigma a')\ u') \in [\![[n/y, u'/x]\sigma\phi]\!]$ by the inner induction hypothesis. By the definition of $[\![\cdot]\!]$ at Π-type and our hypothesis that $\sigma a'$ is reducible at the appropriate Π-type, we have that the given term is in the set $[\![[(S\ n)/y, (\texttt{cons}\ u\ u')/x]\sigma\phi]\!]$. By using **R-Join** on $l \downarrow (S\ n)$, this implies the desired $[\![[l/y, (\texttt{cons}\ u\ u')/x]\phi]\!]$.

Case:

$$\frac{\Gamma \vdash a : [R\ a'\phi\ (\alpha.\phi')/\alpha]\phi' \quad \Gamma \vdash a' : nat}{\Gamma \vdash a : R\ (Sa')\ \phi\ (\alpha.\phi')}$$

By IH we have $\sigma a' \in [\![\texttt{nat}]\!]$, so by **R-Canon**, $\sigma a' \rightsquigarrow_v^* n$ for some n. Also by IH,

$$\begin{array}{rll}
\sigma a & \in [\![[R\ \sigma a'\ \sigma\phi\ (\alpha.\sigma\phi')/\alpha]\sigma\phi']\!] & \\
& = [\![\nabla[R\ \sigma a'\ \sigma\phi\ (\alpha.\sigma\phi')/\alpha]\phi']\!] & \text{lemma 23} \\
& = [\![\nabla[R\ n\ \sigma\phi\ (\alpha.\sigma\phi')/\alpha]\sigma\phi']\!] & \text{lemma 21}
\end{array}$$

At the same time,

$$(R\ (S\sigma a')\ \sigma\phi\ (\alpha.\sigma\phi')) \longmapsto ([R\ n\ \sigma\phi\ (\alpha.\sigma\phi')/\alpha]\sigma\phi'),$$

so

$$\nabla(R\ (S\sigma a')\ \sigma\phi\ (\alpha.\sigma\phi')) = \nabla([R\ n\ \sigma\phi\ (\alpha.\sigma\phi')/\alpha]\sigma\phi').$$

So

$$\begin{aligned}[\![R\ (S\sigma a')\ \sigma\phi\ (\alpha.\sigma\phi')]\!] &= [\![\nabla(R\ (S\sigma a')\ \sigma\phi\ (\alpha.\sigma\phi'))]\!] \\ &= [\![\nabla([R\ n\ \sigma\phi\ (\alpha.\sigma\phi')/\alpha]\sigma\phi')]\!]\end{aligned}$$

Case:

$$\frac{\Gamma \vdash a : [R\ a'\phi\ (\alpha.\phi')/\alpha]\phi' \quad \Gamma \vdash a' : \mathtt{nat}}{\Gamma \vdash a : R\ (Sa')\ \phi\ (\alpha.\phi')}\ \mathtt{foldS}$$

Similar to the previous case.

Case:

$$\frac{\Gamma \vdash a : \phi}{\Gamma \vdash a : R\ 0\ \phi\ (\alpha.\phi')}$$

Similar to `unfoldS` case.

Case:

$$\frac{\Gamma \vdash a : R\ 0\ \phi\ (\alpha.\phi')}{\Gamma \vdash a : \phi}$$

Similar to `foldS` case.