Exploring the Reach of Hereditary Substitution

Harley Eades[1] **and Aaron Stump**[2]

1 **The University of Iowa, Computer Science**
14 MacLean Hall Iowa City, IA 52242-1419, USA
`harley-eades@uiowa.edu`

2 **The University of Iowa, Computer Science**
14 MacLean Hall Iowa City, IA 52242-1419, USA
`aaron-stump@uiowa.edu`

Abstract

This paper investigates the reach of the proof technique of normalization by hereditary substitution due to K. Watkins et al., by showing how it can be applied to three example type systems with advanced features. The first two systems are extensions of Stratified System F (SSF), a type theory of predicative polymorphism studied by D. Leivant. We extend SSF first with sum types and commuting conversions, and then with types expressing equalities between terms. For the third system, we consider an extension of the Simply Typed Lambda Calculus (STLC) with types expressing equalities between types. We show how normalization by hereditary substitution can be applied to these different advanced typing features, and identify several new properties resulting from defining the hereditary substitution function on open terms.

1 Introduction

The Tait-Girard reducibility method is a well-known technique for proving normalization of simply typed and polymorphic lambda calculi; see [10] for an excellent discussion of the reducibility method. This technique is, however, very intricate, which makes it difficult to apply to new theories. Therefore an easier technique is of interest to the research community. The hereditary-substitutions method is qualitatively simpler than proof by reducibility. For example, the proof of the type soundness theorem using reducibility requires a quantification over substitutions satisfying the typing context, but no quantification over substitutions is required for normalization by hereditary substitution. The hereditary-substitutions method is also (at least nominally) proof-theoretically less complex than proof by reducibility. The central concept of reducibility is defined by recursion on types, introducing new quantifiers in alternating polarity. To formalize the argument, one thus needs a logic in which quantified formulas can be defined by recursion. In contrast, the central definitions for the hereditary-substitutions method can be expressed without such rich constructions, and thus should be formalizable in much weaker theories. These advantages suggest that proof by hereditary substitution may be easier to apply to new theories, and simpler to formalize, than proof by reducibility. The drawback is that it is unclear if the method can scale to richer type theories. This paper contributes to understanding which type theories can be shown to be normalizing using this method.

Overview of the method. There are six essential steps that need to be taken in order to conclude normalization by the hereditary substitution proof method. First, an ordering on the types of the type theory under consideration must be defined. Following the ordering on types the hereditary substitution function must be defined. Then it must be shown that the hereditary substitution function satisfies several properties: i. totality and type preservation, ii. normality preservation, and iii. soundness with respect to reduction. The first two properties of the hereditary substitution function are well-known, but the third to our knowledge is novel. The third property is a safety check insuring that the hereditary subtitution function is obeying the reduction strategy of the system it is defined for. This will become more clear when we actually state the results for each system below. In fact we will formally define each of these results for the analyzed type theories below. The next major part of proof by hereditary substitution is defining a semantics for the types of the type theory under consideration. We call this semantics the interpretation of types. The interpretations of types are essentially sets of terms with a common type in a context. In order to conclude type soundness using our semantics for types we must prove a vital result called the main substitution lemma which shows that the interpretations of types are closed under hereditary substitutions.

Related and previous work. The central idea of hereditary substitution is essentially to prove normalization using a lexicographic combination of an ordering on the types and the strict subepxression ordering on proofs. This central idea has been used in normalization proofs dating all the way back to Prawitz in 1965 where he shows normalizaion for natural deduction [18] (Chapter 4). It arises again in Lévy's work in [15]. A proof of normalization of the STLC using this idea can be found in Chapter Four of [10]. A similar proof can be found in [4] (Chapter 2 Theorem 2.2.9). Yet another proof can be found in [12] for STLC. In fact F. Joachimski and R. Matthes actually show weak and strong normalization for STLC using a lexicographic combination of the strict subexpression ordering on types and proofs. They then extend this to show strong normalization for STLC extended with sum types and commuting conversions. Furthermore, he extends his method again to show strong normalization of Gödel's System T. For each system they also define a normalization function on normal terms which amounts to a function similar to hereditary substitution. The constructive content of these proofs was first made explicit by K. Watkins et al. in [21] and R. Adams in [3] for dependent types.

A. Abel in 2006 shows how to implement a normalizer using sized heterogeneous types which is a function similar to the hereditary substitution function in [1]. He then uses hereditary substitution to prove normalization of the type level of type theory with higher-order subtyping in [2]. This results in a purely syntactic metathoery of his type theory. C. Keller and T. Altenkirch recently implemented hereditary substitution as a normalization function for the simply typed λ-calculus in Agda [13]. Their results show that hereditary substitution can be used to decide $\beta\eta$-equality. They found hereditary substitution to be convenient to use in a total Type Theory, because it can be implemented without a termination proof. This is because the hereditary-substitution function can be recognized as structurally recursive, and hence accepted directly by Agda's termination checker.

In previous work, we showed how to apply the hereditary-substitutions method to prove normalization of Stratified System F (SSF), a type theory of predicative polymorphism studied by Leivant [9, 14]. However, the results here are considerably different. In our previous work we used the central idea of the hereditary substitution function implicitly in the proof of the main substitution lemma. In the current paper, we use the hereditary substitution explicitly in the statement of the main substitution lemma and prove several necessary properties of the hereditary substitution function as stated above. We consider this change to be the correct way of using the hereditary substitution function in normalization proofs. We also consider more complicated type systems than the system analyzed in the previous paper, with advanced features that are important in other lambda calculi. We study three systems featuring, respectively, commuting conversions for sum types (Section 2), explicit equalities between terms (Section 3), and explicit equalities between types (Section 4). These systems pose novel challenges for hereditary substitutions. For example, the third of these systems is normalizing only for reduction strategies which do not reduce beneath lambda abstractions. We devise non-trivial adaptations of the hereditary-substitutions method to prove normalization for each of these systems.

Why Hereditary Substitution? To summarize, hereditary substitution is a proof technique for showing normalization of typed lambda calculi. It promises to be easier to use than the reducibility proof method, which is well known to be quite intricate and challenging to adapt to new theories,

Syntax:

$$\begin{array}{lcl}
K & := & *_0 \mid *_1 \mid \ldots \\
T & := & X \mid T \to T \mid \forall X : K.T \mid T + T \\
t & := & x \mid \lambda x : T.t \mid t\,t \mid \Lambda X : K.t \mid t[T] \mid \mathit{inl}(t) \mid \mathit{inr}(t) \mid \text{case } t \text{ of } x.t,x.t
\end{array}$$

Reduction Rules:

$$\begin{array}{lcllcl}
(\Lambda X : *_p.t)[\phi] & \rightsquigarrow & [\phi/X]t & \text{case } \mathit{inl}(t) \text{ of } x.t_1,x.t_2 & \rightsquigarrow & [t/x]t_1 \\
(\lambda x : \phi.t)t' & \rightsquigarrow & [t'/x]t & \text{case } \mathit{inr}(t) \text{ of } x.t_1,x.t_2 & \rightsquigarrow & [t/x]t_2
\end{array}$$

Commuting Conversions:

$$\begin{array}{lcl}
(\text{case } t \text{ of } x.t_1,x.t_2)\ t' & \rightsquigarrow & \text{case } t \text{ of } x.(t_1\ t'),x.(t_2\ t') \\
\text{case } (\text{case } t \text{ of } x.t_1,x.t_2) \text{ of } y.s_1,y.s_2 & \rightsquigarrow & \text{case } t \text{ of} \\
 & & \quad x.(\text{case } t_1 \text{ of } y.s_1,y.s_2), \\
 & & \quad x.(\text{case } t_1 \text{ of } y.s_1,y.s_2)
\end{array}$$

Figure 1 Syntax, Reduction Rules, Commuting Conversions for SSF$^+$

such as ones being developed for dependently typed programming languages like ATS [6], Epigram [16], and others.

2 Stratified System F$^+$ (SSF$^+$)

Stratified System F$^+$ is a predicative polymorphic type theory. Stratified polymorphism is used in predicative type theories for universe hierarchies. SSF$^+$ also has sum types $\phi_1 + \phi_2$, whose elimination form *case t of* $x_1.t_1, x_2.t_2$ is used to case split on a whether or not term t with a sum type is truly x_1 of type ϕ_1, or else x_2 of type ϕ_2. We consider sum types with so-called commuting conversions, which allow independent cases to be permuted past each other (see Fig 1 below). Commuting conversions are well-known to pose technical difficulties for normalization proofs based on reducibility (see [20] and Chapter 10 of [10]). We will see that they can be handled straightforwardly with hereditary substitution.

The syntax, reduction rules, and commuting conversions for SSF$^+$ can be found in Fig. 1. This extension of SSF is based on the version of SSF used in [9]. The kind-assignment rules are defined in Fig. 3 and the type-assignment rules in defined in Fig. 4. The kinding/typing relations depend on well-formed contexts which are defined in Fig. 2. To ensure substitutions over contexts behave in an expected manner, we rename variables as necessary to ensure contexts have at most one declaration per variable. Lastly, throughout this paper we use various basic meta-theoretic results for each system. Due to space constraints we do not show them here, but in the companion report [8]. The reader will also find all omitted proofs in the companion report.

$$\frac{}{\cdot\ \mathit{Ok}} \qquad \frac{\Gamma\ \mathit{Ok}}{\Gamma, X : *_p\ \mathit{Ok}} \qquad \frac{\Gamma \vdash \phi : *_p \qquad \Gamma\ \mathit{Ok}}{\Gamma, x : \phi\ \mathit{Ok}}$$

Figure 2 Well-formedness of Contexts for SSF$^+$

2.1 Ordering on Types

In this section we define an ordering on types. This ordering is curcial for the hereditary substitution method. We will see in Section 2.3 that we prove several properties of the hereditary substitution

$$\frac{\Gamma \vdash \phi_1 : *_p \qquad \Gamma \vdash \phi_2 : *_q}{\Gamma \vdash \phi_1 \to \phi_2 : *_{max(p,q)}} \qquad \frac{\Gamma, X : *_q \vdash \phi : *_p}{\Gamma \vdash \forall X : *_q.\phi : *_{max(p,q)+1}} \qquad \frac{\Gamma \vdash \phi_1 : *_p \qquad \Gamma \vdash \phi_2 : *_q}{\Gamma \vdash \phi_1 + \phi_2 : *_{max(p,q)}} \qquad \frac{\begin{array}{l}\Gamma(X) = *_p \\ \Gamma\ Ok \qquad p \leq q\end{array}}{\Gamma \vdash X : *_q}$$

Figure 3 SSF$^+$ Kinding Rules

$$\frac{\begin{array}{l}\Gamma(x) = \phi \\ \Gamma\ Ok\end{array}}{\Gamma \vdash x : \phi} \qquad \frac{\Gamma, x : \phi_1 \vdash t : \phi_2}{\Gamma \vdash \lambda x : \phi_1.t : \phi_1 \to \phi_2} \qquad \frac{\begin{array}{l}\Gamma \vdash t_1 : \phi_1 \to \phi_2 \\ \Gamma \vdash t_2 : \phi_1\end{array}}{\Gamma \vdash t_1 t_2 : \phi_2} \qquad \frac{\Gamma, X : *_l \vdash t : \phi}{\Gamma \vdash \Lambda X : *_l.t : \forall X : *_l.\phi}$$

$$\frac{\begin{array}{l}\Gamma \vdash t : \forall X : *_l.\phi_1 \\ \Gamma \vdash \phi_2 : *_l\end{array}}{\Gamma \vdash t[\phi_2] : [\phi_2/X]\phi_1} \qquad \frac{\begin{array}{l}\Gamma \vdash t : \phi_1 \\ \Gamma \vdash \phi_2 : *_p\end{array}}{\Gamma \vdash inl(t) : \phi_1 + \phi_2} \qquad \frac{\begin{array}{l}\Gamma \vdash t : \phi_2 \\ \Gamma \vdash \phi_1 : *_p\end{array}}{\Gamma \vdash inr(t) : \phi_1 + \phi_2} \qquad \frac{\begin{array}{l}\Gamma \vdash t : \phi_1 + \phi_2 \\ \Gamma, x : \phi_1 \vdash t_1 : \psi \\ \Gamma, x : \phi_2 \vdash t_2 : \psi\end{array}}{\Gamma \vdash \text{case } t \text{ of } x.t_1,x.t_2 : \psi}$$

Figure 4 SSF$^+$ Type-Assignment Rules

function defined in Section 2.2. The ordering used in these proofs is the lexicographic ordering consisting of the the ordering we are about to define and the strict subexpression ordering on terms. In fact if no ordering on types exists then one cannot prove the necessary properties of the hereditary substitution function needed to conclude normalization.

▶ **Definition 1.** The ordering $>_\Gamma$ is defined as the least relation satisfying the universal closures of the following formulas:

$$\begin{array}{lll} \phi_1 \to \phi_2 & >_\Gamma & \phi_1 \\ \phi_1 \to \phi_2 & >_\Gamma & \phi_2 \end{array} \qquad\qquad \begin{array}{lll} \phi_1 + \phi_2 & >_\Gamma & \phi_1 \\ \phi_1 + \phi_2 & >_\Gamma & \phi_2 \\ \forall X : *_l.\phi & >_\Gamma & [\phi'/X]\phi \text{ where } \Gamma \vdash \phi' : *_l. \end{array}$$

▶ **Theorem 2** (Well-Founded Ordering)**.** *The ordering $>_\Gamma$ is well-founded on types ϕ such that $\Gamma \vdash \phi : *_l$ for some l.*

We need transitivity in a number of places in the proof of the main substitution lemma.

▶ **Lemma 3** (Transitivity of $>_\Gamma$)**.** *Let ϕ, ϕ', and ϕ'' be kindable types. If $\phi >_\Gamma \phi'$ and $\phi' >_\Gamma \phi''$ then $\phi >_\Gamma \phi''$.*

2.2 The Hereditary Substitution Function

We will work through the hereditary substitution method in more detail for this first example system than for the other two. The difference between ordinary capture avoiding substitution and hereditary substitution is if as a result of substitution a new redex is created, then that redex is recursively reduced. The hereditary substitution function is denoted $[t/x]^{\phi}t'$ where ϕ is called the cut type, due to the relation of the hereditary substitution function and cut elimination, and t' is called the principle term of substitution. The purpose of the cut type is that it is used to ensure that the definition of the function is well founded. We will see this connection below. We now turn to defining this function for SSF$^+$.

The definition of the hereditary substitution function for SSF$^+$ is in Fig. 5 and Fig. 6. First, one should read this definition as a mutually recursive function in terms of the hereditary substitution function $[t/x]^{\phi}t'$, the application reduction function $app_{\phi}\ t_1\ t_2$, and case construct reduction function $rcase_{\phi}\ t_0\ x\ t_1\ t_2$. The definitions of all these functions depend on a partial function called $ctype_{\phi}(x,t)$ which returns the type of a term t if that term is in weak-head normal form. This function is equivalent to the $treduce$ function used in [21]. A brief explanation of the definition of the hereditary substitution function is the following. One should read the "where"-conditions below some of the cases of the definitions of each function as preconditions. The definition of $ctype_{\phi}$ is straightforward. Now the definition of the hereditary substitution function has the form one would expect for a substitution function. In fact the only difference between the definition of the hereditary substitution function and capture avoiding substitution is really present in the cases for applications, type instantiation, and case constructs. These are the locations where new redexes can be created by substitution.

Consider application. Based on the definition of the reduction rules we have two forms of redexes: either a β-redex or a commuting coversion where a case construct is applied to an argument. In order to create a new β-redex as a result of substitution we must be substituting into a term of the form $x\ t_1 \cdots t_n$. The hereditary substitution function detects this by using the $ctype_{\phi}$ function. We will show that if this function is defined then the head of the input term must be a variable x thus implying that the input term is of the form we expect. Next we apply the hereditary substitution to the head of the application and if that results in a λ-abstraction then we know that a new redex will be created by substitution. Thus, we recursively reduce this redex using the hereditary substitution function. We will see that the result of the $ctype_{\phi}$ function is in fact a subexpression of the cut type ϕ. This tells us that the cut type in the case for creation of a new β-redex has reduced in the recursive call to the hereditary substitution function.

Now in the definition of the hereditary substitution function where a new redex in the form of the commuting conversion is created – in this case a case construct is applied to an argument – we again know by the $ctype_{\phi}$ function that the head of the application is in the form $x\,t_1 \cdots t_2$. Furthermore, we know applying the hereditary substitution function to the head of the application results in a case construct. So we recursively reduce the created redex in the same way the reduction rules do, but when we push the argument into the branches of the resulting case construct more redexes may be created. So to handle recursively reducing all of the newly created redexes in the branches we call the application reduction function app_{ϕ}. This function reduces redexes by recursively calling itself and the hereditary substitution function. The remaining cases where new redexes are potentially created are similar to these cases. The function $rcase_{\phi}$ handles reducing case constructs.

2.3 Properties of The Hereditary Substitution Function

We now turn to proving several properites of the hereditary substitution function. Since the various functions involved in the definition of the hereditary substitution function including the hereditary substitution function itself depends on the $ctype_{\phi}$ function we first establish its main properties. The major property of this function is that its output is a subexpression of the cut type ϕ. This is stated in part one of the next lemma. Part two is a sanity check which shows that the type returned by $ctype_{\phi}$ is the right type. In other words it is the type of the second argument of the function. The remaining parts of the lemma are used in the proofs of the other properties of the hereditary substitution function. Recall in certain parts of the definition of the hereditary substitution function it must be the case that $ctype_{\phi}$ is defined so the remaining parts of the properties lemma ensure this is the case.

▶ **Lemma 4** (Properties of $ctype_{\phi}$)**.**

i. *If $ctype_{\phi}(x,t) = \phi'$ then $head(t) = x$ and ϕ' is a subexpression of ϕ.*

ii. *If $\Gamma, x : \phi, \Gamma' \vdash t : \phi'$ and $ctype_{\phi}(x,t) = \phi''$ then $\phi' \equiv \phi''$.*

$ctype_\phi(x, x) = \phi$

$ctype_\phi(x, t_1\ t_2) = \phi''$
   Where $type_\phi(x, t_1) = \phi' \to \phi''$.

$app_\phi\ t_1\ t_2 = t_1\ t_2$
   Where t_1 is not a λ-abstraction or a case construct.

$app_\phi\ (\lambda x : \phi'.t_1)\ t_2 = [t_2/x]^{\phi'} t_1$

app_ϕ (case t_0 of $x.t_1$,$x.t_2$) t = case t_0 of $x.(app_\phi\ t_1\ t)$,$x.(app_\phi\ t_2\ t)$

$rcase_\phi\ t_0\ y\ t_1\ t_2$ = case t_0 of $y.t_1$,$y.t_2$
   Where t_0 is not an inject-left or an inject-right term or a case construct.

$rcase_\phi\ inl(t')\ y\ t_1\ t_2 = [t'/y]^{\phi_1}\ t_1$

$rcase_\phi\ inr(t')\ y\ t_1\ t_2 = [t'/y]^{\phi_2}\ t_2$

$rcase_\phi$ (case t'_0 of $x.t'_1$,$x.t'_2$) $y\ t_1\ t_2$ = case t'_0 of $x.(rcase_\phi\ t'_1\ y\ t_1\ t_2)$,$x.(rcase_\phi\ t'_2\ y\ t_1\ t_2)$

Figure 5 Hereditary Substitution Function for Stratified System F$^+$

iii. *If $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$, $\Gamma \vdash t : \phi$, $[t/x]^\phi t_1 = \lambda y : \phi_1.q$, and t_1 is not then there exists a type ψ such that $ctype_\phi(x, t_1) = \psi$.*

iv. *If $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$, $\Gamma \vdash t : \phi$, $[t/x]^\phi t_1$ = case t'_0 of $y.t'_1$,$y.t'_2$, and t_1 is not then there exists a type ψ such that $ctype_\phi(x, t_1) = \psi$.*

v. *If $\Gamma, x : \phi, \Gamma' \vdash$ case t_0 of $y.t_1$,$y.t_2 : \phi'$, $\Gamma \vdash t : \phi$, $[t/x]^\phi t_0$ = case t'_0 of $z.t'_1$,$z.t'_2$, and t_0 is not then there exists a type ψ such that $ctype_\phi(x, t_0) = \psi$.*

vi. *If $\Gamma, x : \phi, \Gamma' \vdash$ case t_0 of $y.t_1$,$y.t_2 : \phi'$, $\Gamma \vdash t : \phi$, $[t/x]^\phi t_0 = inl(t')$, and t_0 is not then there exists a type ψ such that $ctype_\phi(x, t_0) = \psi$.*

vii. *If $\Gamma, x : \phi, \Gamma' \vdash$ case t_0 of $y.t_1$,$y.t_2 : \phi'$, $\Gamma \vdash t : \phi$, $[t/x]^\phi t_0 = inr(t')$, and t_0 is not then there exists a type ψ such that $ctype_\phi(x, t_0) = \psi$.*

We now move on to proving the main properties of the hereditary substitution function. First, we show that for typeable terms it is a total function and the output maintains the same type as the principle term of substitution.

▶ **Lemma 5** (Total and Type Preserving). *Suppose $\Gamma \vdash t : \phi$ and $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$. Then there exists a term t'' such that $[t/x]^\phi t' = t''$ and $\Gamma, \Gamma' \vdash t'' : \phi'$.*

The next result we show is that the hereditary substitution function cannot create new redexes. The following example shows when a particular redex is destroyed by hereditary substitution.

▶ **Example 6.** Let $t \equiv inl(a)$ for some variable a and $t' \equiv$ case (case x of $y.y$,$y.y$) of $z.z$,$z.z$ where $\Gamma \vdash t : (\phi_1 + \phi_2) + \phi$, $\Gamma \vdash a : \phi_1 + \phi_2$, and $\Gamma, x : (\phi_1 + \phi_2) + \phi \vdash t' : \phi'$. Now lets trace the definition of the hereditary substitution function on the input t, x, and t' and compute what the image of $[t/x]^{(\phi_1+\phi_2)+\phi} t'$ will be. Well the first thing the hereditary substitution function does is apply itself to all the parts of the case construct. So we must calculute the following results:

i. First we have to compute $[t/x]^{(\phi_1+\phi_2)+\phi}$(case x of $y.y$,$y.y$). This requires us to compute $[t/x]^{(\phi_1+\phi_2)+\phi} x = t$. Now the hereditary substitution functions checks to see if t is an inject-left term or an inject-right term, if it is then we have created a new redex. It happens that t is so we know $[t/x]^{(\phi_1+\phi_2)+\phi}$(case x of $y.y$,$y.y$) $= [a/y]^{\phi_1+\phi_2} y = a$.

ii. Second we must compute $[t/x]^{(\phi_1+\phi_2)+\phi} y = y$.

Now putting these pieces together we obtain $[t/x]^{(\phi_1+\phi_2)+\phi} t'$ = case a of $z.z$,$z.z$. Clearly, we no longer have the right-hand side of the commuting conversion for case constructs.

$[t/x]^{\phi}x = t$
$[t/x]^{\phi}y = y$
  Where y is a variable distinct from x.
$[t/x]^{\phi}(\lambda y : \phi'.t') = \lambda y : \phi'.([t/x]^{\phi}t')$
$[t/x]^{\phi}(\Lambda X : *_l.t') = \Lambda X : *_l.([t/x]^{\phi}t')$
$[t/x]^{\phi}inr(t') = inr([t/x]^{\phi}t')$
$[t/x]^{\phi}inl(t') = inl([t/x]^{\phi}t')$
$[t/x]^{\phi}(t_1\ t_2) = ([t/x]^{\phi}t_1)\ ([t/x]^{\phi}t_2)$
  Where $([t/x]^{\phi}t_1)$ is not a λ-abstraction or a case construct, or both $([t/x]^{\phi}t_1)$ and t_1 are λ-abstractions or case constructs, or $ctype_{\phi}(x,t_1)$ is undefined.
$[t/x]^{\phi}(t_1\ t_2) = [([t/x]^{\phi}t_2)/y]^{\phi''}s'_1$
  Where $([t/x]^{\phi}t_1) = \lambda y : \phi''.s'_1$ for some y, s'_1, and ϕ'' and $ctype_{\phi}(x,t_1) = \phi'' \to \phi'$.
$[t/x]^{\phi}(t_1\ t_2) = \text{case } w \text{ of } y.(app_{\phi}\ r\ ([t/x]^{\phi}t_2)),y.(app_{\phi}\ s\ ([t/x]^{\phi}t_2))$
  Where $[t/x]^{\phi}t_1 = \text{case } w \text{ of } y.r,y.s$ for some terms w, r, s and variable y, and $ctype_{\phi}(x,t_1) = \phi'' \to \phi'$.
$[t/x]^{\phi}(t'[\phi']) = ([t/x]^{\phi}t')[\phi']$
  Where $[t/x]^{\phi}t'$ is not a type abstraction or t' and $[t/x]^{\phi}t'$ are type abstractions.
$[t/x]^{\phi}(t'[\phi']) = [\phi'/X]s'_1$
  Where $[t/x]^{\phi}t' = \Lambda X : *_l.s'_1$, for some X, s'_1 and $\Gamma \vdash \phi' : *_q$, such that, $q \leq l$ and t' is not a type abstraction.
$[t/x]^{\phi}(\text{case } t_0 \text{ of } y.t_1,y.t_2) = \text{case } ([t/x]^{\phi}t_0) \text{ of } y.([t/x]^{\phi}t_1),y.([t/x]^{\phi}t_2)$
  Where $([t/x]^{\phi}t_0)$ is not an inject-left or an inject-right term or a case construct, or $([t/x]^{\phi}t_0)$ and t_0 are both inject-left or inject-right terms or case constructs, or $ctype_{\phi}(x,t_0)$ is undefined.
$[t/x]^{\phi}(\text{case } t_0 \text{ of } y.t_1,y.t_2) = rcase_{\phi}\ ([t/x]^{\phi}t_0)\ y\ ([t/x]^{\phi}t_1)\ ([t/x]^{\phi}t_2)$
  Where $([t/x]^{\phi}t_0)$ is an inject-left or an inject-right term or a case construct and $ctype_{\phi}(x,t_0) = \phi_1 + \phi_2$.

Figure 6 Hereditary Substitution Function for Stratified System F^{+} Continued

Showing redex preservation for the hereditary substitution function depends on the following function which contructs the set of redexes in a term.

▶ **Definition 7.** The following function constructs the set of redexes within a term:

$$
\begin{array}{lll}
\multicolumn{3}{l}{rset(x) = \emptyset} \\
\multicolumn{3}{l}{rset(\lambda x : \phi.t) = rset(t)} \\
\multicolumn{3}{l}{rset(\Lambda X : *_l.t) = rset(t)} \\
\multicolumn{3}{l}{rset(t_1\ t_2)} \\
= & rset(t_1,t_2) & \text{if } t_1 \text{ is not a } \lambda\text{-abstraction.} \\
= & \{t_1\ t_2\} \cup rset(t'_1,t_2) & \text{if } t_1 \equiv \lambda x : \phi.t'_1. \\
\multicolumn{3}{l}{rset(t''[\phi''])} \\
= & rset(t'') & \text{if } t'' \text{ is not a type absraction.} \\
= & \{t''[\phi'']\} \cup rset(t''') & \text{if } t'' \equiv \Lambda X : *_l.t'''. \\
\multicolumn{3}{l}{rset(inl(t)) = rset(t)} \\
\multicolumn{3}{l}{rset(inr(t)) = rset(t)} \\
\multicolumn{3}{l}{rset(\text{case } t_0 \text{ of } x.t_1,x.t_2)} \\
= & rset(t_0) \cup rset(t_1,t_2) & \text{if } t_1 \text{ is not an inject-left term or an inject-right term.} \\
= & \{\text{case } t_0 \text{ of } x.t_1,x.t_2\} \cup rset(t_0) \cup rset(t_1,t_2) & \text{if } t_1 \text{ is an inject-left term or an inject-right term.}
\end{array}
$$

The extention of $rset$ to multiple arguments is defined as follows:

$$rset(t_1, \ldots, t_n) =^{def} rset(t_1) \cup \cdots \cup rset(t_n).$$

▶ **Lemma 8** (Redex Preserving). *If* $\Gamma \vdash t : \phi$, $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$, *and* t' *then* $|rset(t', t)| \geq |rset([t/x]^{\phi} t')|$.

It is easy to see that the previous lemma can be used to show that if the input to the hereditary substitution function contains no redexes then the result will contain no redexes. That is, the function is normality preserving. This is the key result when using the hereditary substitution function in normalization proofs.

▶ **Lemma 9** (Normality Preserving). *If* $\Gamma \vdash n : \phi$ *and* $\Gamma, x : \phi' \vdash n' : \phi'$ *then there exists a normal term* n'' *such that* $[n/x]^{\phi} n' = n''$.

The final property of the hereditary substitution that needs to be proven is a safety property. Which is that the hereditary substitution function does not deviate from the reduction rules. We call this soundness with respect to reduction. We will see this property come into play in the proof of type soundness.

▶ **Lemma 10** (Soundness with Respect to Reduction). *If* $\Gamma \vdash t : \phi$ *and* $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ *then* $[t/x]t' \rightsquigarrow^* [t/x]^{\phi} t'$.

At this point we are ready to move on to concluding normalization.

2.4 Concluding Normalization

We now define the interpretation $[\![\phi]\!]_\Gamma$ of types ϕ in typing context Γ.

▶ **Definition 11.** The interpretation of types $[\![\phi]\!]$ is defined by:

$$n \in [\![\phi]\!]_\Gamma \iff \Gamma \vdash n : \phi$$

We extend this definition to non-normal terms t in the following way:

$$t \in [\![\phi]\!]_\Gamma \iff \exists n.t \rightsquigarrow^! n \in [\![\phi]\!]_\Gamma$$

We define $t \rightsquigarrow^! t'$ to be $t \rightsquigarrow^* t'$ and t' is normal.

In the introduction we defined semantic inversion and asserted that it must hold with respect to any interpretation of types used in a proof by hereditary substitution. It is easy to see that syntactic inversion holds for every form of the SSF$^+$ typing relation trivially. This fact yields semantic inversion by definition of the interpretation of types. In this paper we will freely use syntactic and semantic inversion lemmas without explicit reference.

Before moving on to proving soundness of typing and concluding normalization we need a basic result about the interpretation of types: type substitution. It is used in the proof of the type soundness theorem (Theorem 14).

▶ **Lemma 12** (Type Substitution for the Interpretation of Types). *If* $n \in [\![\phi']\!]_{\Gamma, X:*_l, \Gamma'}$ *and* $\Gamma \vdash \phi : *_l$ *then* $[\phi/X]n \in [\![[\phi/X]\phi']\!]_{\Gamma, [\phi/X]\Gamma'}$.

Substitution for the interpretation of types is as we have been calling it the main substitution lemma. It is a crucial result, because it is needed in the proof of type soundness and it depends on the hereditary substitution substitution function.

$$\begin{array}{lll} t & := & x \mid \lambda x : \phi.t \mid t\,t \mid \Lambda X : K.t \mid t[\phi] \mid join \\ \phi & := & X \mid \Pi x : \phi.\phi \mid \forall X : K.\phi \mid t = t \\ K & := & *_0 \mid *_1 \mid \ldots \end{array}$$

$$\begin{array}{lll} (\Lambda X : *_p.t)[\phi] & \rightsquigarrow & [\phi/X]t \\ (\lambda x : \phi.t)t' & \rightsquigarrow & [t'/x]t \end{array}$$

Figure 7 Syntax of Terms, Types, and Kinds and Reduction Rules for DSSF$^=$

$$\frac{\Gamma(X) = *_p \quad \Gamma\ Ok \quad p \leq q}{\Gamma \vdash X : *_q} \qquad \frac{\Gamma \vdash \phi_1 : *_p \quad \Gamma, x : \phi_1 \vdash \phi_2 : *_q}{\Gamma \vdash \Pi x : \phi_1.\phi_2 : *_{max(p,q)}} \qquad \frac{\Gamma, X : *_q \vdash \phi : *_p}{\Gamma \vdash \forall X : *_q.\phi : *_{max(p,q)+1}} \qquad \frac{\Gamma \vdash t_1 : \phi \quad \Gamma \vdash t_2 : \phi \quad \Gamma \vdash \phi : *_p}{\Gamma \vdash t_1 = t_2 : *_p}$$

Figure 8 DSSF$^=$ Kinding Rules

▶ **Lemma 13** (Hereditary Substitution for the Interpretation of Types). *If* $n' \in [\![\phi']\!]_{\Gamma,x:\phi,\Gamma'}$, $n \in [\![\phi]\!]_\Gamma$, *then* $[n/x]^\phi n' \in [\![\phi']\!]_{\Gamma,\Gamma'}$.

Proof. By Lemma 5 we know there exists a term $\hat{n}$ such that $[n/x]^\phi n' = \hat{n}$ and $\Gamma, \Gamma' \vdash \hat{n} : \phi'$ and by Lemma 9 $\hat{n}$ is normal. Therefore, $[n/x]^\phi n' = \hat{n} \in [\![\phi']\!]_{\Gamma,\Gamma'}$. ◀

We are now ready to present our main result.

▶ **Theorem 14** (Type Soundness). *If* $\Gamma \vdash t : \phi$ *then* $t \in [\![\phi]\!]_\Gamma$.

▶ **Corollary 15** (Normalization). *If* $\Gamma \vdash t : \phi$ *then* $t \rightsquigarrow^! n$.

3 Dependent Stratified System F$^=$ (DSSF$^=$)

DSSF$^=$ is SSF extended with dependent types and equations between terms. Equations between terms are an important concept in Martin-Löf type theory [11, 17], and play a central role also in dependently typed programming languages, such as the second author's GURU language [19]. The syntax and reduction rules are defined in Fig. 7. The kind-assignment rules are defined in Fig. 8. One thing to note regarding the kind-assignment rules is that the level of an equation is the same level as the type of the terms in the equation. The terms used in an equation must have the same type. Finally, the type-assignment rules are defined in Fig. 9. Note that $t_1 \downarrow t_2$ denotes there exists a term t such that $t_1 \rightsquigarrow^* t$ and $t_2 \rightsquigarrow^* t$.

3.1 Syntactic Inversion

This section covers syntactic inversion of the typing relation. In Sect. 3.3 we define the interpretation of types and semantic inversion must hold for this definition. Since the interpretation of types is simply the restriction of the typing relation to normal forms syntactic inversion implies semantic inversion. Hence, we only need to show syntactic inversion. Syntactic inversion depends on a judgment called type syntactic equality. It is defined in Fig. 10. We show that syntactic conversion holds for syntactic type equality in Lemma 16. We only state the syntactic inversion lemma for the typing relation, because syntactic inversion for the kinding relation is trivial. Note that we use syntactic inversion for kinding throughout the paper without indication. First we define some convenient syntax used in the statement of the following lemma. We write $\exists(a_1, a_2, \ldots, a_n)$ for

$$\frac{\Gamma\ Ok \quad \Gamma(x) = \phi}{\Gamma \vdash x : \phi} \qquad \frac{\Gamma, x : \phi_1 \vdash t : \phi_2}{\Gamma \vdash \lambda x : \phi_1.t : \Pi x : \phi_1.\phi_2} \qquad \frac{\Gamma \vdash t_2 : \phi_1 \quad \Gamma \vdash t_1 : \Pi x : \phi_1.\phi_2}{\Gamma \vdash t_1\ t_2 : [t_2/x]\phi_2}$$

$$\frac{\Gamma, X : *_l \vdash t : \phi}{\Gamma \vdash \Lambda X : *_l.t : \forall X : *_l.\phi} \qquad \frac{\Gamma \vdash t : \forall X : *_l.\phi_1 \quad \Gamma \vdash \phi_2 : *_l}{\Gamma \vdash t[\phi_2] : [\phi_2/X]\phi_1} \qquad \frac{t_1 \downarrow t_2 \quad \Gamma\ Ok \quad \Gamma \vdash t_1 : \phi \quad \Gamma \vdash t_2 : \phi}{\Gamma \vdash join : t_1 = t_2}$$

$$\frac{\Gamma \vdash t_0 : t_1 = t_2 \quad \Gamma \vdash t : [t_1/x]\phi}{\Gamma \vdash t : [t_2/x]\phi}$$

Figure 9 DSSF$^=$ Type-Assignment Rules

$$\frac{\Gamma \vdash p : t_1 = t_2}{\Gamma \vdash [t_1/x]\phi \approx [t_2/x]\phi}\ \text{TE}_{\text{Q}_1} \qquad \frac{\Gamma \vdash [t_1/x]\phi \approx [t_1/x]\phi' \quad \Gamma \vdash p : t_1 = t_2}{\Gamma \vdash [t_1/x]\phi \approx [t_2/x]\phi'}\ \text{TE}_{\text{Q}_2}$$

Figure 10 DSSF$^=$ Type Syntactic Equality

$\exists a_1.\exists a_2 \ldots \exists a_n$. Note that the proofs of the following lemmas depend on a number of other auxiliary lemmas. They can also be found in the companion report.

The first lemma is type syntactic conversion which states that if a term t inhabits a type ϕ then it inhabits all types equivalent to ϕ. Following this is injectivity of Π-types which is needed in the proof of syntactic inversion. Finally, we conclude syntactic inversion.

▶ **Lemma 16** (Type Syntactic Conversion). *If* $\Gamma \vdash t : \phi$ *and* $\Gamma \vdash \phi \approx \phi'$ *then* $\Gamma \vdash t : \phi'$.

▶ **Lemma 17** (Injectivity of Π-Types for Type Equality). *If* $\Gamma \vdash \Pi y : \phi_1.\phi_2 \approx \Pi y : \phi_1'.\phi_2'$ *then* $\Gamma \vdash \phi_1 \approx \phi_1'$ *and* $\Gamma, y : \phi_1 \vdash \phi_2 \approx \phi_2'$.

▶ **Lemma 18** (Syntactic Inversion).

i. *If* $\Gamma \vdash \lambda x : \phi_1.t : \phi$ *then* $\exists \phi_2.\ \Gamma, x : \phi_1 \vdash t : \phi_2 \wedge \Gamma \vdash \Pi x : \phi_1.\phi_2 \approx \phi$.

ii. *If* $\Gamma \vdash t_1\ t_2 : \phi$ *then* $\exists (x, \phi_1, \phi_2)$.
$\Gamma \vdash t_1 : \Pi x : \phi_1.\phi_2 \wedge \Gamma \vdash t_2 : \phi_1 \wedge \Gamma \vdash \phi \approx [t_2/x]\phi_2$.

iii. *If* $\Gamma \vdash \Lambda X : *_l.t : \phi$ *then* $\exists \phi'.\ \Gamma, X : *_l \vdash t : \phi' \wedge \Gamma \vdash \phi \approx \forall X : *_l.\phi'$.

iv. *If* $\Gamma \vdash t[\phi_2] : \phi$ *then* $\exists (\phi_1, \phi_2)$.
$\Gamma \vdash t : \forall X : *_l.\phi_1 \wedge \Gamma \vdash \phi_2 : *_l \wedge \Gamma \vdash \phi \approx [\phi_2/X]\phi_1$.

v. *If* $\Gamma \vdash join : \phi$ *then* $\exists (t_1, t_2, \phi')$.
$t_1 \downarrow t_2 \wedge \Gamma \vdash t_1 : \phi' \wedge \Gamma \vdash t_2 : \phi' \wedge \Gamma \vdash \phi \approx t_1 = t_2 \wedge \Gamma\ Ok$.

3.2 The Ordering on Types and The Hereditary Substitution Function

The ordering on types is defined as follows:

▶ **Definition 19.** The ordering $>_\Gamma$ is defined as the least relation satisfying the universal closure of the following formulas:

$$\begin{array}{lll} \Pi x : \phi_1.\phi_2 & >_\Gamma & \phi_1 \\ \Pi x : \phi_1.\phi_2 & >_\Gamma & [t/x]\phi_2, \text{ where } \Gamma \vdash t : \phi_1. \\ \forall X : *_l.\phi & >_\Gamma & [\phi'/X]\phi, \text{ where } \Gamma \vdash \phi' : *_l. \end{array}$$

$[t/x]^{\phi}x = t$

$[t/x]^{\phi}y = y$
Where y is a variable distinct from x.

$[t/x]^{\phi}join = join$

$[t/x]^{\phi}(\lambda y : \phi'.t') = \lambda y : \phi'.([t/x]^{\phi}t')$

$[t/x]^{\phi}(\Lambda X : *_l.t') = \Lambda X : *_l.([t/x]^{\phi}t')$

$[t/x]^{\phi}(t_1\ t_2) = ([t/x]^{\phi}t_1)\ ([t/x]^{\phi}t_2)$
Where $([t/x]^{\phi}t_1)$ is not a λ-abstraction, $([t/x]^{\phi}t_1)$ and t_1 are λ-abstractions, or $ctype_{\phi}(x, t_1)$ is undefined.

$[t/x]^{\phi}(t_1\ t_2) = [([t/x]^{\phi}t_2)/y]^{\phi''}s'_1$
Where $([t/x]^{\phi}t_1) \equiv \lambda y : \phi''.s'_1$ for some y and s'_1 and t_1 is not a λ-abstraction, and $ctype_{\phi}(x, t_1) = \Pi y : \phi''.\phi'$.

$[t/x]^{\phi}(t'[\phi']) = ([t/x]^{\phi}t')[\phi']$
Where $[t/x]^{\phi}t'$ is not a type abstraction or t' and $[t/x]^{\phi}t'$ are type abstractions.

$[t/x]^{\phi}(t'[\phi']) = [\phi'/X]s'_1$
Where $[t/x]^{\phi}t' \equiv \Lambda X : *_l.s'_1$, for some X, s'_1 and $\Gamma \vdash \phi' : *_q$, such that, $q \le l$ and t' is not a type abstraction.

Figure 11 Hereditary Substitution Function for Stratified System $F^{=}$

Just as we seen before this is the ordering used in the proofs of the properties of the hereditary substitution function which will be defined next. As one might have expected this is a well-founded ordering.

▶ **Theorem 20** (Well-Founded Ordering). *The ordering $>_{\Gamma}$ is well-founded on types ϕ such that $\Gamma \vdash \phi : *_l$ for some l.*

The type-syntactic-equality relation respects this ordering.

▶ **Lemma 21.** *If $\Gamma \vdash \phi' \approx \phi''$ and $\phi >_{\Gamma} \phi'$ then $\phi >_{\Gamma} \phi''$.*

The following lemma is used in the proof of totality of the hereditary substitution function.

▶ **Lemma 22.** *If $\Gamma \vdash \psi \approx \Pi y : \phi_1.\phi_2$ and ψ is a subexpression of ϕ'' then $\phi'' >_{\Gamma} \phi_1$ and $\phi'' >_{\Gamma, y:\phi_1} \phi_2$.*

We now define the hereditary substitution function for DSSF$^{=}$. The function is fully defined in Fig. 11. We do not repeat the definition of $ctype_{\phi}$ for DSSF$^{=}$, because it is exactly the same as the previous system. The definition is rather straight forward. Due to space constraints and the fact that all of the properties of the hereditary substitution function are so similar to the previous type theory they can be found in the companion report.

3.3 Concluding Normalization

We are now ready to conlcude normalization. We will accomplish this by first defining the interpretation of types, then proving the interpretation of types are closed under hereditary substitutions, and finally proving type soundness. The interpretation of types are defined exactly the same way as they were for SSF^{+}. We do not repeat that definition here. Just as in the previous type system semantic inversion must hold for the previous definition. This is implied by syntactic inversion, which we proved in the previous section. The next two lemmas are used in the proof of the type soundness theorem.

$$\begin{array}{rcl} t & ::= & x \mid \lambda x.t \mid t\,t \mid \mathit{join} \\ v & ::= & \lambda x.t \mid \mathit{join} \mid s \\ s & ::= & x \mid s\,v \end{array} \qquad \begin{array}{rcl} \phi & ::= & X \mid \phi \to \phi \mid \phi = \phi \\ \Gamma & ::= & \cdot \mid \Gamma,\, x : \phi \mid \Gamma,\, X \end{array}$$

Figure 12 Syntax of Terms, Types, and Contexts for STLC$^=$

$$\mathcal{E}[(\lambda x.t)\ v] \quad \leadsto \quad \mathcal{E}[[v/x]t] \qquad \textit{where:} \qquad \mathcal{E} \quad ::= \quad * \mid \mathcal{E}\ t \mid v\ \mathcal{E}$$

Figure 13 Call-By-Value Reduction Rules for STLC$^=$

▶ **Corollary 23** (Type Substitution for the Interpretation of Types). *If* $n \in [\![\phi']\!]_{\Gamma,X:*_l,\Gamma'}$ *and* $\Gamma \vdash \phi : *_l$ *then* $[\phi/X]n \in [\![[\phi/X]\phi']\!]_{\Gamma,[\phi/X]\Gamma'}$.

▶ **Lemma 24** (Semantic Equality). *If* $\Gamma \vdash p : t_1 = t_2$ *then* $[\![[t_1/x]\phi]\!]_\Gamma = [\![[t_2/x]\phi]\!]_\Gamma$.

We now conclude type soundness for SSF$^=$, and hence, normalization.

▶ **Lemma 25** (Hereditary Substitution for the Interpretation of Types). *If* $n' \in [\![\phi']\!]_{\Gamma,x:\phi,\Gamma'}$, $n \in [\![\phi]\!]_\Gamma$, *then* $[n/x]^\phi n' \in [\![[n/x]\phi']\!]_{\Gamma,[n/x]\Gamma'}$.

▶ **Theorem 26** (Type Soundness). *If* $\Gamma \vdash t : \phi$ *then* $t \in [\![\phi]\!]_\Gamma$.

▶ **Corollary 27** (Normalization). *If* $\Gamma \vdash t : \phi$ *then* $t \leadsto^! n$.

4 Simply Typed λ-Calculus$^=$ (STLC$^=$)

In the previous section we proved normalization for a system with equations between terms. We now turn to a similar system, but instead of equations between terms we consider equations between types. This feature is important for systems like F_ω or the Calculus of Constructions (and its many variants), where type-level computation and type-level redexes are allowed (see, e.g., [5]). While type-level computation is often accommodated through an implicit conversion relation, if one wishes to use explicit equational reasoning, then equalities between types are required.

The system we examine here is an extension of the Simply Typed λ-Calculus. We omit type quantification to focus solely on equality types, but include type variables to allow formation of non-trivial such equalities. The syntax for terms, types, and contexts is defined in Fig. 12. The kind-assignment and type-assignment rules are defined in Fig. 14 and Fig. 15 respectively. This system has a call-by-value (CBV) reduction strategy defined by the rules in Fig. 13. One of the main reasons we use CBV instead of full β-reduction is that full β-reduction mixed with an inconsistent context allows diverging terms to type check. Consider the following example.

▶ **Example 28.** The following term is a typeable diverging term of STLC$^=$ under full β-reduction:

$$\lambda p.((\lambda x.x\ x)\ (\lambda x.x\ x))$$

We can assign this term the type $(X = X \to X) \to X$ in the context consisting just of X. The intuition is that if we assume that X equals $X \to X$, we can then assign the variable x a type X, which we can then convert with Conv to $X \to X$ to type the self-application x .

We now move on to proving normalization using hereditary substitution.

$$\frac{X \in \Gamma}{\Gamma \vdash X}\ \textsc{KVar} \qquad \frac{\Gamma \vdash \phi_1 \qquad \Gamma \vdash \phi_2}{\Gamma \vdash \phi_1 \to \phi_2}\ \textsc{KArrow} \qquad \frac{\Gamma \vdash \phi_1 \qquad \Gamma \vdash \phi_2}{\Gamma \vdash \phi_1 = \phi_2}\ \textsc{KEq}$$

Figure 14 STLC$^=$ Kinding Rules

$$\frac{\Gamma(x) = \phi}{\Gamma \vdash x : \phi}\ \textsc{Var} \qquad \frac{\Gamma \vdash \phi_1 \qquad \Gamma, x : \phi_1 \vdash t : \phi_2}{\Gamma \vdash \lambda x.t : \phi_1 \to \phi_2}\ \textsc{Lam} \qquad \frac{}{\Gamma \vdash \mathit{join} : T = T}\ \textsc{Join}$$

$$\frac{\Gamma \vdash t_1 : \phi_1 \to \phi_2 \qquad \Gamma \vdash t_2 : \phi_1}{\Gamma \vdash t_1\ t_2 : \phi_2}\ \textsc{App} \qquad \frac{\Gamma \vdash t : [\phi_1/X]\phi \qquad \Gamma \vdash t' : \phi_1 = \phi_2}{\Gamma \vdash t : [\phi_2/X]T}\ \textsc{Conv}$$

Figure 15 STLC$^=$ Type-Assignment Rules

4.1 The Hereditary Substitution Function and Its Propeties

The hereditary substitution function is defined in Fig. 16. It depends on the $ctype_\phi$ function, but we do not redined it here. It is equivalent to the previous definitions when one removes the case for type instantiation. We prove all the same properties of the hereditary substitution function as we did for the two previouse systems. Due to their similiarity to the previous systems we do not show them here. They can be found in the appendix of the companion report.

4.2 Concluding Normalization

Before concluding normalization we first define the interpretation of types.

▶ **Definition 29.** The interpretation of types $[\![\phi]\!]$ is defined as follows:

$$\begin{array}{lcl}
x \in [\![\phi]\!]_\Gamma & \iff & \exists(\phi', p).(\Gamma \vdash p : \phi = \phi' \wedge \Gamma(x) = \phi') \\
\lambda x.t \in [\![\phi]\!]_\Gamma & \iff & \Gamma \vdash \lambda x.t : \phi \wedge \exists(\phi_1, \phi_2, p).(\Gamma \vdash p : \phi = \phi_1 \to \phi_2 \wedge \\
 & & (Con(\Gamma, x : \phi_1) \implies t \in [\![\phi_2]\!]_{\Gamma, x:\phi_1})) \\
\mathit{join} \in [\![\phi]\!]_\Gamma & \iff & \Gamma \vdash \mathit{join} : \phi \wedge \exists(\phi', \phi'', p).(\Gamma \vdash p : \phi = (\phi' = \phi'') \wedge \forall \sigma : \Gamma.\sigma\phi' \equiv \sigma\phi''). \\
s\ v \in [\![\phi]\!]_\Gamma & \iff & \Gamma \vdash s\ v : \phi \wedge \exists \phi'.(s \in [\![\phi' \to \phi]\!]_\Gamma \wedge v \in [\![\phi']\!]_\Gamma)
\end{array}$$

We define $Con(\Gamma)$ to be Γ is consistent. The definition of the interpretatin of types are for CBV normal forms. We extend this definition to non-normal terms in the following way, $t \in [\![\phi]\!]_\Gamma$ if

$[t/x]^\phi x = t$

$[t/x]^\phi y = y$
Where y is a variable distinct from x.

$[t/x]^\phi (\lambda y : \phi'.t') = \lambda y : \phi'.([t/x]^\phi t')$

$[t/x]^\phi \mathit{join} = \mathit{join}$

$[t/x]^\phi (t_1\ t_2) = ([t/x]^\phi t_1)\ ([t/x]^\phi t_2)$
Where $([t/x]^\phi t_1)$ is not a λ-abstraction or $([t/x]^\phi t_1)$ and t_1 are λ-abstractions, or $ctype_\phi(x, t_1)$ is undefined.

$[t/x]^\phi (t_1\ t_2) = [([t/x]^\phi t_2)/y]^{\phi_1} s'_1$
Where $([t/x]^\phi t_1) \equiv \lambda y : \phi_1.s'_1$ for some y and s'_1 and t_1 is not a λ-abstraction, and $ctype_\phi(x, t_1) = \phi_1 \to \phi_2$.

Figure 16 The Hereditary Substitution Function for STLC$^=$

and only if $t \rightsquigarrow_\beta v \in [\![\phi]\!]_\Gamma$. This definition is substantially different then the previous definitions. First, this definition uses full β-reduction not CBV. We must use full β-reduction in the definition of the interpretation of types, because we need them to be closed under hereditary substitution, but hereditary substitution will reduce under λ-abstractions. Second, since we have equations between types we cannot say that ϕ itself carries over to the right hand side of the definitions for variables, λ-abstractions, and join, but rather there exist a proof p and a type ϕ', such that p's type is an equation between ϕ and ϕ'.

One crucial point to make regarding the definition of the interpretation of types is that we only know the body of a λ-abstraction is in an interpretation of some type if the context is consistent. Since the reduction strategy is CBV the body of λ-abstractions can be anything even potentially diverging terms. Now in this system the only time we can type check a diverging term is if the context is inconsistent, just as we saw in Example 28. Hence, we only know the body of a λ-abstraction is in the interpretation of some type if the context is consistent. Without this requirement on the bodies of λ-abstractions this proof will not work. Also, the clause for equality types quantifies over substitutions which satisfy a context – $\sigma : \Gamma$ – defined as follows:

$$\frac{}{\cdot : \cdot}\ \textsc{SubEmpty} \qquad \frac{\vdash t : \phi \qquad \sigma : \Gamma}{(\{x \mapsto t\} \cup \sigma) : (x : \phi, \Gamma)}\ \textsc{SubTerm} \qquad \frac{\vdash \phi \qquad \sigma : [\phi/X]\Gamma}{(\{X \mapsto \phi\} \cup \sigma) : (X, \Gamma)}\ \textsc{SubType}$$

We now move on to showing soundness of typing. Similarly to the proof for DSSF$^=$ we need semantic equality stated in the next lemma. Following the semantic equality lemma is the main substitution lemma. One thing to note is that this lemma uses full β-reduction not CBV.

▶ **Lemma 30** (Semantic Equality). *If $\Gamma \vdash p : \phi_1 = \phi_2$ and $v \in [\![[\phi_1/X]\phi]\!]_\Gamma$ then $v \in [\![[\phi_2/X]\phi]\!]_\Gamma$.*

▶ **Lemma 31** (Weakening for the Interpretation of Types). *If $v \in [\![\phi]\!]_\Gamma$ then $v \in [\![\phi]\!]_{\Gamma,\Gamma'}$ for any context Γ' which does not overlap with Γ.*

▶ **Lemma 32** (Hereditary Substitution for the Interpretation of Types). *If $v' \in [\![\phi']\!]_{\Gamma,x:\phi,\Gamma'}$ and $v \in [\![\phi]\!]_\Gamma$ then $[v/x]^\phi v' \in [\![\phi']\!]_{\Gamma,\Gamma'}$.*

We now conclude soundness of typing and hence normalization for STLC$^=$.

▶ **Theorem 33** (Type Soundness). *If $\Gamma \vdash t : \phi$ then $t \in [\![\phi]\!]_\Gamma$.*

▶ **Corollary 34** (Normalization). *If $\Gamma \vdash t : \phi$ and $Con(\Gamma)$ then there exists a value v such that $t \rightsquigarrow^! v$.*

5 Conclusion

We have applied the proof technique of normalization by hereditary substitution, which uses the notion of interpretation of types and the central idea of hereditary substitution, to analyze three advanced typing features: sum types with commuting conversions, equality types between terms, and equality types between types. Our long-term goal is to apply the hereditary-substitutions method to proof-theoretically more complex type theories, in particular Gödel's System T. For this, we conjecture that a proof-theoretically more complex ordering on types will be required, and hence are exploring extensions of SSF to higher ordinals. A first step in this direction has already been taken by Danner and Leivant, using the ordinal ω^ω [7].

References

1 Andreas Abel. Implementing a normalizer using sized heterogeneous types. In *In Workshop on Mathematically Structured Functional Programming, MSFP*, 2006.

2 Andreas Abel and Dulma Rodriguez. Syntactic metatheory of higher-order subtyping. In *Proceedings of the 22nd international workshop on Computer Science Logic*, CSL '08, pages 446–460, Berlin, Heidelberg, 2008. Springer-Verlag.

3 Robin Adams. *A Modular Hierarchy of Logical Frameworks*. PhD thesis, 2004.

4 R.M. Amadio and P.L. Curien. *Domains and lambda-calculi*. Cambridge tracts in theoretical computer science. Cambridge University Press, 1998.

5 H. Barendregt. Lambda calculi with types. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 117–309. Oxford University Press, 1992.

6 C. Chen and H. Xi. Combining Programming with Theorem Proving. In *Proceedings of the 10th International Conference on Functional Programming (ICFP05)*, Tallinn, Estonia, September 2005.

7 N. Danner and D. Leivant. Stratified polymorphism and primitive recursion. *Mathematical. Structures in Comp. Sci.*, 9(4):507–522, 1999.

8 H. Eades and A. Stump. Exploring the Reach of Hereditary Substitution. Available from `http://www.cs.uiowa.edu/~heades/`.

9 H. Eades and A. Stump. Hereditary substitution for stratified system f. *International Workshop on Proof-Search in Type Theories, PSTT'10*, July 2010.

10 J. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, April 1989.

11 M. Hofmann and T. Streicher. The Groupoid Model Refutes Uniqueness of Identity Proofs. In *Ninth Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 208–212. IEEE Computer Society, 1994.

12 Felix Joachimski and Ralph Matthes. Short proofs of normalization for the simply-typed λ-calculus, permutative conversions and gödel's t, 1999.

13 C. Keller and T. Altenkirch. Normalization by hereditary substitutions. In *Proceedings of the third ACM SIGPLAN workshop on Mathematically structured functional programming*, MSFP '10, pages 3–10, New York, NY, USA, 2010. ACM.

14 D. Leivant. Finitely stratified polymorphism. *Inf. Comput.*, 93(1):93–113, 1991.

15 Jean-Jacques Lévy. An algebraic interpretation of the $\lambda\beta k$-calculus; and an application of a labelled λ-calculus. *Theoretical Computer Science*, 2(1):97 – 114, 1976.

16 C. McBride and J. McKinna. The View from the Left. *Journal of Functional Programming*, 14(1), 2004.

17 B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory*. Oxford University Press, 1990.

18 Dag Prawitz. *Natural Deduction: A Proof-Theoretical Study*. Dover Publications, 1965.

19 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.

20 M. Tatsuta and G. Mints. A simple proof of second-order strong normalization with permutative conversions. *Annals of Pure and Applied Logic*, 136:134–155, 2005.

21 K. Watkins, I. Cervesato, F. Pfenning, and D. Walker. A concurrent logical framework: The propositional fragment. *Types for Proofs and Programs*, 3085:355–377, 2004.

A Proofs of Results Pertaining to SSF$^+$

A.1 Basic Syntactic Lemma for SSF$^+$

We now state several results about the kinding relation all of which are used throughout the remaining proofs for SSF$^+$.

▶ **Lemma 35.** *If* $\Gamma \vdash \phi : *_p$ *then* Γ *Ok.*

▶ **Lemma 36** (Level Weakening for Kinding)**.** *If* $\Gamma \vdash \phi : *_r$ *and* $r < s$ *then* $\Gamma \vdash \phi : *_s$.

▶ **Lemma 37** (Substitution for Kinding, Context-Ok)**.** *Suppose* $\Gamma \vdash \phi' : *_p$. *If* $\Gamma, X : *_p, \Gamma' \vdash \phi : *_q$ *with a derivation of depth d, then* $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]\phi : *_q$ *with a derivation of depth d. Also, if* $\Gamma, X : *_p, \Gamma'$ *Ok with a derivation of depth d, then* $\Gamma, [\phi'/X]\Gamma'$ *Ok with a derivation of depth d.*

▶ **Lemma 38** (Regularity)**.** *If* $\Gamma \vdash t : \phi$ *then* $\Gamma \vdash \phi : *_p$ *for some p.*

A.2 Proof of Theroem 2

The depth function, defined in the following definition, is used in the following proof.

▶ **Definition 39.** The depth of a type ϕ is defined as follows:

$$\begin{array}{lcl} depth(X) & = & 1 \\ depth(\phi \to \phi') & = & depth(\phi) + depth(\phi') \\ depth(\phi + \phi') & = & depth(\phi) + depth(\phi') \\ depth(\forall X : *_l.\phi) & = & depth(\phi) + 1 \end{array}$$

We define the following metric (l, d) in lexicographic combination, where l is the level of a type ϕ and d is the depth of ϕ.

▶ **Lemma 40** (Well-Founded Measure)**.** *If* $\phi >_\Gamma \phi'$ *then* $(l, d) > (l', d')$, *where* $\Gamma \vdash \phi : *_l$, $depth(\phi) = d$, $\Gamma \vdash \phi : *_{l'}$, *and* $depth(\phi') = d'$.

Finally, the proof of well-foundedness of $>_\Gamma$. If there exists an infinite decreasing sequence using our ordering on types, then there is an infinite decreasing sequence using our measure by Lemma 40, but that is impossible.

A.3 Proof of Lemma 3

Suppose $\phi >_\Gamma \phi'$ and $\phi' >_\Gamma \phi''$. If $\phi \equiv \phi_1 \to \phi_2$ or $\phi \equiv \phi_1 + \phi_2$ then, ϕ' must be a subexpression of ϕ. Now if $\phi' \equiv \phi'_1 \to \phi'_2$ or $\phi' \equiv \phi'_1 + \phi'_2$ then, ϕ'' must be a subexpression of ϕ', which implies that ϕ'' is a subexpression of ϕ. Thus, $\phi >_\Gamma \phi''$. If $\phi' \equiv \forall X : *_l.\phi'_1$ then, there exists a type ϕ'_2 where, $\Gamma \vdash \phi'_2 : *_l$, such that, $\phi'' \equiv [\phi'_2/X]\phi'_1$. The level of ϕ' is $max(l, l') + 1$, where l' is the level of ϕ'_1, the level of ϕ'' is $max(l, l')$, and the level of ϕ is $max(max(l, l') + 1, p)$, where p is the level of the type, which is, the second subexpression of ϕ. Clearly, $max(max(l, l') + 1, p) \geq max(l, l')$, thus, $\phi >_\Gamma \phi''$.

If $\phi \equiv \forall X : *_l.\phi_1$, then $\phi' \equiv [\phi_2/X]\phi_1$ for some type ϕ_2, where $\Gamma \vdash \phi_2 : *_l$. If $[\phi_2/X]\phi_1 \equiv \phi'_1 \to \phi'_2$ or $[\phi_2/X]\phi_1 \equiv \phi'_1 + \phi'_2$, then the level of ϕ' is $max(p, q)$, where p is the level of ϕ'_1 and q is the level of ϕ'_2. Now ϕ'' must be a subexpression of ϕ', hence the level of ϕ'' is either p or q. Now, since the level of ϕ is greater than the level of ϕ' and we know, $max(p, q)$ is greater than both p and q then $\phi >_\Gamma \phi''$. If $[\phi_2/X]\phi_1 \equiv \forall Y : *_{l'}.\phi'_1$, then $\phi'' \equiv [\phi'_2/X]\phi'_1$. Now if p is the level of ϕ_1, then the level of ϕ is $max(l, p) + 1$ and the level of ϕ' must be $max(l, p)$ since we know the level of ϕ' is greater than the level of ϕ'' then clearly, the level of ϕ is greater than the level of ϕ''. Thus, $\phi >_\Gamma \phi''$.

A.4 Proof of Lemma 35

This is a proof by structural induction on the kinding derivation of $\Gamma \vdash \phi : *_p$.
Case.

$$\frac{\Gamma(X) = *_p \qquad p \leq q \qquad \Gamma\ Ok}{\Gamma \vdash X : *_q}$$

By inversion of the kind-checking rule $\Gamma\ Ok$.
Case.

$$\frac{\Gamma \vdash \phi_1 : *_p \qquad \Gamma \vdash \phi_2 : *_q}{\Gamma \vdash \phi_1 \to \phi_2 : *_{max(p,q)}}$$

By the induction hypothesis, $\Gamma \vdash \phi_1 : *_p$ and $\Gamma \vdash \phi_2 : *_q$ both imply $\Gamma\ Ok$. Since the arrow-type kind-checking rule does not modify Γ in anyway Γ will remain Ok.
Case.

$$\frac{\Gamma \vdash \phi_1 : *_p \qquad \Gamma \vdash \phi_2 : *_q}{\Gamma \vdash \phi_1 + \phi_2 : *_{max(p,q)}}$$

By the induction hypothesis, $\Gamma \vdash \phi_1 : *_p$ and $\Gamma \vdash \phi_2 : *_q$ both imply $\Gamma\ Ok$. Since the sum-type kind-checking rule does not modify Γ in anyway Γ will remain Ok.
Case.

$$\frac{\Gamma, X : *_q \vdash \phi : *_p}{\Gamma \vdash \forall X : *_q.\phi : *_{max(p,q)+1}}$$

By the induction hypothesis $\Gamma, X : *_p\ Ok$, and by inversion of the type-variable well-formed contexts rule $\Gamma\ Ok$.

A.5 Proof of Lemma 36

We show level weakening for kinding by structural induction on the kinding derivation of $\phi : *_r$.
Case.

$$\frac{\Gamma(X) = *_p \qquad p \leq q \qquad \Gamma\ Ok}{\Gamma \vdash X : *_q}$$

By assumption we know $q < s$, hence by reapplying the rule and transitivity we obtain $\Gamma \vdash X : *_s$.
Case.

$$\frac{\Gamma \vdash \phi_1 : *_p \qquad \Gamma \vdash \phi_2 : *_q}{\Gamma \vdash \phi_1 \to \phi_2 : *_{max(p,q)}}$$

By the induction hypothesis $\Gamma \vdash \phi_1 : *_s$ and $\Gamma \vdash \phi_2 : *_s$ for some arbitrary $s > max(p,q)$. Therefore, by reapplying the rule we obtain $\Gamma \vdash \phi_1 \to \phi_2 : *_s$.
Case.

$$\frac{\Gamma \vdash \phi_1 : *_p \qquad \Gamma \vdash \phi_2 : *_q}{\Gamma \vdash \phi_1 + \phi_2 : *_{max(p,q)}}$$

By the induction hypothesis $\Gamma \vdash \phi_1 : *_s$ and $\Gamma \vdash \phi_2 : *_s$ for some arbitrary $s > max(p,q)$. Therefore, by reapplying the rule we obtain $\Gamma \vdash \phi_1 + \phi_2 : *_s$.

Case.

$$\frac{\Gamma, X : *_q \vdash \phi' : *_p}{\Gamma \vdash \forall X : *_q.\phi' : *_{max(p,q)+1}}$$

We know by assumption that $max(p, q) + 1 < s$ which implies that $max(p, q) < s - 1$. Now by the induction hypothesis $\Gamma, X : *_q \vdash \phi' : *_{s-1}$. Lastly, we reapply the rule and obtain $\Gamma \vdash \forall X : *_q.\phi' : *_s$.

A.6 Proof of Substitution for Kinding, Context-Ok

This is a prove by induction on d. We prove the first implication first, and then the second, doing a case analysis for each implication on the form of the derivation whose depth is being considered.

Case.

$$\frac{(\Gamma, X : *_p, \Gamma')(Y) = *_r \qquad r \leq s \qquad \Gamma, X : *_p, \Gamma'\ Ok}{\Gamma, X : *_p, \Gamma' \vdash Y : *_s}$$

By assumption $\Gamma \vdash \phi' : *_p$. We must consider whether or not $X \equiv Y$. If $X \equiv Y$ then $[\phi'/X]Y \equiv \phi'$, $r = p$, and $q = s$; this conclusion is equivalent to $\Gamma, [\phi'/X]\Gamma' \vdash \phi' : *_q$ and by the induction hypothesis $\Gamma, [\phi'/X]\Gamma'\ Ok$. If $X \not\equiv Y$ then $[\phi'/X]Y \equiv Y$ and by the induction hypothesis $\Gamma, [\phi'/X]\Gamma'\ Ok$, hence, $\Gamma, [\phi'/X]\Gamma' \vdash Y : *_q$.

Case.

$$\frac{\Gamma, X : *_p, \Gamma' \vdash \phi_1 : *_r \qquad \Gamma, X : *_p, \Gamma' \vdash \phi_2 : *_s}{\Gamma, X : *_p, \Gamma' \vdash \phi_1 \to \phi_2 : *_{max(r,s)}}$$

Here $q = max(r, s)$ and by the induction hypothesis $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]\phi_1 : *_r$ and $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]\phi_2 : *_s$. We can now reapply the rule to get $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X](\phi_1 \to \phi_2) : *_q$.

Case.

$$\frac{\Gamma, X : *_p, \Gamma' \vdash \phi_1 : *_r \qquad \Gamma, X : *_p, \Gamma' \vdash \phi_2 : *_s}{\Gamma, X : *_p, \Gamma' \vdash \phi_1 + \phi_2 : *_{max(r,s)}}$$

Here $q = max(r, s)$ and by the induction hypothesis $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]\phi_1 : *_r$ and $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]\phi_2 : *_s$. We can now reapply the rule to get $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X](\phi_1 + \phi_2) : *_q$.

Case.

$$\frac{\Gamma, X : *_q, \Gamma', Y : *_r \vdash \phi : *_s}{\Gamma, X : *_p, \Gamma' \vdash \forall Y : *_r.\phi : *_{max(r,s)+1}}$$

Here $q = max(r, s) + 1$ and by the induction hypothesis $\Gamma, [\phi'/X]\Gamma', Y : *_r \vdash [\phi'/X]\phi : *_s$. We can reapply this rule to get $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]\forall Y : *_r.\phi : *_q$.

We now show the second implication. The case were $d = 0$ cannot arise, since it requires the context to be empty. Suppose $d = n + 1$. We do a case analysis on the last rule applied in the derivation of $\Gamma, X : *_p, \Gamma'$.

Case. Suppose $\Gamma' = \Gamma'', Y : *_q$.

$$\frac{\Gamma, X : *_p, \Gamma'' \; Ok}{\Gamma, X : *_p, \Gamma'', Y : *_q \; Ok}$$

By the induction hypothesis, $\Gamma, [\phi'/X]\Gamma'' \; Ok$. Now, by reapplying the rule above $\Gamma, [\phi'/X]\Gamma'', Y : *_q \; Ok$, hence $\Gamma, [\phi'/X]\Gamma' \; Ok$, since $X \not\equiv Y$.

Case. Suppose $\Gamma' = \Gamma'', y : \phi$.

$$\frac{\Gamma, X : *_p, \Gamma'' \vdash \phi : *_q \qquad \Gamma, X : *_p, \Gamma'' \; Ok}{\Gamma, X : *_p, \Gamma'', y : \phi \; Ok}$$

By the induction hypothesis, $\Gamma', [\phi'/X]\Gamma'' \vdash [\phi'/X]\phi : *_q$ and $\Gamma', [\phi'/X]\Gamma'' \; Ok$. Thus, by reapplying the rule above $\Gamma, [\phi'/X]\Gamma'', x : [\phi'/X]\phi \; Ok$, therefore,
$\Gamma, [\phi'/X]\Gamma' \; Ok$.

A.7 Proof of Type Ordering

This is a proof by case analysis on the kinding derivation of $\Gamma \vdash \phi : *_p$, with a case analysis on the derivation of $\phi >_\Gamma \phi'$.

Case.

$$\frac{\Gamma(X) = *_p \qquad p \leq q \qquad \Gamma \; Ok}{\Gamma \vdash X : *_q}$$

This case cannot arise, because we do not have $X >_\Gamma \phi$ for any type ϕ.

Case.

$$\frac{\Gamma \vdash \phi_1 : *_p \qquad \Gamma \vdash \phi_2 : *_q}{\Gamma \vdash \phi_1 \to \phi_2 : *_{max(p,q)}}$$

By analysis of the derivation of the assumed ordering statement, we must have $\phi' \equiv \phi_1$ or $\phi' \equiv \phi_2$. If $\phi' \equiv \phi_1$ and $p \geq q$ then we have the required kind derivation for ϕ'. If $p < q$ then by level weakening $\Gamma \vdash \phi_1 : *_q$, and we have the required kinding derivation for ϕ'. The case for when $\phi' \equiv \phi_2$ is similar.

Case.

$$\frac{\Gamma \vdash \phi_1 : *_p \qquad \Gamma \vdash \phi_2 : *_q}{\Gamma \vdash \phi_1 + \phi_2 : *_{max(p,q)}}$$

By analysis of the derivation of the assumed ordering statement, we must have $\phi' \equiv \phi_1$ or $\phi' \equiv \phi_2$. If $\phi' \equiv \phi_1$ and $p \geq q$ then we have the required kind derivation for ϕ'. If $p < q$ then by level weakening $\Gamma \vdash \phi_1 : *_q$, and we have the required kinding derivation for ϕ'. The case for when $\phi' \equiv \phi_2$ is similar.

Case.

$$\frac{\Gamma, X : *_r \vdash \phi : *_s}{\Gamma \vdash \forall X : *_r.\phi : *_{max(r,s)+1}}$$

By analysis of the derivation of the assumed ordering statement, we must have $\phi' \equiv [\phi''/X]\phi$, for some type ϕ'' with $\Gamma \vdash \phi'' : *_r$. Let $t = max(r, s) + 1$. Clearly, $s < t$, hence by level weakening $\Gamma, X : *_r \vdash \phi : *_t$ and by substitution for kinding $\Gamma \vdash [\phi''/X]\phi : *_t$, and we have the required kinding derivation for ϕ'.

A.8 Proof of Lemma 44

Assume $\phi >_\Gamma \phi'$ for some types ϕ and ϕ'. We case split on the form of ϕ. Clearly, ϕ is not a type variable.

Case. Suppose $\phi \equiv \Pi x : \phi_1.\phi_2$. Then ϕ' must be of the form ϕ_1 or $[t/x]\phi_2$, for some term $\Gamma \vdash t : \phi_1$. In both cases we have two cases to consider; either ϕ and ϕ' have the same level or they do not. Consider the first form and suppose they have the same level. Then it is clear that $depth(\phi) > depth(\phi')$. Now consider the latter form and suppose ϕ and ϕ' have the same level. Then clearly $depth(\phi) > depth(\phi')$. In either form if the level of ϕ and ϕ' are different, then the level of ϕ is larger than the level of ϕ'. In all cases $(l, d) > (l', d')$.

Case. Suppose $\phi \equiv \forall X : *_l.\phi_1$. Then ϕ' must be of the form $[\phi_2/X]\phi_1$ for some type $\Gamma \vdash \phi_2 : *_l$. It is obvious that the level of ϕ is always larger than the level of ϕ'. Hence, $(l, d) > (l', d')$.

A.9 Proof of Theorem 20

If there exists a infinite decreasing sequence using our ordering on types, then there is an infinite decreasing sequence using our measure by Lemma 44, but that is impossible.

A.10 Proof of Regularity

This proof is by structural induction on the derivation of $\Gamma \vdash t : \phi$.

Case.

$$\frac{\Gamma(x) = \phi \qquad \Gamma\ Ok}{\Gamma \vdash x : \phi}$$

By the definition of well-formedness contexts $\Gamma \vdash \phi : *_p$ for some p.

Case.

$$\frac{\Gamma, x : \phi_1 \vdash t : \phi_2}{\Gamma \vdash \lambda x : \phi_1.t : \phi_1 \to \phi_2}$$

By the induction hypothesis $\Gamma \vdash \phi_1 : *_p, \Gamma, x : \phi_1 \vdash \phi_2 : *_q$ and by Lemma **??**, $\Gamma \vdash \phi_2 : *_q$. By applying the arrow-type kind-checking rule we get $\Gamma \vdash \phi_1 \to \phi_2 : *_{max(p,q)}$.

Case.

$$\frac{\Gamma \vdash t : \phi_1 + \phi_2 \qquad \Gamma, x : \phi_1 \vdash t_1 : \psi \qquad \Gamma, x : \phi_2 \vdash t_2 : \psi}{\Gamma \vdash \text{case } t \text{ of } x.t_1, x.t_2 : \psi}$$

By the induction hypothesis, $\Gamma, x : \phi_1 \vdash \psi : *_p$, for some p. By Lemma **??**, $\Gamma \vdash \psi : *_p$.

Case.

$$\frac{\Gamma \vdash t : \phi_1 \qquad \Gamma \vdash \phi_2 : *_p}{\Gamma \vdash inl(t) : \phi_1 + \phi_2}$$

By the induction hypothesis, $\Gamma \vdash \phi_1 : *_q$, and by inversion of the type-checking rule above, $\Gamma \vdash \phi_2 : *_p$. Finally, by applying the sum-type kind-checking rule, $\Gamma \vdash \phi_1 + \phi_2 : *_{max(q,p)}$.

Case.

$$\frac{\Gamma \vdash t : \phi_2 \qquad \Gamma \vdash \phi_1 : *_p}{\Gamma \vdash inr(t) : \phi_1 + \phi_2}$$

Similar to the inject-left case above.

Case.

$$\frac{\Gamma \vdash t_1 : \phi_1 \to \phi_2 \qquad \Gamma \vdash t_2 : \phi_1}{\Gamma \vdash t_1\ t_2 : \phi_2}$$

By the induction hypothesis $\Gamma \vdash \phi_1 \to \phi_2 : *r$ and $\Gamma \vdash \phi_1 : *_p$. By inversion of the arrow-type kind-checking rule $r = max(p, q)$, for some q, which implies $\Gamma \vdash \phi_2 : *_q$.

Case.

$$\frac{\Gamma, X : *_p \vdash t : \phi}{\Gamma \vdash \Lambda X : *_p.t : \forall X : *_q.\phi}$$

By the induction hypothesis $\Gamma, X : *_q \vdash \phi : *_p$. By applying the forall-type kind-checking rule $\Gamma \vdash \forall X.\phi : *_{max(p,q)+1}$.

Case.

$$\frac{\Gamma \vdash t : \forall X : *_p.\phi_1 \qquad \Gamma \vdash \phi_2 : *_p}{\Gamma \vdash t[\phi_2] : [\phi_2/X]\phi_1}$$

By assumption $\Gamma \vdash \phi_2 : *_r$. By the induction hypothesis $\Gamma \vdash \forall X : *_p.\phi_1 : *_s$ and by inversion of the forall-type kind-checking rule $r = max(p, q) + 1$, for some q, which implies $\Gamma, X : *_p \vdash \phi_1 : *_q$. Now, by Lemma **??**, $\Gamma \vdash [\phi_2/X]\phi_1 : *_q$.

A.11 Proof of the Properties of $ctype_\phi$

We prove part one first. This is a proof by induction on the structure of t.

Case. Suppose $t \equiv x$. Then $ctype_\phi(x, x) = \phi$. Clearly, $head(x) = x$ and ϕ is a subexpression of itself.

Case. Suppose $t \equiv t_1\ t_2$. Then $ctype_\phi(x, t_1\ t_2) = \phi''$ when $ctype_\phi(x, t_1) = \phi' \to \phi''$. Now $t > t_1$ so by the induciton hypothesis $head(t_1) = x$ and $\phi' \to \phi''$ is a subexpression of ϕ. Therefore, $head(t_1\ t_2) = x$ and certainly ϕ'' is a subexpression of ϕ.

We now prove part two. This is also a proof by induction on the structure of t.

Case. Suppose $t \equiv x$. Then $ctype_\phi(x, x) = \phi$. Clearly, $\phi \equiv \phi$.

Case. Suppose $t \equiv t_1\ t_2$. Then $ctype_\phi(x, t_1\ t_2) = \phi_2$ when $ctype_\phi(x, t_1) = \phi_1 \to \phi_2$. By inversion on the assumed typing derivation we know there exists type ϕ'' such that $\Gamma, x : \phi, \Gamma' \vdash t_1 : \phi'' \to \phi'$. Now $t > t_1$ so by the induciton hypothesis $\phi_1 \to \phi_2 \equiv \phi'' \to \phi'$. Therefore, $\phi_1 \equiv \phi''$ and $\phi_2 \equiv \phi'$.

Next we prove part three. This is a proof by induction on the structure of $t_1\ t_2$.

The only possiblities for the form of t_1 is x or $\hat{t}_1\ \hat{t}_2$. All other forms would not result in $[t/x]^\phi t_1$ being a λ-abstraction and t_1 not. If $t_1 \equiv x$ then there exist a type ϕ'' such that $\phi \equiv \phi'' \to \phi'$ and $ctype_\phi(x, x\ t_2) = \phi'$ when $ctype_\phi(x, x) = \phi \equiv \phi'' \to \phi'$ in this case. We know ϕ'' to exist by inversion on $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$.

Now suppose $t_1 \equiv (\hat{t}_1\ \hat{t}_2)$. Now knowing t_1' to not be a λ-abstraction implies that $\hat{t}_1$ is also not a λ-abstraction or $[t/x]^{\phi}t_1$ would be an application instead of a λ-abstraction. So it must be the case that $[t/x]^{\phi}\hat{t}_1$ is a λ-abstraction and $\hat{t}_1$ is not. Since $t_1\ t_2 > t_1$ we can apply the induction hypothesis to obtain there exists a type ψ such that $ctype_{\phi}(x, \hat{t}_1) = \psi$. Now by inversion on $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$ we know there exists a type ϕ'' such that $\Gamma, x : \phi, \Gamma' \vdash t_1 : \phi'' \to \phi'$. We know $t_1 \equiv (\hat{t}_1\ \hat{t}_2)$ so by inversion on $\Gamma, x : \phi, \Gamma' \vdash t_1 : \phi'' \to \phi'$ we know there exists a type ψ'' such that $\Gamma, x : \phi, \Gamma' \vdash \hat{t}_1 : \psi'' \to (\phi'' \to \phi')$. By part two of Lemma 4 we know $\psi \equiv \psi'' \to (\phi'' \to \phi')$ and $ctype_{\phi}(x, t_1) = ctype_{\phi}(x, \hat{t}_1\ \hat{t}_2) = \phi'' \to \phi'$ when $ctype_{\phi}(x, \hat{t}_1) = \psi'' \to (\phi'' \to \phi')$, because we know $ctype_{\phi}(x, \hat{t}_1) = \psi$.

The proofs of the remaining parts are similar to the proof of part three.

A.12 Proof of Totality and Type Preservation

This is a proof by induction on the lexicorgraphic combination (ϕ, t') of $>_{\Gamma,\Gamma'}$ and the strict subexpression ordering. We case split on t'.

Case. Suppose t' is either x or a variable y distinct from x. Trivial in both cases.

Case. Suppose $t' \equiv \lambda y : \phi_1.t'_1$. By inversion on the typing judgement we know $\Gamma, x : \phi, \Gamma', y : \phi_1 \vdash t'_1 : \phi_2$. We also know $t' > t'_1$, hence we can apply the induction hypothesis to obtain $[t/x]^{\phi}t'_1 = \hat{t}'_1$ and $\Gamma, \Gamma', y : \phi_1 \vdash \hat{t} : \phi_2$ for some term $\hat{t}'_1$. By the definition of the hereditary substitution function $[t/x]^{\phi}t' = \lambda y : \phi_1.[t/x]^{\phi}t'_1 = \lambda y : \phi_1.\hat{t}'_1$. It suffices to show that $\Gamma, \Gamma' \vdash \lambda y : \phi_1.\hat{t}'_1 : \phi_1 \to \phi_2$. By simply applying the λ-abstraction typing rule using $\Gamma, \Gamma', y : \phi_1 \vdash \hat{t} : \phi_2$ we obtain $\Gamma, \Gamma' \vdash \lambda y : \phi_1.\hat{t}'_1 : \phi_1 \to \phi_2$.

Case. Suppose $t' \equiv \Lambda X : *_l.t'_1$. Similar to the previous case.

Case. Suppose $t' \equiv t'_1\ t'_2$. By inversion we know $\Gamma, x : \phi, \Gamma' \vdash t'_1 : \phi'' \to \phi'$ and $\Gamma, x : \phi, \Gamma' \vdash t'_2 : \phi''$ for some types ϕ' and ϕ''. Clearly, $t' > t'_i$ for $i \in \{1, 2\}$. Thus, by the induction hypothesis there exists terms m_1 and m_2 such that $[t/x]^{\phi}t'_i = m_i$, $\Gamma, \Gamma' \vdash m_1 : \phi'' \to \phi'$ and $\Gamma, \Gamma' \vdash m_2 : \phi''$ for $i \in \{1, 2\}$. We case split on whether or not m_1 is a λ-abstraction, a case construct and t'_1 is not, or $ctype_{\phi}(x, t'_1)$ is undefined. We only consider the non-trivial cases when $m_1 \equiv \lambda y : \phi''.m'_1$ and t'_1 is not a λ-abstraction or $m_1 \equiv$ case m'_0 of $y.m'_1$,$y.m'_2$, $ctype_{\phi}(x, t'_1) = \psi'' \to \psi'$, and t'_1 is not a case construct. Suppose the former. Now by Lemma 4 it is the case that there exists a ψ such that $ctype_{\phi}(x, t'_1) = \psi$, $\psi \equiv \phi'' \to \phi'$, and ψ is a subexpression of ϕ, hence $\phi >_{\Gamma,\Gamma'} \phi''$. Then $[t/x]^{\phi}(t'_1\ t'_2) = [m_2/y]^{\psi''}m'_1$. Therefore, by the induction hypothesis there exists a term m such that $[m_2/y]^{\phi''}m'_1 = m$ and $\Gamma, \Gamma' \vdash m : \phi''$.

Suppose $m_1 \equiv$ case m'_0 of $y.m'_1$,$y.m'_2$ and t'_1 is not a case construct. Now $[t/x]^{\phi}t' =$ case m'_0 of $y.app_{\phi}\ m'_1\ m_2$,$y.app_{\phi}$ By inversion on $\Gamma, \Gamma' \vdash m_1 : \phi'' \to \phi'$ we know there exists terms ϕ_1 and ϕ_2 such that $\Gamma, \Gamma' \vdash m'_0 : \phi_1 + \phi_2$ and $\Gamma, \Gamma', y : \phi_i \vdash m'_i : \phi'' \to \phi'$ for $i \in \{1, 2\}$. It suffcies to show that there exists terms q and q' such that $app_{\phi}\ m'_1\ m_2 = q$ and $app_{\phi}\ m'_2\ m_2 = q'$. To obtain this result we prove the following proposition. Note that $ctype_{\phi}(x, t'_1) = \psi'' \to \psi'$ which by Lemma 4 is equivalent to $\phi'' \to \phi'$ and is a subexpression of ϕ, hence $\phi >_{\Gamma,\Gamma'} \phi''$ and $\phi >_{\Gamma,\Gamma'} \phi'$.

Proposition. For all $\Gamma \vdash m_2 : \phi''$ and $\Gamma \vdash m'_1 : \phi'' \to \phi'$ there exists a term q such that $app_{\phi}\ m'_1\ m_2 = q$ and $\Gamma \vdash q : \phi'$.

We prove this by nested induction on the ordering (ϕ, t', m'_1) and case splitting on the structure of m'_1.

Case. Suppose m'_1 is neither a λ-abstraction or a case construct. Then $app_\phi\ m'_1\ m_2 = m'_1\ m_2$. Take $m'_1\ m_2$ for q and by applying the application typing rule we know $\Gamma \vdash m'_1\ m_2 : \phi'$.

Case. Suppose $m'_1 \equiv \lambda z : \phi''.m''_1$. Then $app_\phi\ m'_1\ m_2 = [m_2/z]^{\phi''} m''_1$. By inversion on the assumption $\Gamma \vdash m'_1 : \phi'' \rightarrow \phi'$ we know $\Gamma, z : \phi'' \vdash m''_1 : \phi'$. Since $\phi >_\Gamma \phi''$ we can apply the outter induction hypothesis to obtain there there exists a q such that $[m_2/z]^{\phi''} m''_1 = q$ and $\Gamma \vdash q : \phi'$. Therefore, $app_\phi\ m'_1\ m_2 = q$.

Case. Suppose $m'_1 \equiv$ case m''_0 of $z.m''_1$,$z.m''_2$. Then $app_\phi\ m'_1\ m_2 =$ case m''_0 of $z.app_\phi\ m''_1\ m_2$,$z.app_\phi\ m''_2\ m_2$. By inversion on the assumption $\Gamma \vdash m'_1 : \phi'' \rightarrow \phi'$ we know there exists types ϕ_1 and ϕ_2 such that $\Gamma \vdash m''_0 : \phi_1 + \phi_2$, $\Gamma, z : \phi_1 \vdash m''_1 : \phi'' \rightarrow \phi'$ and $\Gamma, z : \phi_2 \vdash m''_2 : \phi'' \rightarrow \phi'$. Since $m'_1 > m''_1$ and $m'_1 > m''_2$ we can apply the inner induction hypothesis to obtain there exists terms q' and q'' such that $app_\phi\ m''_1\ m_2 = q'$, $\Gamma, z : \phi_1 \vdash q' : \phi'$, $app_\phi\ m''_1\ m_2 = q''$ and $\Gamma, z : \phi_2 \vdash q'' : \phi'$. Hence, $app_\phi\ m'_1\ m_2 =$ case m''_0 of $z.app_\phi\ m''_1\ m_2$,$z.app_\phi\ m''_2\ m_2 =$ case m''_0 of $z.q'$,$z.q''$. It suffices to to show that $\Gamma \vdash$ case m''_0 of $z.q'$,$z.q'' : \phi'$. This is a simple consequence of applying the case-construct typing rule using $\Gamma \vdash m''_0 : \phi_1 + \phi_2$, $\Gamma, z : \phi_1 \vdash q' : \phi'$, and $\Gamma, z : \phi_2 \vdash q'' : \phi'$.

By the previous proposition there exists terms q and q' such that $[t/x]^\phi t' =$ case m'_0 of $y.app_\phi\ m'_1\ m_2$,$y.app_\phi\ m'_2\ m_2 =$ case m'_0 of $y.q$,$y.q'$, where $app_\phi\ m'_1\ m_2 = q$, $\Gamma, \Gamma', y : \phi_1 \vdash q : \phi'$, $app_\phi\ m'_1\ m_2 = q'$ and $\Gamma, \Gamma', y : \phi_2 \vdash q' : \phi'$. It suffices to show that $\Gamma, \Gamma' \vdash$ case m'_0 of $y.q$,$y.q' : \phi'$. From above we know that $\Gamma, \Gamma' \vdash m'_0 : \phi_1 + \phi_2$, $\Gamma, \Gamma', y : \phi_1 \vdash q : \phi'$ and $\Gamma, \Gamma', y : \phi_2 \vdash q' : \phi'$. Thus, by applying the case-construct typing rule we obtain $\Gamma, \Gamma' \vdash$ case m'_0 of $y.q$,$y.q' : \phi'$.

Case. Suppose $t' \equiv t'_1[\phi'']$. Similar to the previous case.

Case. Suppose $t' \equiv inl(t)$. Trivial.

Case. Suppose $t' \equiv inr(t)$. Trivial.

Case. Suppose $t' \equiv$ case m_0 of $y.m_1$,$y.m_2$. By inversion on the assumption $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ we know the following:

$$\begin{array}{l} \Gamma, x : \phi, \Gamma' \vdash m_0 : \phi_1 + \phi_2, \text{ for some types } \phi_1 \text{ and } \phi_2, \\ \Gamma, x : \phi, \Gamma', y : \phi_1 \vdash m_1 : \phi, and \\ \Gamma, x : \phi, \Gamma', y : \phi_2 \vdash m_2 : \phi. \end{array}$$

It is easy to see that $t' > m_i$ for all $i \in \{0, 1, 2\}$. Hence, by the induction hypothesis there exists terms m'_0, m'_1, and m'_2 such that $[t/x]^\phi m_i = m'_i$ for all $i \in \{0, 1, 2\}$,

$$\begin{array}{ll} \text{(i)} & \Gamma, \Gamma \vdash m'_0 : \phi_1 + \phi_2, \\ \text{(ii)} & \Gamma, \Gamma', y : \phi_1 \vdash m'_1 : \phi', \text{ and} \\ \text{(iii)} & \Gamma, \Gamma', y : \phi_2 \vdash m'_2 : \phi'. \end{array}$$

We have two cases to consider.

Case. Suppose m_0 and m'_0 are inject-left terms, inject-right terms, or case constructs, or m_0 is an inject-left term, inject-right term, or a case-construct and m'_0 is not, or m_0 and m'_0 are neither inject-left terms, inject-right terms, or case constructs, or $ctype_\phi(x, m_0)$ is undefined. Then $[t/x]^\phi$(case m_0 of $y.m_1$,$y.m_2$) $=$ case m'_0 of $y.m'_1$,$y.m'_2$ and by applying the case-construct typing rule to i - iii above we obtain $\Gamma, \Gamma' \vdash$ case m'_0 of $y.m'_1$,$y.m'_2 : \phi'$.

Case. Suppose m'_0 is an inject-left term, inject-right term, or case construct and $ctype_\phi(x, m_0) = \psi_1 + \psi_2$. Then $[t/x]^\phi$(case m_0 of $y.m_1$,$y.m_2$) $= rcase_\phi\ m'_0\ y\ m'_1\ m'_2$ and by Lemma 4 we know $\psi_1 + \psi_2 \equiv \phi_1 + \phi_2$ and is a subexpression of ϕ. It suffices to show that there exists some term q such that $rcase_\phi\ m'_0\ y\ m'_1\ m'_2 = q$ and $\Gamma, \Gamma' \vdash q : \phi'$. We obtain this result by the following proposition.

Proposition. For all $\Gamma \vdash q_0 : \phi$, $\Gamma, y : \phi_1 \vdash q_1 : \phi'$, and $\Gamma, y : \phi_2 \vdash q_2 : \phi'$ there exists a term $\hat{q}$ such that $rcase_{\phi}\ q_0\ y\ q_1\ q_2 = \hat{q}$ and $\Gamma \vdash \hat{q} : \phi'$. We prove this by induction on the the ordering (ϕ, t', q_0) and case split on the structure of q_0.

Case. Suppose q_0 is not an inject-left term, inject-right term, or a case construct. Then $rcase_{\phi}\ q_0\ y\ q_1\ q_2 =$ case q_0 of $y.q_1$,$y.q_2$ and by applying the case-construct typing rule using the assumptions $\Gamma \vdash q_0 : \phi$, $\Gamma, y : \phi_1 \vdash q_1 : \phi'$, and $\Gamma, y : \phi_2 \vdash q_2 : \phi'$ we obtain $\Gamma, \Gamma' \vdash$ case q_0 of $y.q_1$,$y.q_2 : \phi'$.

Case. Suppose $q_0 \equiv inl(q_0')$. Then $rcase_{\phi}\ q_0\ y\ q_1\ q_2 = [q_0'/y]^{\phi_1} q_1$ and by inversion on $\Gamma \vdash q_0 : \phi$ we know $\Gamma \vdash q_0' : \phi_1$. It suffices to show that there exists a term $\hat{q}$ such that $[q_0'/y]^{\phi_1} q_1 = \hat{q}$ and $\Gamma \vdash \hat{q} : \phi'$. Since $\phi >_{\Gamma} \phi'$ we can apply the the outer induction hypothesis to obtain that there exists such a term $\hat{q}$.

Case. Suppose $q_0 \equiv inl(q_0')$. Similar to the previous case.

Case. Suppose $q_0 \equiv$ case q_0' of $z.q_1'$,$z.q_2'$. Then $rcase_{\phi}\ q_0\ y\ q_1\ q_2 =$ case q_0' of $z.(rcase_{\phi}\ q_1'\ y\ q_1\ q_2)$,$z.(rcase_{\phi}\ q_2'\ y\ q_1\ q_2)$. We know by assumption that $\Gamma \vdash q_0 : \phi$, $\Gamma \vdash q_0 : \phi$, and $\Gamma, y : \phi_1 \vdash q_1 : \phi'$ so by inversion we know the following:

(i) $\Gamma \vdash q_0' : \phi_1' + \phi_2'$, for some types ϕ_1' and ϕ_2',
(ii) $\Gamma, z : \phi_1' \vdash q_1' : \phi$, and
(iii) $\Gamma, z : \phi_2' \vdash q_2' : \phi$.

Now $q_0 > q_1'$ and $q_0 > q_1'$ so we can apply the induction hypothesis twice to obtain terms $\hat{q}_1$ and $\hat{q}_2$ such that $rcase_{\phi}\ q_1'\ y\ q_1\ q_2 = \hat{q}_1$, $rcase_{\phi}\ q_1'\ y\ q_1\ q_2 = \hat{q}_1$, $\Gamma, z : \phi_1' \vdash \hat{q}_1 : \phi'$ and $\Gamma, z : \phi_2' \vdash \hat{q}_2 : \phi'$. So case q_0' of $z.(rcase_{\phi}\ q_1'\ y\ q_1\ q_2)$,$z.(rcase_{\phi}\ q_2'\ y\ q_1\ q_2) =$ case q_0' of $z.\hat{q}_1$,$z.\hat{q}_2$. It suffices to show that case q_0' of $z.\hat{q}_1$,$z.\hat{q}_2 = \hat{q}$ and $\Gamma \vdash \hat{q} : \phi$ for some term $\hat{q}$. Now $q_0 > q_0'$ so we can apply the induction hypothesis to obtain our result, but before we can we must show that $\Gamma \vdash$ case q_0' of $z.\hat{q}_1$,$z.\hat{q}_2 : \phi'$. This is a direct consequence of applying the case-construct typing rule using i, $\Gamma, z : \phi_1' \vdash \hat{q}_1 : \phi'$ and $\Gamma, z : \phi_2' \vdash \hat{q}_2 : \phi'$. Therefore, by the induction hypothesis there exists a term $\hat{q}$ such that case q_0' of $z.\hat{q}_1$,$z.\hat{q}_2 = \hat{q}$ and $\Gamma \vdash \hat{q} : \phi$.

A.13 Proof of Redex Preservation

This is a proof by induction on the lexicorgraphic combination (ϕ, t') of $>_{\Gamma,\Gamma'}$ and the strict subexpression ordering. We case split on the structure of t'.

Case. Let $t' \equiv x$ or $t' \equiv y$ where y is distinct from x. Trivial.

Case. Let $t' \equiv \lambda x : \phi_1.t''$. Then $[t/x]^{\phi} t' \equiv \lambda x : \phi_1.[t/x]^{\phi} t''$. Now

$$\begin{array}{lll} rset(\lambda x : \phi_1.t'', t) & = & rset(\lambda x : \phi_1.t'') \cup rset(t) \\ & = & rset(t'') \cup rset(t) \\ & = & rset(t'', t). \end{array}$$

We know that $t' > t''$ by the strict subexpression ordering, hence by the induction hypothesis $|rset(t'', t)| \geq |rset([t/x]^{\phi} t'')|$ which implies $|rset(t', t)| \geq |rset([t/x]^{\phi} t')|$.

Case. Let $t' \equiv \Lambda X : *_l.t''$. Similar to the previous case.

Case. Let $t' \equiv inl(t'')$. We know $rset(t', t) = rset(t'', t)$. Since $t' > t''$ we can apply the induction hypothesis to obtain $|rset(t'', t)| \geq |rset([t/x]^{\phi} t'')|$. This implies $|rset(t', t)| \geq |rset([t/x]^{\phi} t')|$.

Case. Let $t' \equiv inr(t'')$. Similar to the previous case.

Case. Let $t' \equiv t_1'\ t_2'$. First consider when t_1' is not a λ-abstraction or a case construct. Then

$$rset(t_1'\ t_2', t) = rset(t_1', t_2', t)$$

Clearly, $t' > t_i'$ for $i \in \{1, 2\}$, hence, by the induction hypothesis $|rset(t_i', t)| \geq |rset([t/x]^{\phi} t_i')|$. We have three cases to consider. That is whether or not $[t/x]^{\phi} t_1'$ is a λ-abstraction, a case construct, or neither, or $ctype_{\phi}(x, t_1')$ is undefined. Suppose it is a λ-abstraction. Then by Lemma 4 $ctype_{\phi}(x.t_1') = \psi$ and by inversion on $\Gamma, x : \phi, \Gamma' \vdash t_1'\ t_2' : \phi'$ there exists a type ϕ'' such that

$\Gamma, x : \phi, \Gamma' \vdash t_1 : \phi'' \to \phi'$. Again, by Lemma 4 $\psi \equiv \phi'' \to \phi'$. Thus, $ctype_\phi(x, t'_1) = \phi'' \to \phi'$ and $\phi'' \to \phi'$ is a subexpression of ϕ. So by the definition of the hereditary substitution function $[t/x]^\phi t'_1\ t'_2 = [([t/x]^\phi t'_2)/y]^{\phi''} t''_1$, where $[t/x]^\phi t'_1 = \lambda y : \phi''.t''_1$. Hence,

$$|rset([t/x]^\phi t'_1\ t'_2)| = |rset([([t/x]^\phi t'_2)/y]^{\phi''} t''_1)|.$$

Now $\phi >_{\Gamma,\Gamma'} \phi''$ so by the induction hypothesis

$$\begin{array}{lll} |rset([([t/x]^\phi t'_2)/y]^{\phi''} t''_1)| & \leq & |rset([t/x]^\phi t'_2, t''_1)| \\ & \leq & |rset(t'_2, t''_1, t)| \\ & = & |rset(t'_2, [t/x]^\phi t'_1, t)| \\ & \leq & |rset(t'_2, t'_1, t)| \\ & = & |rset(t'_1, t'_2, t)|. \end{array}$$

Suppose $[t/x]^\phi t'_1 = \text{case } t''_0 \text{ of } y.t''_1, y.t''_2$. Then

$$\begin{array}{lll} |rset([t/x]^\phi (t'_1\ t'_2))| & = & |rset(\text{case } t''_0 \text{ of } y.(app_\phi\ t''_1\ [t/x]^\phi t'_2), y.(app_\phi\ t''_2\ [t/x]^\phi t'_2))| \\ & = & |rset(t''_0, (app_\phi\ t''_1\ [t/x]^\phi t'_2), (app_\phi\ t''_2\ [t/x]^\phi t'_2))|. \end{array}$$

We know $t' > t'_1$ and $t' > t'_2$ so by the induction hypothesis

$$\begin{array}{lll} |rset([t/x]^\phi t'_1)| & = & |rset(t''_0, t''_1, t''_2)| \\ & \leq & |rset(t'_1, t)| \end{array}$$

and

$$|rset([t/x]^\phi t'_2)| \leq |rset(t'_2, t)|.$$

By inversion on $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$ there exists a type ϕ'' such that $\Gamma, x : \phi, \Gamma' \vdash t'_1 : \phi'' \to \phi'$. So by Lemma 4, $ctype_\phi(x, t'_1) = \psi$, $\psi \equiv \phi'' \to \phi'$, and ψ is a subexpression of ϕ. Hence, $\phi >_{\Gamma,\Gamma'} \phi''$ and $\phi >_{\Gamma,\Gamma'} \phi'$. At this point we must show the following proposition.

Proposition. For all $\Gamma \vdash t_1 : \phi'' \to \phi'$ and $\Gamma \vdash t_2 : \phi''$ we have $|rset(app_\phi\ t_1\ t_2)| \leq |rset(t_1, t_2)|$.

We prove this by nested induction on the ordering (ϕ, t', t_1) and case split on the structure of t_1.

Case. Suppose t_1 is not a λ-abstraction or a case construct. Then $app_\phi\ t_1\ t_2 = t_1\ t_2$ and $|rset(t_1\ t_2)| = |rset(t_1, t_2)|$. Thus, $|rset(app_\phi\ t_1\ t_2)| \leq |rset(t_1, t_2)|$.

Case. Suppose $t_1 \equiv \lambda y : \phi''.t''_1$. Then $app_\phi\ t_1\ t_2 = [t_2/y]^{\phi''} t''_1$. By inversion on the assumption $\Gamma \vdash t_1 : \phi'' \to \phi'$ we know $\Gamma, y : \phi'' \vdash t''_1 : \phi'$. We know $\phi >_{\Gamma,\Gamma'} \phi''$ so by the outter induction hypothesis

$$\begin{array}{lll} |rset([t_2/y]^{\phi''} t''_1)| & \leq & |rset(t''_1, t_2) \\ & = & |rset(t_1, t_2). \end{array}$$

Thus, $|rset(app_\phi\ t_1\ t_2) \leq |rset(t_1, t_2)|$.

Case. Suppose $t_1 \equiv \text{case } t''_0 \text{ of } y.t''_1, y.t''_2$. Then

$$app_\phi\ t_1\ t_2 = \text{case } t''_0 \text{ of } y.app_\phi\ t''_1\ t_2, y.app_\phi\ t''_2\ t_2.$$

By inversion on the assumption $\Gamma \vdash t_1 : \phi'' \to \phi'$ we know $\Gamma, y : \phi''_1 \vdash t''_1 : \phi'' \to \phi'$ and $\Gamma, y : \phi''_2 \vdash t''_2 : \phi'' \to \phi'$. Since $t_1 > t''_1$ and $t_1 > t''_2$ we can apply the induction hypothesis to obtain

$$|rset(app_\phi\ t''_1\ t_2)| \leq |rset(t''_1, t_2)|$$

and

$$|rset(app_\phi\ t''_2\ t_2)| \leq |rset(t''_2, t_2)|.$$

Suppose t''_0 is not an inject-left or an inject-right term. Then

$$\begin{array}{lll} |rset(t_1, t_2)| & = & |rset(\text{case } t''_0 \text{ of } y.t''_1, y.t''_2, t_2)| \\ & = & |rset(t''_0, t''_1, t''_2, t_2)| \end{array}$$

and

$$\begin{aligned}
|rset(app_\phi\ t_1\ t_2)| &= |rset(\text{case } t''_0 \text{ of } y.app_\phi\ t''_1\ t_2, y.app_\phi\ t''_2\ t_2)|\\
&= |rset(t''_0) \cup rset(app_\phi\ t''_1\ t_2) \cup rset(app_\phi\ t''_2\ t_2)|\\
&\leq |rset(t''_0) \cup rset(t''_1, t_2) \cup rset(t''_2, t_2)|\\
&= |rset(t''_0) \cup rset(t''_1, t''_2, t_2)|\\
&= |rset(t''_0, t''_1, t''_2, t_2)|.
\end{aligned}$$

Therefore, $|rset(app_\phi\ t_1\ t_2)| \leq |rset(t_1, t_2)|$.

Now suppose $t''_0 \equiv inl(\hat{t}_0)$. We only show the case when t''_0 is an inject-left term, because the case when it is an inject-right term is similar. By definition we know

$$\begin{aligned}
|rset(t_1, t_2)| &= |rset(\text{case } t''_0 \text{ of } y.t''_1, y.t''_2, t_2)|\\
&= |\{\text{case } t''_0 \text{ of } y.t''_1, y.t''_2\} \cup rset(t''_0, t''_1, t''_2, t_2)|
\end{aligned}$$

and

$$\begin{aligned}
|rset(app_\phi\ t_1\ t_2)| &= |rset(\text{case } t''_0 \text{ of } y.app_\phi\ t''_1\ t_2, y.app_\phi\ t''_2\ t_2)|\\
&= |\{\text{case } t''_0 \text{ of } y.app_\phi\ t''_1\ t_2, y.app_\phi\ t''_2\ t_2\} \cup rset(t''_0) \cup rset(app_\phi\ t''_1\ t_2) \cup rset(app_\phi\ t''_2\ t_2\\
&\leq |\{\text{case } t''_0 \text{ of } y.app_\phi\ t''_1\ t_2, y.app_\phi\ t''_2\ t_2\} \cup rset(t''_0) \cup rset(t''_1, t_2) \cup rset(t''_2, t_2)|\\
&= |\{\text{case } t''_0 \text{ of } y.app_\phi\ t''_1\ t_2, y.app_\phi\ t''_2\ t_2\} \cup rset(t''_0) \cup rset(t''_1, t''_2, t_2)|\\
&= |\{\text{case } t''_0 \text{ of } y.app_\phi\ t''_1\ t_2, y.app_\phi\ t''_2\ t_2\} \cup rset(t''_0, t''_1, t''_2, t_2)|.
\end{aligned}$$

Therefore, $|rset(app_\phi\ t_1\ t_2)| \leq |rset(t_1, t_2)|$.

Suppose $[t/x]^\phi t'_1$ is not a λ-abstractions or a case construct, or $ctype_\phi(x, t'_1)$ is undefined. Then

$$\begin{aligned}
rset([t/x]^\phi(t'_1\ t'_2)) &= rset([t/x]^\phi t'_1\ [t/x]^\phi t'_2)\\
&= rset([t/x]^\phi t'_1, [t/x]^\phi t'_2).\\
&\leq rset(t'_1, t'_2, t).
\end{aligned}$$

Next suppose $t'_1 \equiv \lambda y : \phi_1.t''_1$. Then

$$rset((\lambda y : \phi_1.t''_1)\ t'_2, t) = \{(\lambda y : \phi_1.t''_1)\ t'_2\} \cup rset(t''_1, t'_2, t).$$

By the definition of the hereditary substitution function,

$$\begin{aligned}
rset([t/x]^\phi(\lambda y : \phi_1.t''_1)\ t'_2) &= rset([t/x]^\phi(\lambda y : \phi_1.t''_1)\ [t/x]^\phi t'_2)\\
&= rset((\lambda y : \phi_1.[t/x]^\phi t''_1)\ [t/x]^\phi t'_2)\\
&= \{(\lambda y : \phi_1.[t/x]^\phi t''_1)\ [t/x]^\phi t'_2\} \cup rset([t/x]^\phi t''_1) \cup rset([t/x]^\phi t'_2).
\end{aligned}$$

Since $t' > t''_1$ and $t' > t'_2$ we can apply the induction hypothesis to obtain, $|rset(t''_1, t)| \geq |rset([t/x]^\phi t''_1)|$ and $|rset(t'_2, t)| \geq |rset([t/x]^\phi t'_2)|$. Therefore,
$|\{(\lambda y : \phi_1.t''_1)\, t'_2\} \cup rset(t''_1, t) \cup rset(t'_2, t)| \geq |\{(\lambda y : \phi_1.[t/x]^\phi t''_1)\ [t/x]^\phi t'_2\} \cup rset([t/x]^\phi t''_1) \cup rset([t/x]^\phi t'_2)|$.

Finally, suppose $t'_1 \equiv \text{case } t''_0 \text{ of } y.t''_1, y.t''_2$. Then

$$\begin{aligned}
|rset([t/x]^\phi(t'_1\ t'_2))| &= |rset((\text{case } [t/x]^\phi t''_0 \text{ of } y.[t/x]^\phi t''_1, y.[t/x]^\phi t''_2)\ [t/x]^\phi t'_2)|\\
&= |\{[t/x]^\phi(t'_1\ t'_2)\} \cup rset([t/x]^\phi t''_0, [t/x]^\phi t''_1, [t/x]^\phi t''_2, [t/x]^\phi t'_2)|.
\end{aligned}$$

Now $t' > t''_0$, $t' > t''_1$, $t' > t''_2$, and $t' > t'_2$ so by the induction hypothesis

$$\begin{aligned}
&|rset([t/x]^\phi t''_0| \leq |rset(t''_0, t)|,\\
&|rset([t/x]^\phi t''_1| \leq |rset(t''_1, t)|,\\
&|rset([t/x]^\phi t''_2| \leq |rset(t''_2, t)|, \text{ and}\\
&|rset([t/x]^\phi t'_2| \leq |rset(t'_2, t)|.
\end{aligned}$$

Hence,

$$|rset([t/x]^\phi t''_0, [t/x]^\phi t''_1, [t/x]^\phi t''_2, [t/x]^\phi t'_2)| \leq |rset(t''_0, t''_1, t''_2, t'_2, t)|.$$

Now

$$\begin{array}{rcl}|rset(t'_1\ t'_2, t)| & = & |rset((\text{case } t''_0 \text{ of } y.t''_1, y.t''_2)\ t'_2, t)| \\ & = & |\{t'_1\ t'_2\} \cup rset(t''_0, t''_1, t''_2, t'_2, t)|.\end{array}$$

Therefore, $|rset([t/x]^{\phi}(t'_1\ t'_2))| \leq |rset(t'_1\ t'_2, t)|$.

Case. Let $t' \equiv$ case t'_0 of $y.t'_1$,$y.t'_2$. Suppose t'_0 is not an inject-left term, and inject-right term, or a case-construct. First we know

$$|rset(t', t)| = |rset(t'_0, t'_1, t'_2, t)|.$$

Now we have several cases to consider, when $[t/x]^{\phi}t'_0$ is an inject-left term, an inject-right term, a case construct, something else entirely, or $ctype_{\phi}(x, t'_0)$ is undefined. Suppose it is something else entirely or $ctype_{\phi}(x, t'_0)$ is undefined. Then

$$\begin{array}{rcl}|rset([t/x]^{\phi}t')| & = & |rset(\text{case } [t/x]^{\phi}t'_0 \text{ of } y.([t/x]^{\phi}t'_1), y.([t/x]^{\phi}t'_2))| \\ & = & |rset([t/x]^{\phi}t'_0, ([t/x]^{\phi}t'_1), ([t/x]^{\phi}t'_2))|.\end{array}$$

We can see that $t' > t'_0$, $t' > t'_1$, $t' > t'_2$ so by the induction hyothesis

$$\begin{array}{l}|rset([t/x]^{\phi}t'_0| \leq |rset(t'_0, t)|, \\ |rset([t/x]^{\phi}t'_1| \leq |rset(t'_1, t)|, \text{ and} \\ |rset([t/x]^{\phi}t'_2| \leq |rset(t'_2, t)|.\end{array}$$

This implies that $|rset([t/x]^{\phi}t'_0, ([t/x]^{\phi}t'_1), ([t/x]^{\phi}t'_2))| \leq |rset(t'_0, t'_1, t'_2, t)|$. Therefore, $|rset(t', t)| \geq |rset([t/x]^{\phi}t')$.

Now suppose $[t/x]^{\phi}t'_0 \equiv inl(t''_0)$. We only show the case for when $[t/x]^{\phi}t'_0$ is an inject-left term, because the case for when it is an inject-right term is similar. We can see that

$$\begin{array}{rcl}|rset(t', t)| & = & |rset(\text{case } t'_0 \text{ of } y.t'_1, y.t'_2, t)| \\ & = & |rset(t'_0, t'_1, t'_2, t)|\end{array}$$

and $|rset([t/x]^{\phi}t')| = |rset([t''_0/y]^{\phi_1}([t/x]^{\phi}t'_1))|$. By inversion on $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ we know there exists types ϕ_1 and ϕ_2 such that $\Gamma, x : \phi, \Gamma' \vdash t_0 : \phi_1 + \phi_2$. So by Lemma 4 there exists a type ψ such that $ctype_{\phi}(x, t'_0) = \psi$, $\psi \equiv \phi_1 + \phi_2$, and ψ is a subexpression of ϕ. Thus, $\phi >_{\Gamma,\Gamma'} \phi_1$, $\phi >_{\Gamma,\Gamma'} \phi_2$, $t > t'_0$, and $t > t'_1$ so we can apply the induction hypothesis to obtain

$$\begin{array}{rcl}|rset([t''_0/y]^{\phi_1}([t/x]^{\phi}t'_1))| & \leq & |rset(t''_0, [t/x]^{\phi}t'_1)| \\ & = & |rset([t/x]^{\phi}t'_0, [t/x]^{\phi}t'_1)| \\ & \leq & |rset(t'_0, t'_1, t)| \\ & \leq & |rset(t'_0, t'_1, t'_2, t)|.\end{array}$$

Next suppose $[t/x]^{\phi}t'_0 \equiv$ case t''_0 of $y.t''_1$,$y.t''_2$. Then

$$|rset([t/x]^{\phi}t')| = |rset(rcase_{\phi}\ [t/x]^{\phi}t'_0\ y\ [t/x]^{\phi}t'_1\ [t/x]^{\phi}t'_2)|$$

and

$$\begin{array}{rcl}|rset(t', t)| & = & |rset(\text{case } t'_0 \text{ of } y.t'_1, y.t'_2, t)| \\ & = & |rset(t'_0, t_1, t'_2, t)|.\end{array}$$

Note that by inverision on $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ we know there exists types ϕ_1 and ϕ_2 such that $\Gamma, x : \phi, \Gamma' \vdash t'_0 : \phi_1 + \phi_2$. So by Lemma 4 there exists a type ψ such that $ctype_{\phi}(x, t'_0) = \psi$, $\psi \equiv \phi_1 + \phi_2$, and ψ is a subexpression of ϕ. Thus, $\phi >_{\Gamma,\Gamma'} \phi_1$ and $\phi >_{\Gamma,\Gamma'} \phi_2$. It suffices to show that $|rset(rcase_{\phi}\ [t/x]^{\phi}t'_0\ [t/x]^{\phi}t'_1\ [t/x]^{\phi}t'_2)| \leq |rset([t/x]^{\phi}t'_0, [t/x]^{\phi}t'_1, [t/x]^{\phi}t'_2)|$ which is a consequence of the following proposition.

Proposition. For all $\Gamma \vdash t : \phi_1 + \phi_2$, $\Gamma, y : \phi_1 \vdash t'_1 : \phi'$, and $\Gamma, y : \phi_2 \vdash t'_2 : \phi'$ we have $|rset(rcase_{\phi}\ t\ y\ t'_1\ t'_2)| \leq |rset(t, t'_1, t'_2)|$.

We prove this proposition by nested induction on (ϕ, t', t) and we case split on t.

Case. Suppose $t \equiv inl(t')$. Then

$$rset(rcase_{\phi}\ t\ y\ t'_1\ t'_2) = rset([t'/y]^{\phi_1}t'_1).$$

By inversion on $\Gamma \vdash t : \phi_1 + \phi_2$ we know $\Gamma \vdash t' : \phi_1$. So by the outer induction hypothesis

$$\begin{array}{lcl} |rset([t'/y]^{\phi_1}t'_1)| & \leq & |rset(t'_1, t')| \\ & = & |rset(t, t'_1)| \\ & \leq & |rset(t, t'_1, t'_2)|. \end{array}$$

Therefore, $|rset(rcase_\phi\ t\ y\ t'_1\ t'_2)| \leq |rset(t, t'_1, t'_2)|$.

Case. Suppose $t \equiv inr(t')$. This case is similar to the previous case.

Case. Suppose $t \equiv \text{case}\ t_0\ \text{of}\ z.t_1, z.t_2$. Then

$$\text{rcase}_\phi\ t\ y\ t'_1\ t'_2 = \text{case}\ t_0\ \text{of}\ z.(\text{rcase}_\phi\ t_1\ y\ t'_1\ t'_2), z.(\text{rcase}_\phi\ t_2\ y\ t'_1\ t'_2).$$

Now $t > t'_i$ for $i \in \{0, 1, 2\}$. Before we can apply the inductive hypothesis we must show that t_1 and t_2 are typeable. By inversion on the assumption $\Gamma \vdash t : \phi_1 + \phi_2$ we know $\Gamma, z : \phi'_1 \vdash t_1 : \phi_1 + \phi_2$ and $\Gamma, z : \phi'_2 \vdash t_2 : \phi_1 + \phi_2$. So by the inner induction hypothesis $|rset(\text{rcase}_\phi\ t_i\ y\ t'_1\ t'_2)| \leq |rset(t_i, t'_1, t'_2)|$.

We have two cases to consider either t_0 is not an inject-left term or an inject-right term, or it is. If not then

$$rset(\text{rcase}_\phi\ t\ y\ t'_1\ t'_2) = rset(t_0, \text{rcase}_\phi\ t_1\ y\ t'_2\ t'_2, \text{rcase}_\phi\ t_2\ y\ t'_2\ t'_2)$$

otherwise

$$\begin{array}{l} rset(\text{rcase}_\phi\ t\ y\ t'_1\ t'_2) \\ \quad = \{\text{rcase}_\phi\ t\ y\ t'_1\ t'_2\} \cup rset(t_0, \text{rcase}_\phi\ t_1\ y\ t'_1\ t'_2, \text{rcase}_\phi\ t_2\ y\ t'_1\ t'_2). \end{array}$$

Suppose t_0 is not an inject-left or an inject-right term. Then

$$|rset(t, t'_1, t'_2)| = |rset(t_0, t_1, t_2, t'_1, t'_2)|.$$

Now we know

$$\begin{array}{l} |rset(t_0, \text{rcase}_\phi\ t_1\ y\ t'_2\ t'_2, \text{rcase}_\phi\ t_2\ y\ t'_2\ t'_2)| \\ \quad = |rset(t_0) \cup rset(\text{rcase}_\phi\ t_1\ y\ t'_2\ t'_2) \cup rset(\text{rcase}_\phi\ t_2\ y\ t'_2\ t'_2)| \\ \quad \leq |rset(t_0) \cup rset(t_1, t'_2, t'_2) \cup rset(t_1, t'_2, t'_2)| \\ \quad = |rset(t_0, t_1, t_2, t'_2, t'_2)|. \end{array}$$

Therefore, $|rset(\text{rcase}_\phi\ t\ y\ t'_1\ t'_2)| \leq |rset(t, t'_1, t'_2)|$.

Now suppose $t_0 \equiv inl(t'_0)$. Then

$$|rset(t, t'_1, t'_2)| = |\{t\} \cup rset(t_0, t_1, t_2, t'_1, t'_2)|.$$

It suffices to show that

$$\begin{array}{l} |\{\text{rcase}_\phi\ t\ y\ t'_1\ t'_2\} \cup rset(t_0, \text{rcase}_\phi\ t_1\ y\ t'_1\ t'_2, \text{rcase}_\phi\ t_2\ y\ t'_1\ t'_2)| \\ \quad \leq |\{t\} \cup rset(t_0, t_1, t_2, t'_1, t'_2)|. \end{array}$$

Let $A = \text{rcase}_\phi\ t_1\ y\ t'_1\ t'_2$ and $B = \text{rcase}_\phi\ t_2\ y\ t'_1\ t'_2$. Since $|rset(\text{rcase}_\phi\ t_i\ y\ t'_1\ t'_2)| \leq |rset(t_i, t'_1, t'_2)|$ we obtain the following:

$$\begin{array}{l} |\{\text{rcase}_\phi\ t\ y\ t'_1\ t'_2\} \cup rset(t_0, A, B)| \\ \quad = |\{\text{rcase}_\phi\ t\ y\ t'_1\ t'_2\}| + |rset(t_0)| + |rset(A)| + |rset(B)| \\ \quad \leq |\{\text{rcase}_\phi\ t\ y\ t'_1\ t'_2\}| + |rset(t_0)| + |rset(t_1, t'_1, t'_2)| + |rset(t_2, t'_1, t'_2)| \\ \quad = |\{\text{rcase}_\phi\ t\ y\ t'_1\ t'_2\}| + |rset(t_0, t_1, t_2, t'_1, t'_2)| \\ \quad = |\{t\} \cup rset(t_0, t_1, t_2, t'_1, t'_2)|. \end{array}$$

The case when t_0 is an inject-right term is similar to the case when it is an inject-left term.

Now by the previous proposition we know

$$|rset(rcase_\phi\ [t/x]^\phi t'_0\ [t/x]^\phi t'_1\ [t/x]^\phi t'_2)| \leq |rset([t/x]^\phi t'_0, [t/x]^\phi t'_1, [t/x]^\phi t'_2)|,$$

becuase by Lemma 5 t'_0, t'_1, and t'_2 have the same types as $[t/x]^\phi t'_0$, $[t/x]^\phi t'_1$, and $[t/x]^\phi t'_2$. Now $t' > t'_0$, $t' > t'_1$, and $t' > t'_2$, so

$$\begin{array}{l} |rset([t/x]^\phi t'_0)| \leq |rset(t'_0, t)|, \\ |rset([t/x]^\phi t'_1)| \leq |rset(t'_1, t)|, \text{ and} \\ |rset([t/x]^\phi t'_2)| \leq |rset(t'_2, t)|. \end{array}$$

Thus,

$$\begin{array}{lcl} |rset([t/x]^\phi t'_0, [t/x]^\phi t'_1, [t/x]^\phi t'_2)| & \leq & |rset(t'_0, t'_1, t'_2, t)| \\ & = & |rset(t', t)|. \end{array}$$

Suppose $t'_0 \equiv inl(t''_0)$. Again, we only show the case for when t'_0 is an inject-left term. We know

$$\begin{array}{rcl} |rset([t/x]^{\phi}t')| & = & |rset(\text{case } [t/x]^{\phi}t'_0 \text{ of } y.[t/x]^{\phi}t'_1, y.[t/x]^{\phi}t'_2)| \\ & = & |rset(\text{case } inl([t/x]^{\phi}t''_0) \text{ of } y.[t/x]^{\phi}t'_1, y.[t/x]^{\phi}t'_2)| \\ & = & |\{[t/x]^{\phi}t'\} \cup rset([t/x]^{\phi}t''_0, [t/x]^{\phi}t'_1, [t/x]^{\phi}t'_2)| \end{array}$$

and

$$\begin{array}{rcl} |rset(t',t)| & = & |rset(\text{case } t'_0 \text{ of } y.t'_1, y.t'_2, t)| \\ & = & |rset(\text{case } inl(t''_0) \text{ of } y.t'_1, y.t'_2, t)| \\ & = & |\{t'\} \cup rset(t''_0, t'_1, t'_2)|. \end{array}$$

Now $t' > t''_0$, $t' > t'_1$, and $t' > t'_2$ so by the induction hypothesis

$$\begin{array}{rcl} |rset([t/x]^{\phi}t''_0)| & \leq & |rset(t''_0, t)| \\ |rset([t/x]^{\phi}t'_1)| & \leq & |rset(t'_1, t)| \\ |rset([t/x]^{\phi}t'_2)| & \leq & |rset(t'_2, t)|. \end{array}$$

Therefore, $|rset([t/x]^{\phi}t''_0, [t/x]^{\phi}t'_1, [t/x]^{\phi}t'_2)| \leq |rset(t''_0, t'_1, t'_2, t)|$ which implies that $|rset([t/x]^{\phi}t')| \leq |rset(t',t)|$.

Finally, suppose $t'_0 \equiv \text{case } t''_0 \text{ of } z.t''_1, z.t''_2$. Then

$$\begin{array}{rcl} |rset([t/x]^{\phi}t')| & = & |rset(\text{case } [t/x]^{\phi}t'_0 \text{ of } y.[t/x]^{\phi}t'_1, y.[t/x]^{\phi}t'_2)| \\ & = & |rset(\text{case case } [t/x]^{\phi}t''_0 \text{ of } z.[t/x]^{\phi}t''_0 t''_1, z.[t/x]^{\phi}t''_0 t''_2 \text{ of } y.[t/x]^{\phi}t'_1, y.[t/x]^{\phi}t'_2)| \\ & = & |\{[t/x]^{\phi}t'\} \cup rset([t/x]^{\phi}t'_0, [t/x]^{\phi}t'_1, [t/x]^{\phi}t'_2)| \end{array}$$

and

$$\begin{array}{rcl} |rset(t',t)| & = & |rset(\text{case } t'_0 \text{ of } y.t'_1, y.t'_2, t)| \\ & = & |rset(\text{case case } t''_0 \text{ of } z.t''_1, z.t''_2 \text{ of } y.t'_1, y.t'_2, t)| \\ & = & |\{t'\} \cup rset(t'_0, t'_1, t'_2)|. \end{array}$$

Now $t' > t''_0$, $t' > t'_1$, and $t' > t'_2$ so by the induction hypothesis

$$\begin{array}{rcl} |rset([t/x]^{\phi}t'_0)| & \leq & |rset(t'_0, t)|, \\ |rset([t/x]^{\phi}t'_1)| & \leq & |rset(t'_1, t)|, \text{ and} \\ |rset([t/x]^{\phi}t'_2)| & \leq & |rset(t'_2, t)|. \end{array}$$

Therefore, $|rset([t/x]^{\phi}t''_0, [t/x]^{\phi}t'_1, [t/x]^{\phi}t'_2)| \leq |rset(t''_0, t'_1, t'_2, t)|$ which implies that $|rset([t/x]^{\phi}t')| \leq |rset(t',t)|$.

A.14 Proof of Normality Preservation

By Lemma 5 we know there exists a term n'' such that $[n/x]^{\phi}n' = t$ and by Lemma 8 $|rset(n',n)| \geq |rset([n/x]^{\phi}n')|$. Hence, $|rset(n',n)| \geq |rset(t)|$, but $|rset(n',n)| = 0$. Therefore, $|rset(t)| = 0$ which implies n'' has no redexes. It suffices to show that n'' has no structural redexes. We prove this by induction on the lexicographic ordering (ϕ, n'). We case split on the structure of n'.

Case. Suppose n' is a variable x or y distinct from x. Trivial in both cases.

Case. Suppose $n' \equiv \lambda y : \phi''.\hat{n'}$. Then $[n/x]^{\phi}n' = \lambda y : \phi''.[t/x]^{\phi}\hat{n'}$. By inversion on the assumption $\Gamma, x : \phi' \vdash n' : \phi'$ we know $\Gamma, x : \phi', \Gamma', y : \phi'' \vdash \hat{n'} : \phi'$. Since $n' > \hat{n}$ we can apply the induction hypothesis to obtain there exists a term t' such that $[t/x]^{\phi}\hat{n} = t'$ and t' has no structural redexes. Therefore, neither does $\lambda y : \phi''.[t/x]^{\phi}\hat{n'}$.

Case. Suppose $n' \equiv \Lambda X : *_l.\hat{n}$. Similar to the previous case.

Case. Suppose $n' \equiv inl(n'_0)$. Similar to the λ-abstraction case.

Case. Suppose $n' \equiv n'_1\ n'_2$. By inversion we know $\Gamma, x : \phi, \Gamma' \vdash n'_1 : \phi'' \to \phi'$ and $\Gamma, x : \phi, \Gamma' \vdash n'_2 : \phi''$ for some types ϕ' and ϕ''. Clearly, $n' > n'_i$ for $i \in \{1,2\}$. Thus, by the induction hypothesis there exists normal terms m_1 and m_2 such that $[n/x]^{\phi}n'_i = m_i$ such that m_i have no structural redexes. We case split on whether or not m_1 is a λ-abstraction or a case construct and n'_1 is not, or $ctype_{\phi}(x, n'_1)$ is undefined. We only consider the non-trivial cases when $m_1 \equiv \lambda y : \phi''.m'_1$

or $m_1 \equiv$ case m'_0 of $y.m'_1$,$y.m'_2$ and n'_1 is not a λ-abstraction or a case construct. Suppose the former. Now by Lemma 4 there exists a type ψ such that $ctype_\phi(x, n'_1) = \psi$, $\psi \equiv \phi'' \to \phi'$, and ψ is a subexpression of ϕ, hence $\phi >_{\Gamma,\Gamma'} \phi''$. So $[n/x]^\phi(n'_1\ n'_2) = [m_2/y]^{\phi''} m'_1$ and by the induction hypothesis there exists a term m such that $[m_2/y]^{\phi''} m'_1 = m$ and m has no structural redexes..

Suppose $m_1 \equiv$ case m'_0 of $y.m'_1$,$y.m'_2$. By inversion on $\Gamma,\Gamma' \vdash m_1 : \phi'' \to \phi'$ we know there exists terms ϕ_1 and ϕ_2 such that $\Gamma,\Gamma' \vdash m'_0 : \phi_1 + \phi_2$ and $\Gamma,\Gamma',y : \phi_i \vdash m'_i : \phi'' \to \phi'$ for $i \in \{1,2\}$. Note that by Lemma 4 there exists a type ψ such that $ctype_\phi(x,n'_1) = \psi$, $\psi \equiv \phi'' \to \phi'$, and ψ is a subexpression of ϕ, hence $\phi >_{\Gamma,\Gamma'} \phi'$ and $\phi >_{\Gamma,\Gamma'} \phi''$. Now $[t/x]^\phi t' =$ case m'_0 of $y.app_\phi\ m'_1\ m_2$,$y.app_\phi\ m'_2\ m_2$. It suffcies to show that there exists terms q and q' such that $app_\phi\ m'_1\ m_2 = q$, $app_\phi\ m'_2\ m_2 = q'$ and q and q' have no structural redexes. To obtain this result we prove the following proposition.

Proposition. For all normal terms m_2 and m'_1 such that $\Gamma \vdash m_2 : \phi''$ and $\Gamma \vdash m'_1 : \phi'' \to \phi'$ there exists a term q such that $app_\phi\ m'_1\ m_2 = q$ and q has no structural redexes.

We prove this by nested induction on the ordering (ϕ, n', m'_1) and case splitting on the structure of m'_1.

Case. Suppose m'_1 is neither a λ-abstraction or a case construct. Then $app_\phi\ m'_1\ m_2 = m'_1\ m_2$. Take $m'_1\ m_2$ for q and we know q has no structural redexes, because m'_1 and m_2 are normal.

Case. Suppose $m'_1 \equiv \lambda z : \phi''.m''_1$. Then $app_\phi\ m'_1\ m_2 = [m_2/z]^{\phi''} m''_1$. By inversion on the assumption $\Gamma \vdash m'_1 : \phi'' \to \phi'$ we know $\Gamma, z : \phi'' \vdash m''_1 : \phi'$. Since $\phi >_\Gamma \phi''$ we can apply the outter induction hypothesis to obtain there there exists a q such that $[m_2/z]^{\phi''} m''_1 = q$ and q has no structural redexes.

Case. Suppose $m'_1 \equiv$ case m''_0 of $z.m''_1$,$z.m''_2$. Then $app_\phi\ m'_1\ m_2 =$ case m''_0 of $z.app_\phi\ m''_1\ m_2$,$z.app_\phi\ m''_2\ m_2$. By inversion on the assumption $\Gamma \vdash m'_1 : \phi'' \to \phi'$ we know there exists types ϕ_1 and ϕ_2 such that $\Gamma \vdash m''_0 : \phi_1 + \phi_2$, $\Gamma, z : \phi_1 \vdash m''_1 : \phi'' \to \phi'$ and $\Gamma, z : \phi_2 \vdash m''_2 : \phi'' \to \phi'$. Since $m'_1 > m''_1$ and $m'_1 > m''_2$ we can apply the inner induction hypothesis to obtain there exists terms q' and q'' such that $app_\phi\ m''_1\ m_2 = q'$, q' has no structural redexes, $app_\phi\ m''_1\ m_2 = q''$ and q'' has no structural redexes. Hence, $app_\phi\ m'_1\ m_2 =$ case m''_0 of $z.app_\phi\ m''_1\ m_2$,$z.app_\phi\ m''_2\ m_2 =$ case m''_0 of $z.q'$,$z.q''$ and case m''_0 of $z.q'$,$z.q''$ has no structural redexes. Note that m''_0 is normal, because m'_1 is normal.

By the previous proposition there exists terms q and q' such that
$[n/x]^\phi n' =$ case m'_0 of $y.app_\phi\ m'_1\ m_2$,$y.app_\phi\ m'_2\ m_2 =$ case m'_0 of $y.q$,$y.q'$, where $app_\phi\ m'_1\ m_2 = q$, $app_\phi\ m'_1\ m_2 = q'$, and q and q' have no structural redexes. Thus, case m'_0 of $y.q$,$y.q'$ has no structural redexes.

Case. Suppose $n' \equiv$ case m_0 of $y.m_1$,$y.m_2$. By inversion on the assumption $\Gamma, x : \phi, \Gamma' \vdash n' : \phi'$ we know the following:

$$\begin{array}{l} \Gamma, x : \phi, \Gamma' \vdash m_0 : \phi_1 + \phi_2, \text{for some types } \phi_1 \text{ and } \phi_2, \\ \Gamma, x : \phi, \Gamma', y : \phi_1 \vdash m_1 : \phi, and \\ \Gamma, x : \phi, \Gamma', y : \phi_2 \vdash m_2 : \phi. \end{array}$$

It is easy to see that $n' > m_i$ for all $i \in \{0,1,2\}$. Hence, by the induction hypothesis there exists terms m'_0, m'_1, and m'_2 such that $[t/x]^\phi m_i = m'_i$ and m'_i have no structural redexes for all $i \in \{0,1,2\}$. We have two cases to consider.

Case. Suppose m'_0 is not an inject-left term, inject-right term, or case construct, or m_0 is an inject-left term, an inject-right term, or a case construct, or $ctype_\phi(x, m_0)$ is undefined. Then $[n/x]^\phi(\text{case } m_0 \text{ of } y.m_1,y.m_2) = \text{case } m'_0 \text{ of } y.m'_1,y.m'_2$ which has no structural redexes.

Case. Suppose m'_0 is an inject-left term, inject-right term, or case construct and m_0 is not an inject-left term, an inject-right term, or a case construct. Then $[n/x]^\phi(\text{case } m_0 \text{ of } y.m_1,y.m_2) = rcase_\phi\ m'_0\ y\ m'_1\ m'_2$, where by Lemma 4 there exists a type ψ such that $ctype_\phi(x, m_0) = \psi$, $\psi \equiv \phi_1 + \phi_2$, and ψ is a subexpression of ϕ, hence $\phi >_{\Gamma,\Gamma'} \phi_1$ and $\phi >_{\Gamma,\Gamma'} \phi_2$. Consider the case when $m'_0 \equiv inl(m''_0)$. Then we know by the definition of $rcase$ that $rcase_\phi\ m'_0\ y\ m'_1\ m'_2 = [m''_0/y]^{\phi_1} m'_1$. Clearly, $\phi >_{\Gamma,\Gamma'} \phi_1$ hence by the the induction hypothesis there exists a term r such that $[m''_0/y]^{\phi_1} m'_1 = r$ and r has no structural redexes. Similarly for when $m'_0 \equiv inr(m''_0)$. So suppose $m'_0 \equiv \text{case } m''_0 \text{ of } z.m''_1,z.m''_2$ then it suffices to show that there exists some term q such that $rcase_\phi\ m'_0\ y\ m'_1\ m'_2 = q$ and q has no structural redexes. We obtain this result by the following proposition.

Proposition. For all normal terms q and q_1 such that $\Gamma \vdash q_0 : \phi$, $\Gamma, y : \phi_1 \vdash q_1 : \phi'$, and $\Gamma, y : \phi_2 \vdash q_2 : \phi'$ there exists a term $\hat{q}$ such that $rcase_\phi\ q_0\ y\ q_1\ q_2 = \hat{q}$ and $\hat{q}$ has no structural redexes. We prove this by induction on the the ordering (ϕ, n', q_0) and case split on the structure of q_0.

Case. Suppose q_0 is not an inject-left term, inject-right term, or a case construct. Then $rcase_\phi\ q_0\ y\ q_1\ q_2 = \text{case } q_0 \text{ of } y.q_1,y.q_2$ which has no structural redexes.

Case. Suppose $q_0 \equiv inl(q'_0)$. Then $rcase_\phi\ q_0\ y\ q_1\ q_2 = [q'_0/y]^{\phi_1} q_1$ and by inversion on $\Gamma \vdash q_0 : \phi$ we know $\Gamma \vdash q'_0 : \phi_1$. It suffices to show that there exists a term $\hat{q}$ such that $[q'_0/y]^{\phi_1} q_1 = \hat{q}$ and $\hat{q}$ has no structural redexes. Clearly, $\phi >_\Gamma \phi'$ so by the outer induction hypothesis there exists such a term $\hat{q}$.

Case. Suppose $q_0 \equiv inl(q'_0)$. Similar to the previous case.

Case. Suppose $q_0 \equiv \text{case } q'_0 \text{ of } z.q'_1,z.q'_2$. Then $rcase_\phi\ q_0\ y\ q_1\ q_2 = \text{case } q'_0 \text{ of } z.(rcase_\phi\ q'_1\ y\ q_1\ q_2),z.(rcase_\phi\ q'_2\ y\ q_1\ q_2)$. We know by assumption that $\Gamma \vdash q_0 : \phi$, $\Gamma \vdash q_0 : \phi$, and $\Gamma, y : \phi_1 \vdash q_1 : \phi'$ so by inversion we know the following:

(i) $\Gamma \vdash q'_0 : \phi'_1 + \phi'_2$, for some types ϕ'_1 and ϕ'_2,
(ii) $\Gamma, z : \phi'_1 \vdash q'_1 : \phi$, and
(iii) $\Gamma, z : \phi'_2 \vdash q'_2 : \phi$.

Now $q_0 > q'_1$ and $q_0 > q'_1$ so we can apply the inner induction hypothesis twice to obtain terms $\hat{q}_1$ and $\hat{q}_2$ such that $rcase_\phi\ q'_1\ y\ q_1\ q_2 = \hat{q}_1$, $rcase_\phi\ q'_1\ y\ q_1\ q_2 = \hat{q}_1$ where $\hat{q}_1$ and $\hat{q}_2$ have no structural redexes. So $\text{case } q'_0 \text{ of } z.(rcase_\phi\ q'_1\ y\ q_1\ q_2),z.(rcase_\phi\ q'_2\ y\ q_1\ q_2) = \text{case } q'_0 \text{ of } z.\hat{q}_1,z.\hat{q}_2$. It suffices to show that $\text{case } q'_0 \text{ of } z.\hat{q}_1,z.\hat{q}_2 = \hat{q}$ for some normal term $\hat{q}$. Now $q_0 > q'_0$ so we can apply the induction hypothesis to obtain our result, but before we can we must show that $\Gamma \vdash \text{case } q'_0 \text{ of } z.\hat{q}_1,z.\hat{q}_2 : \phi'$. This is a direct consequence of applying the case-construct typing rule using i, $\Gamma, z : \phi'_1 \vdash \hat{q}_1 : \phi'$ and $\Gamma, z : \phi'_2 \vdash \hat{q}_2 : \phi'$. Therefore, by the inner induction hypothesis there exists a term $\hat{q}$ such that $\text{case } q'_0 \text{ of } z.\hat{q}_1,z.\hat{q}_2 = \hat{q}$ and $\hat{q}$ is has no structural redexes.

Case. Suppose $n' \equiv n'_1[\phi'']$. Since $n' > n'_1$ we can apply the induction hypothesis to obtain $[n'/x]^\phi n'_1$ has no structural redexes. We case split on whether or not $[n'/x]^\phi n'_1$ is a type abstraction and n'_1 is not. The case where it is not is trivial so we only consider the case where $[n'/x]^\phi n'_1 \equiv \Lambda X : *_l.s'$ for some normal term s'. Then $[n'/x]^\phi n' = [\phi'/X]s'$ has no structural redexes, because s' is normal.

A.15 Proof of Soundness with Respect to Reduction

This is a proof by induction on the lexicorgraphic combination (ϕ, t') of $>_\Gamma$ and the strict subexpression ordering. We case split on the structure of t'. When applying the induction hypothesis we must show that the input terms to the substitution and the hereditary substitution functions are typeable. We do not explicitly state typing results that are simple conseqences of inversion.

Case. Suppose t' is a variable x or y distinct from x. Trivial in both cases.

Case. Suppose $t' \equiv \lambda y : \phi'.\hat{t}$. Then $[t/x]^\phi(\lambda y : \phi'.\hat{t}) = \lambda y : \phi'.([t/x]^\phi\hat{t})$. Now $t' > \hat{t}$ so we can apply the induction hypothesis to obtain $[t/x]\hat{t} \rightsquigarrow^* [t/x]^\phi\hat{t}$. At this point we can see that since $\lambda y : \phi'.[t/x]\hat{t} \equiv [t/x](\lambda y : \phi'.\hat{t})$ and we may conclude that $\lambda y : \phi'.[t/x]\hat{t} \rightsquigarrow^* \lambda y : \phi'.[t/x]^\phi\hat{t}$.

Case. Suppose $t' \equiv \Lambda X : *_l.\hat{t}$. Similar to the previous case.

Case. Suppose $t' \equiv inl(t'_0)$. Then $[t/x]^\phi t' = inl([t/x]^\phi t'_0)$. We can see that $t' > t'_0$ so by the induction hypothesis $[t/x]t'_0 \rightsquigarrow^* [t/x]^\phi t'_0$. Hence, $inl([t/x]t'_0) \rightsquigarrow^* inl([t/x]^\phi t'_0)$ which implies that $[t/x](inl(t'_0)) \rightsquigarrow^* [t/x]^\phi(inl(t'_0))$.

Case. Suppose $t' \equiv inr(t'_0)$. Similar to the previous case.

Case. Suppose $t' \equiv \text{case } t'_0 \text{ of } y.t'_1,y.t'_2$. Clearly, $t' > t'_0$, $t' > t'_1$, and $t' > t'_2$, so we can apply the induction hypothesis to conclude $[t/x]t'_0 \rightsquigarrow^* [t/x]^\phi t'_0$, $[t/x]t'_1 \downarrow [t/x]^\phi t'_1$, and $[t/x]t'_2 \rightsquigarrow^* [t/x]^\phi t'_2$. We have several cases to consider, either when $[t/x]^\phi t'_0$ is an inject-left term or an inject-right term and t'_0 is not, when $[t/x]^\phi t'_0$ is a case construct and t'_0 is not, or $[t/x]^\phi t'_0$ is not an inject-left term, an inject-right term, or a case construct, or $ctype_\phi(x, t'_0)$ is undefined. The cases when $[t/x]^\phi t'_0$ is not an inject-left term, an inject-right term, or a case construct, or $ctype_\phi(x, t'_0)$ is undefined are trivial.

Let's consider the case when $[t/x]^\phi t'_0$ is an inject-left term or an inject-right term and t'_0 is not. Since the case when $[t/x]^\phi t'_0$ is an inject-left term is similar to the case when it is an inject-right term we only consider the former. Suppose $[t/x]^\phi t'_0 = inl(t''_0)$ and t'_0 is not an inject-left term. By Lemma 4 there exists a type ψ such that $ctype_\phi(x, t'_0) = \psi$, $\psi \equiv \phi_1 + \phi_2$, and ψ is a subexpression of ϕ, where by inversion on $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ there exists types ϕ_1 and ϕ_2 such that $\Gamma, x : \phi, \Gamma' \vdash t'_0 : \phi_1 + \phi_2$. Thus, $\phi >_{\Gamma,\Gamma'} \phi_1$ and $\phi >_{\Gamma,\Gamma'} \phi_2$. So $[t/x]^\phi t' = [t''_0/y]^{\phi_1}([t/x]^\phi t'_1)$ and we know from above that $[t/x]t'_1 \rightsquigarrow^* [t/x]^\phi t'_1$. Now $\phi >_{\Gamma,\Gamma'} \phi_1$, so by the induction hypothesis, $[t''_0/y]([t/x]^\phi t'_1) \rightsquigarrow^* [t''_0/y]^{\phi_1}([t/x]^\phi t'_1)$. Thus, $[t''_0/y]([t/x]t'_1) \rightsquigarrow^* [t''_0/y]^{\phi_1}([t/x]^\phi t'_1)$. It suffices to show $[t/x]t' \rightsquigarrow^* [t''_0/y]([t/x]t'_1)$. We can see that

$$\begin{array}{rcl} [t/x]t' & = & [t/x](\text{case } x \text{ of } y.t'_1,y.t'_2) \\ & \equiv & \text{case } [t/x]x \text{ of } y.[t/x]t'_1,y.[t/x]t'_2 \\ & \equiv & \text{case } inl(t''_0) \text{ of } y.[t/x]t'_1,y.[t/x]t'_2 \\ & \rightsquigarrow & [t''_0/y]([t/x]t'_1). \end{array}$$

Suppose $[t/x]t'_0 = \text{case } t''_0 \text{ of } z.t''_1,z.t''_2$ and t'_0 is not. It suffices to show that $[t/x]t \rightsquigarrow^* [t/x]^\phi t'$, which is equivalent to showing $[t/x](\text{case } t'_0 \text{ of } y.t'_1,y.t'_2) \rightsquigarrow^* [t/x]^\phi(\text{case } t'_0 \text{ of } y.t'_1,y.t'_2)$. Now

$$[t/x]^\phi(\text{case } t'_0 \text{ of } y.t'_1,y.t'_2) \;=\; \text{case } t''_0 \text{ of } z.(\text{rcase}_\phi\ t''_1\ y\ t'_1\ t'_2),z.(\text{rcase}_\phi\ t''_1\ y\ t'_1\ t'_2)$$

and

$$\begin{array}{rcl} [t/x](\text{case } t'_0 \text{ of } y.t'_1,y.t'_2) & = & \text{case } [t/x]t'_0 \text{ of } y.[t/x]t'_1,y.[t/x]t'_2 \\ & \rightsquigarrow^* & \text{case (case } t''_0 \text{ of } z.t''_1,z.t''_2) \text{ of } y.[t/x]t'_1,y.[t/x]t'_2 \\ & \rightsquigarrow & \text{case } t''_0 \text{ of } z.(\text{case } t''_1 \text{ of } y.t'_1,y.t'_2),z.(\text{case } t''_2 \text{ of } y.t'_1,y.t'_2), \end{array}$$

because we know from above that $[t/x]t'_0 \rightsquigarrow^* [t/x]^\phi t'_0$. So it suffices to show that $(\text{case } t''_1 \text{ of } y.t'_1,y.t'_2) \rightsquigarrow^* (\text{rcase}_\phi\ t''_1\ y\ t'_1\ t'_2)$ and $(\text{case } t''_2 \text{ of } y.t'_1,y.t'_2) \rightsquigarrow^* (\text{rcase}_\phi\ t''_2\ y\ t'_1\ t'_2)$, because we know from above that $[t/x]t_i \rightsquigarrow^* [t/x]^\phi t'_i$. This is a consequence of the following proposition. First note that again by Lemma 4 there exists a type ψ such that $ctype_\phi(x, t'_0) = \psi$, $\psi \equiv \phi_1 + \phi_2$, and ψ

is a subexpression of ϕ, where by inversion on the assumption $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ there exists types ϕ_1 and ϕ_2 such that $\Gamma, x : \phi, \Gamma' \vdash t'_0 : \phi_1 + \phi_2$. Hence, $\phi >_{\Gamma,\Gamma'} \phi_1$ and $\phi >_{\Gamma,\Gamma'} \phi_2$.

Proposition. For all $\Gamma \vdash t_0 : \phi_1 + \phi_2$, $\Gamma, y : \phi_1 \vdash t_1 : \phi''$ and $\Gamma, y : \phi_2 \vdash t_2 : \phi''$ we have (case t_0 of $y.t_1$,$y.t_2$) $\rightsquigarrow^*$ (rcase$_\phi$ $t_0\ y\ t_1\ t_2$).

We prove this by nested induction on the ordering (ϕ, t', t_0) and case splitting on the structure of t_0.

Case. Suppose t_0 is not an inject-left term, an inject-right term, or a case construct. Then

$$\text{rcase}_\phi\ t_0\ y\ t_1\ t_2 = \text{case}\ t_0\ \text{of}\ y.t_1,y.t_2.$$

Case. Suppose $t_0 \equiv inl(t'_0)$. Then

$$\text{rcase}_\phi\ t_0\ y\ t_1\ t_2 = [t'_0/y]^{\phi_1} t_1$$

and

$$\begin{array}{rcl} \text{case}\ t_0\ \text{of}\ y.t_1,y.t_2 & \equiv & \text{case}\ inl(t'_0)\ \text{of}\ y.t_1,y.t_2 \\ & \rightsquigarrow & [t'_0/y]t_1. \end{array}$$

Now $\phi >_\Gamma \phi_1$ so by the outer-induction hypothesis $[t'_0/y]t_1 \rightsquigarrow^* [t'_0/y]^{\phi_1} t_1$. Therefore, (case t_0 of $y.t_1$,$y.t_2$) $\rightsquigarrow^*$ (rcase$_\phi$ $t_0\ y\ t_1\ t_2$).

Case. Suppose $t_0 \equiv inl(t'_0)$. Similar to the previous case.

Case. Suppose $t_0 \equiv$ case t'_0 of $z.t'_1$,$z.t'_2$. Then

$$\text{rcase}_\phi\ t_0\ y\ t_1\ t_2 = \text{case}\ t'_0\ \text{of}\ z.(\text{rcase}_\phi\ t'_1\ y\ t_1\ t_2),z.(\text{rcase}_\phi\ t'_2\ y\ t_1\ t_2)$$

and

$$\begin{array}{rcl} \text{case}\ t_0\ \text{of}\ y.t_1,y.t_2 & \equiv & \text{case}\ (\text{case}\ t'_0\ \text{of}\ z.t'_1,z.t'_2)\ \text{of}\ y.t_1,y.t_2 \\ & \rightsquigarrow & \text{case}\ t'_0\ \text{of}\ z.(\text{case}\ t'_1\ \text{of}\ y.t_1,y.t_2),z.(\text{case}\ t'_2\ \text{of}\ y.t_1,y.t_2). \end{array}$$

Trivially, $t_0 > t'_1$ and $t_0 > t'_2$ so by the inner-induction hypothesis (case t'_1 of $y.t_1$,$y.t_2$) $\rightsquigarrow^*$ (rcase$_\phi$ $t'_1\ y\ t_1\ t_2$) and (case t'_2 of $y.t_1$,$y.t_2$) $\rightsquigarrow^*$ (rcase$_\phi$ $t'_2\ y\ t_1\ t_2$). Therefore, (case t_0 of $y.t_1$,$y.t_2$) $\rightsquigarrow^*$ (rcase$_\phi$ $t_0\ y\ t_1\ t_2$).

Case. Suppose $t' \equiv t'_1\ t'_2$. By Lemma 5 there exists terms $\hat{t}'_1$ and $\hat{t}'_2$ such that $[t/x]^\phi t'_1 = \hat{t}'_1$ and $[t/x]^\phi t'_2 = \hat{t}'_2$. Since $t' > t'_1$ and $t' > t'_2$ we can apply the induction hypothesis to obtain $[t/x]t'_1 \rightsquigarrow^* \hat{t}'_1$ and $[t/x]t'_2 \rightsquigarrow^* \hat{t}'_2$. Now we case split on whether or not $\hat{t}'_1$ is a λ-abstraction and t'_1 is not, $\hat{t}'_1$ is a case construct and t'_1 is not, $ctype_\phi(x, t'_1)$ is undefined, or $\hat{t}'_1$ is neither a λ-abstraction or a case construct. If $ctype_\phi(x, t'_1)$ is undefined or $\hat{t}'_1$ is neither a λ-abstraction or a case construct then $[t/x]^\phi t' = ([t/x]^\phi t'_1)\ ([t/x]^\phi t'_2) \equiv \hat{t}'_1\ \hat{t}'_2$. Thus, $[t/x]t' \rightsquigarrow^* [t/x]^\phi t'$, because $[t/x]t' = ([t/x]t'_1)\ ([t/x]t'_2)$. So suppose $\hat{t}'_1 \equiv \lambda y : \phi'.\hat{t}''_1$ and t'_1 is not a λ-abstraction. By Lemma 4 there exists a type ψ such that $ctype_\phi(x, t'_1) = \psi$, $\psi \equiv \phi'' \to \phi'$, and ψ is a subexpression of ϕ, where by inversion on $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ there exists a type ϕ'' such that $\Gamma, x : \phi, \Gamma' \vdash t'_1 : \phi'' \to \phi'$. Then by the definiton of the hereditary substitution function $[t/x]^\phi(t'_1\ t'_2) = [\hat{t}'_2/y]^{\phi'} \hat{t}''_1$. Now we know $\phi >_{\Gamma,\Gamma'} \phi'$ so we can apply the induction hypothesis to obtain $[\hat{t}'_2/y]\hat{t}''_1 \rightsquigarrow^* [\hat{t}'_2/y]^{\phi'} \hat{t}''_1$. Now by knowing that $(\lambda y : \phi'.\hat{t}''_1)\ t'_2 \rightsquigarrow [\hat{t}'_2/y]\hat{t}''_1$ and by the previous fact we know $(\lambda y : \phi'.\hat{t}''_1)\ t'_2 \rightsquigarrow^* [\hat{t}'_2/y]^{\phi'} \hat{t}''_1$. We now make use of the well known result of full β-reduction. The result is stated as

$$\frac{\begin{array}{l} a \rightsquigarrow^* a' \\ b \rightsquigarrow^* b' \qquad a'\ b' \rightsquigarrow^* c \end{array}}{a\ b \rightsquigarrow^* c}$$

where a, a', b, b', and c are all terms. We apply this result by instantiating a, a', b, b', and c with $[t/x]t'_1$, $\hat{t}'_1$, $[t/x]t'_2$, $\hat{t}'_2$, and $[\hat{t}'_2/y]^{\phi'} \hat{t}''_1$ respectively. Therefore, $[t/x](t'_1\ t'_2) \rightsquigarrow^* [\hat{t}'_2/y]^{\phi'} \hat{t}''_1$.

Finally, suppose $\hat{t}'_1 \equiv$ case t_0 of $y.t_1$,$y.t_2$ and t'_0 is not a case construct. By Lemma 4 there exists a type ψ such that $ctype_\phi(x, t'_1) = \psi$, $\psi \equiv \phi'' \to \phi'$ and ψ is a subexpression of ϕ, where by inversion on the assumption $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ there exists a type ϕ'' such that $\Gamma, x : \phi, \Gamma' \vdash t'_1 : \phi'' \to \phi'$. Now

$$[t/x]^{\phi}(t'_1\ t'_2) = \text{case } t_0 \text{ of } y.(app_{\phi}\ t_1\ ([t/x]^{\phi}t'_2)), y.(app_{\phi}\ t_1\ ([t/x]^{\phi}t'_2))$$

and

$$[t/x](t'_1\ t'_2) = ([t/x]t'_1)([t/x]t'_2).$$

Clearly, $t' > t'_1$ and $t' > t'_2$, so by the induction hypothesis $[t/x]t'_1 \rightsquigarrow^* [t/x]^{\phi}t'_1$ and $[t/x]t'_2 \rightsquigarrow^* [t/x]^{\phi}t'_2$. Thus,

$$\begin{array}{rcl} ([t/x]t'_1)\ ([t/x]t'_2) & \rightsquigarrow^* & (\text{case } t_0 \text{ of } y.t_1, y.t_2)\ ([t/x]t'_2) \\ & \rightsquigarrow & \text{case } t_0 \text{ of } y.(app_{\phi}\ t_1([t/x]t'_2)), y.(app_{\phi}\ t_2([t/x]t'_2)) \end{array}$$

and

$$\begin{array}{c} ((\text{case } t_0 \text{ of } y.(app_{\phi}\ t_1\ ([t/x]t'_2)), y.(app_{\phi}\ t_1\ ([t/x]t'_2)))) \rightsquigarrow^* \\ (\text{case } t_0 \text{ of } y.(app_{\phi}\ t_1\ ([t/x]^{\phi}t'_2)), y.(app_{\phi}\ t_1\ ([t/x]^{\phi}t'_2))). \end{array}$$

It suffices to show that $(t_1\ ([t/x]t'_2)) \rightsquigarrow^* (app_{\phi}\ t_1\ ([t/x]t'_2))$ and $(t_2\ ([t/x]t'_2)) \rightsquigarrow^* (app_{\phi}\ t_2\ ([t/x]t'_2))$. This is a consequence of the following proposition:

Proposition. For all $\Gamma \vdash t_1 : \phi_1 \to \phi_2$ and $\Gamma \vdash t_2 : \phi_1$ we have $(t_1\ t_2) \rightsquigarrow^* (app_{\phi}\ t_1\ t_2)$.

We prove this by nested induction on the ordering (ϕ, t', t_1) and case split on the structure of t_1.

Case. Suppose t_1 is not a λ-abstraction or a case construct. Then $app_{\phi}\ t_1\ t_2 = t_1\ t_2$.

Case. Suppose $t_1 \equiv \lambda y : \phi_1.t''_1$. Then $app_{\phi}\ t_1\ t_2 = [t_2/y]^{\phi_1}t''_1$. Clearly, $\phi >_{\Gamma} \phi_1$ so by the outer-induction hypothesis $[t_2/y]t''_1 \rightsquigarrow^* [t_2/y]^{\phi_1}t''_1$. Therefore, $(t_1\ t_2) \rightsquigarrow^* (app_{\phi}\ t_1\ t_2)$.

Case. Suppose $t_1 \equiv \text{case } t'_0 \text{ of } y.t'_1, y.t'_2$. Then

$$app_{\phi}\ t_1\ t_2 = \text{case } t'_0 \text{ of } y.(app_{\phi}\ t'_1\ t_2), y.(app_{\phi}\ t'_2\ t_2)$$

and

$$\begin{array}{rcl} (t_1\ t_2) & = & (\text{case } t'_0 \text{ of } y.t'_1, y.t'_2)\ t_2 \\ & \rightsquigarrow & \text{case } t'_0 \text{ of } y.(t'_1\ t_2), y.(t'_2\ t_2). \end{array}$$

We can see that $t_1 > t'_1$ and $t_1 > t'_2$ so by the inner-induction hypothesis, $(t'_1\ t_2) \rightsquigarrow^* (app_{\phi}\ t'_1\ t_2)$ and $(t'_2\ t_2) \rightsquigarrow^* (app_{\phi}\ t'_2\ t_2)$. Therefore,

$$(\text{case } t'_0 \text{ of } y.(t'_1\ t_2), y.(t'_2\ t_2)) \rightsquigarrow^* (\text{case } t'_0 \text{ of } y.(app_{\phi}\ t'_1\ t_2), y.(app_{\phi}\ t'_2\ t_2)),$$

which implies $(t_1\ t_2) \rightsquigarrow^* (app_{\phi}\ t_1\ t_2)$.

Case. Suppose $t' \equiv t'_1[\phi'']$. Since $t' > t'_1$ we can apply the induction hypothesis to obtain $[t/x]t'_1 \rightsquigarrow^* [t'/x]^{\phi}t'_1$. We case split on whether or not $[t'/x]^{\phi}t'_1$ is a type abstraction and t'_1 is not. The case where it is not is trivial so we only consider the case where $[t'/x]^{\phi}t'_1 \equiv \Lambda X : *_l.s'$. Then $[t'/x]^{\phi}t' = [\phi'/X]s'$. Now we have $[t/x]t'_1 \rightsquigarrow^* [t'/x]^{\phi}t'_1$ and $[t/x](t'_1[\phi]) \equiv ([t/x]t'_1)[\phi] \rightsquigarrow^* ([t'/x]^{\phi}t'_1)[\phi] \rightsquigarrow [\phi/X]s'$. Thus, $[t/x]t' \rightsquigarrow^* [t'/x]^{\phi}t'$.

A.16 Proof of Lemma 12

This proof is by structural induction on n.

Case. n is a variable y. Clearly, $[\phi/X]n \equiv [\phi/X]y = y \in [\![\phi']\!]_{\Gamma, X:*_l, \Gamma'}$, and $(\Gamma, [\phi/X]\Gamma')(y) = [\phi/X]\phi'$. Also, we have $\Gamma, [\phi/X]\Gamma' \vdash [\phi/X]\phi' : *_p$ for some p, by Lemma 37. Hence, by the definition of the interpretation of types, $y \in [\![[\phi/X]\phi']\!]_{\Gamma, [\phi/X]\Gamma'}$.

Case. Let $n \equiv \lambda y : \psi.n'$. By the definition of the interpretation of types $\phi' \equiv \psi \to \psi'$. By the induction hypothesis $[\phi/X]n' \in [\![[\phi/X]\psi']\!]_{\Gamma, \Gamma', y:[\phi/X]\psi}$. Again by the definition of the interpretation of types $\lambda y : [\phi/X]\psi.[\phi/X]n' \equiv [\phi/X](\lambda y : \psi.n') \in [\![[\phi/X]\phi']\!]_{\Gamma, [\phi/X]\Gamma'}$.

Case. Let $n \equiv n_1 n_2$. By the definition of the interpretation of types $\phi' \equiv \psi$, $n_1 \in [\![\psi' \to \psi]\!]_{\Gamma, X:*_q, \Gamma'}$, and $n_2 \in [\![\psi']\!]_{\Gamma, X:*_q, \Gamma'}$. By the induction hypothesis $[\phi/X]n_1 \in [\![[\phi/X](\psi' \to \psi)]\!]_{\Gamma, [\phi/X]\Gamma'}$ and $[\phi/X]n_2 \in [\![[\phi/X]\psi']\!]_{\Gamma, [\phi/X]\Gamma'}$. Now by the definition of the interpretation of types $([\phi/X]n_1)([\phi/X]n_2) \in [\![[\phi/X]\psi]\!]_{\Gamma, [\phi/X]\Gamma'}$, since $[\phi/X]n_1$, cannot be a λ-abstraction.

Case. Let $n \equiv \Lambda Y : *_q.n'$. By the definition of the interpretation of types $\phi' = \forall Y : *_q.\psi$ and $n' \in [\![\psi]\!]_{\Gamma,X:*_l,\Gamma',Y:*_q}$. By the induction hypothesis $[\phi/X]n' \in [\![[\phi/X]\psi]\!]_{\Gamma,[\phi/X]\Gamma',Y:*_q}$ and by the definition of the interpretation of types $\Lambda Y : *_q.[\phi/X]n' \in [\![\forall Y : *_q.[\phi/X]\psi]\!]_{\Gamma,[\phi/X]\Gamma'}$ which is equivalent to $[\phi/X](\Lambda Y : *_q.n') \in [\![[\phi/X](\forall Y : *_q.\psi)]\!]_{\Gamma,[\phi/X]\Gamma'}$.

Case. Let $n \equiv n'[\psi]$. By the definition of the interpretation of types $\phi' = [\psi/Y]\psi'$, for some Y, ψ, and there exists a q such that $\Gamma, X : *_l, \Gamma' \vdash \psi : *_q$, and $n' \in [\![\forall Y : *_q.\psi']\!]_{\Gamma,X:*_l,\Gamma'}$. By the induction hypothesis $[\phi/X]n' \in [\![[\phi/X](\forall Y : *_q.\psi')]\!]_{\Gamma,[\phi/X]\Gamma'}$. Therefore, by the definition of the interpretation of types $([\phi/X]n')[\psi] \in [\![[\psi/Y]([\phi/X]\psi')]\!]_{\Gamma,[\phi/X]\Gamma'}$, which is equivalent to $[\phi/X](n'[\psi]) \in [\![[\phi/X]([\psi/Y]\psi')]\!]_{\Gamma,[\phi/X]\Gamma'}$.

Case. Let $n \equiv inl(n')$. By the definition of the interpretation of types, $\phi' \equiv \phi'_1 + \phi'_2$, for some types ϕ'_1 and ϕ'_2, and $n' \in [\![\phi'_1]\!]_{\Gamma,X:*_l,\Gamma'}$. By the induction hypothesis, $[\phi/X]n' \in [\![[\phi/X]\phi'_1]\!]_{\Gamma,[\phi/X]\Gamma'}$. Thus, by the definition of the interpretation of types, $inl([\phi/X]n') \equiv [\phi/X]inl(n') \in [\![[\phi/X]\phi']\!]_{\Gamma,[\phi/X]\Gamma'}$.

Case. Let $n \equiv inr(n')$. Similar to the inject-left case above.

Case. Let $n \equiv$ case n_0 of $y.n_1$,$y.n_2$. By the definition of the interpretation of types, $n_0 \in [\![\phi'_1 + \phi'_2]\!]_{\Gamma,X:\phi,\Gamma'}$, for some types ϕ'_1 and ϕ'_2, $n_1 \in [\![\phi']\!]_{\Gamma,X:\phi,\Gamma',y:\phi'_1}$, and $n_2 \in [\![\phi']\!]_{\Gamma,X:\phi,\Gamma',y:\phi'_2}$. By the induction hypothesis, $[\phi/X]n_0 \in [\![[\phi/X](\phi'_1+\phi'_2)]\!]_{\Gamma,[\phi/X]\Gamma'}$, $[\phi/X]n_1 \in [\![[\phi/X]\phi']\!]_{\Gamma,[\phi/X]\Gamma',y:[\phi/X]\phi'_1}$, and $[\phi/X]n_2 \in [\![[\phi/X]\phi']\!]_{\Gamma,[\phi/X]\Gamma',y:[\phi/X]\phi'_2}$. Finally, by the definition of the interpretation of types, $case\ [\phi/X]n_0\ of\ y.[\phi/X]n_1, y.[\phi/X]n_2 \equiv$ $[\phi/X]$(case n_0 of $y.n_1$,$y.n_2$) $\in [\![[\phi/X]\phi']\!]_{\Gamma,[\phi/X]\Gamma'}$.

A.17 Proof of Type Soundness

This is a proof by induction on the structure of the typing derivation of t.

Case.

$$\frac{\Gamma(x) = \phi \qquad \Gamma\ Ok}{\Gamma \vdash x : \phi}$$

By regularity $\Gamma \vdash \phi : *_l$ for some l, hence $[\![\phi]\!]_\Gamma$ is nonempty. Clearly, $x \in [\![\phi]\!]_\Gamma$ by the definition of the interpretation of types.

Case.

$$\frac{\Gamma, x : \phi_1 \vdash t : \phi_2}{\Gamma \vdash \lambda x : \phi_1.t : \phi_1 \to \phi_2}$$

By the induction hypothesis $t \in [\![\phi_2]\!]_{\Gamma,x:\phi_1}$ and by the definition of the interpretation of types $t \leadsto^! n \in [\![\phi_2]\!]_{\Gamma,x:\phi_1}$ and $\Gamma, x : \phi_1 \vdash n : \phi_2$. Thus, by applying the λ-abstraction type-checking rule, $\Gamma \vdash \lambda x : \phi_1.n : \Pi x : \phi_1.\phi_2$ so by the definition of the interpretation of types $\lambda x : \phi_1.n \in [\![\phi_1 \to \phi_2]\!]_\Gamma$. Thus, according to the definition of the interpretation of types $\lambda x : \phi_1.t \leadsto^! \lambda x : \phi_1.n \in [\![\phi_1 \to \phi_2]\!]_\Gamma$.

Case.

$$\frac{\Gamma \vdash t_1 : \phi_2 \to \phi_1 \qquad \Gamma \vdash t_2 : \phi_2}{\Gamma \vdash t_1\ t_2 : \phi_1}$$

By the induction hypothesis $t_1 \leadsto^! n_1 \in [\![\phi_2 \to \phi_1]\!]_\Gamma$, $t_2 \leadsto^! n_2 \in [\![\phi_2]\!]_\Gamma$, $\Gamma \vdash \phi_2 \to \phi_1 : *_p$, and $\Gamma \vdash \phi_2 : *_q$. Inversion on the arrow-type kind-checking rule yields, $\Gamma \vdash \phi_1 : *_r$, and by Lemma **??**, $\Gamma, x : \phi_2, \Gamma' \vdash \phi_1 : *_r$.

Now we know from above that $n_1 \in [\![\phi_2 \to \phi_1]\!]_\Gamma$ and $n_2 \in [\![\phi_2]\!]_\Gamma$, hence $\Gamma \vdash n_1 : \phi_2 \to \phi_1$ and $\Gamma \vdash n_2 : \phi_2$. It suffices to show that $n_1\ n_2 \in [\![\phi_2]\!]_\Gamma$. Clearly, $n_1\ n_2 = [n_1/z](z\ n_2)$ for some variable $z \notin FV(n_1, n_2)$. Lemma 5, Lemma 10, and Lemma 9 allow us to conclude that $[n_1/z](z\ n_2) \rightsquigarrow^* [n_1/z]^{\phi_2 \to \phi_1}(z\ n_2)$, $\Gamma \vdash [n_1/z]^{\phi_2\to\phi_1}(z\ n_2) : \phi_2$, and $[n_1/z]^{\phi_2\to\phi_1}(z\ n_2)$ is normal. Thus, $t_1\ t_2 \rightsquigarrow^* n_1\ n_2 = [n_1/z](z\ n_2) \rightsquigarrow^! [n_1/z]^{\phi_2\to\phi_1}(z\ n_2) \in [\![\phi_2]\!]_\Gamma$.

Case.

$$\frac{\Gamma, X : *_p \vdash t : \phi}{\Gamma \vdash \Lambda X : *_p.t : \forall X : *_p.\phi}$$

By the induction hypothesis and definition of the interpretation of types $t \in [\![\phi]\!]_{\Gamma,X:*_p}$, $t \rightsquigarrow^! n \in [\![\phi]\!]_{\Gamma,X:*_p}$ and $\Lambda X : *_p.n \in [\![\phi]\!]_\Gamma$. Again, by definition of the interpretation of types $\Lambda X : *_p.t \rightsquigarrow^! \Lambda X : *_p.n \in [\![\phi]\!]_\Gamma$.

Case.

$$\frac{\Gamma \vdash t : \forall X : *_l.\phi_1 \qquad \Gamma \vdash \phi_2 : *_l}{\Gamma \vdash t[\phi_2] : [\phi_2/X]\phi_1}$$

By the induction hypothesis $t \in [\![\forall X : *_l.\phi_1]\!]_\Gamma$ and by the definition of the interpretation of types we know $t \rightsquigarrow^! n \in [\![\forall X : *_l.\phi_1]\!]_\Gamma$. We case split on whether or not n is a type abstraction. If not then again, by the definition of the interpretation of types $n[\phi_2] \in [\![[\phi_2/X]\phi_1]\!]_\Gamma$, therefore $t \in [\![[\phi_2/X]\phi_1]\!]_\Gamma$. Suppose $n \equiv \Lambda X : *_l.n'$. Then $t[\phi_2] \rightsquigarrow^* (\Lambda X : *_l.n')[\phi_2] \rightsquigarrow [\phi_2/X]n'$. By the definition of the interpretation of types $n' \in [\![\phi_1]\!]_{\Gamma,X:*_l}$. Therefore, by Lemma 12 $[\phi_2/X]n' \in [\![[\phi_2/X]\phi_1]\!]_\Gamma$.

Case.

$$\frac{\Gamma \vdash t : \phi_1 \qquad \Gamma \vdash \phi_2 : *_p}{\Gamma \vdash inl(t) : \phi_1 + \phi_2}$$

By the induction hypothesis, $t \in [\![\phi_1]\!]_\Gamma$ and by the definition of the interpretation of types, $t \rightsquigarrow^! n \in [\![\phi_1]\!]_\Gamma$, and $inl(n) \in [\![\phi_1 + \phi_2]\!]_\Gamma$. Again, by the definition of the interpretation of types $inl(t) \rightsquigarrow^! inl(n) \in [\![\phi_1 + \phi_2]\!]_\Gamma$.

Case.

$$\frac{\Gamma \vdash t : \phi_2 \qquad \Gamma \vdash \phi_1 : *_p}{\Gamma \vdash inr(t) : \phi_1 + \phi_2}$$

Similar the inject-left case above.

Case.

$$\frac{\Gamma \vdash t_0 : \phi_1 + \phi_2 \qquad \Gamma, x : \phi_1 \vdash t_1 : \psi \qquad \Gamma, x : \phi_2 \vdash t_2 : \psi}{\Gamma \vdash \text{case } t_0 \text{ of } x.t_1,x.t_2 : \psi}$$

By the induction hypothesis and the definition of the interpretation of types $t_0 \rightsquigarrow^! n_0 \in [\![\phi_1 + \phi_2]\!]_\Gamma$ and $\Gamma \vdash n_0 : \phi_1 + \phi_2$, $t_1 \rightsquigarrow^! n_1 \in [\![\psi]\!]_{\Gamma,x:\phi_1}$ and $\Gamma, x : \phi_1 \vdash n_1 : \psi$, and $t_2 \rightsquigarrow^! n_2 \in [\![\psi]\!]_{\Gamma,x:\phi_2}$ and $\Gamma, x : \phi_2 \vdash n_2 : \psi$. Clearly,

$$\begin{array}{rcl} \text{case } t_0 \text{ of } x.t_1,x.t_2 & \rightsquigarrow^* & \text{case } n_0 \text{ of } x.n_1,x.n_2 \\ & = & [n_0/z](\text{case } z \text{ of } x.n_1,x.n_2), \end{array}$$

for some variable $z \notin FV(n_0, n_1, n_2) \cup \{x\}$. Lemma 5, Lemma 10, and Lemma 9 allow us to conclude that $[n_0/z](\text{case } z \text{ of } x.n_1,x.n_2) \rightsquigarrow^* [n_0/z]^{\phi_1+\phi_2}(\text{case } z \text{ of } x.n_1,x.n_2)$, $\Gamma \vdash [n_0/z]^{\phi_1+\phi_2}(\text{case } z \text{ of } x.n_1,x.n_2) : \psi$, and $[n_0/z]^{\phi_1+\phi_2}(\text{case } z \text{ of } x.n_1,x.n_2)$ is normal. Thus, $[n_0/z]^{\phi_1+\phi_2}(\text{case } z \text{ of } x.n_1,x.n_2) \in [\![\psi]\!]_\Gamma$ and we obtain case t_0 of $x.t_1,x.t_2 \in [\![\psi]\!]_\Gamma$.

B Proofs of Results Pertaining to DSSF$^{=}$

B.1 Miscellaneous Definitions and Results

This section contains definitions and lemmas that were omitted from the main part of the paper due to space constraints. First are a few basic syntactic lemmas.

▶ **Lemma 41** (Type Equality Context Conversion). *If* $\Gamma, x : [\phi_1/X]\phi, \Gamma' \vdash t : \phi'$ *and* $\Gamma \vdash p : \phi_1 = \phi_2$ *then* $\Gamma, x : [\phi_2/X]\phi, \Gamma' \vdash t : \phi'$.

Proof. This hold by straightforward induction on the assumed typing derivation. ◀

▶ **Lemma 42** (Syntactic Inversion).

i. *If* $\Gamma \vdash \lambda x : \phi_1.t : \phi_1 \to \phi_2$ *then* $\Gamma, x : \phi_1 \vdash t : \phi_2$.
ii. *If* $\Gamma \vdash t_1\ t_2 : \phi_2$ *then there exists a type* ϕ_1, *such that,* $\Gamma \vdash t_1 : \phi_1 \to \phi_2$ *and* $\Gamma \vdash t_2 : \phi_1$.

The depth function, defined in the following definition, is used in the proof of Lemma 44 and is used to show that our ordering on types is well founded.

▶ **Definition 43.** The depth of a type ϕ is defined as follows:

$$\begin{array}{lcl}
depth(t) & = & 0\text{, where } t \text{ is any term.}\\
depth(X) & = & 1\\
depth(\Pi x : \phi.\phi') & = & depth(\phi) + depth(\phi')\\
depth(\forall X : *_l.\phi) & = & depth(\phi) + 1
\end{array}$$

We use the metric (l, d) in lexicographic combination, where l is the level of a type ϕ, and d is the depth of ϕ in the proof of the next lemma.

▶ **Lemma 44** (Well-Founded Measure). *If* $\phi >_\Gamma \phi'$ *then* $(l, d) > (l', d')$, *where* $\Gamma \vdash \phi : *_l$, $depth(\phi) = d$, $\Gamma \vdash \phi : *_{l'}$, *and* $depth(\phi') = d'$.

The next few lemmas are results about the kinding and typing relations as well as well-formed contexts. The first lemma states that every kindable type is kindable within a well-formed environment.

▶ **Lemma 45.** *If* $\Gamma \vdash \phi : *_p$ *then* $\Gamma\ Ok$.

The Type Substitution for the Interpretation of Types lemma is needed in the proof of the Type Substitution for Kinding, Typing, Context-Ok lemma, which is needed in the proof of the main substitution lemma.

▶ **Lemma 46** (Type Substitution for Kinding, Typing, and Context-Ok). *Suppose* $\Gamma \vdash \phi' : *_p$. *Then*

i. *if* $\Gamma, X : *_p, \Gamma' \vdash \phi : *_q$ *with a derivation of depth d, then* $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]\phi : *_q$ *with a derivation of depth d,*
ii. *if* $\Gamma, X : *_l, \Gamma' \vdash t : \phi$ *with a derivation of depth d, then* $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t : [\phi'/X]\phi$ *with a derivation of depth d, and*
iii. *if* $\Gamma, X : *_p, \Gamma'\ Ok$ *with a derivation of depth d, then* $\Gamma, [\phi'/X]\Gamma'\ Ok$ *with a derivation of depth d.*

Next we show that the kinding and typing relations are closed under term substitutions.

▶ **Lemma 47** (Term Substitution for Kinding, Typing, and Context-Ok). *Suppose* $\Gamma \vdash t' : \phi'$. *Then* *i.* *if* $\Gamma, x : \phi', \Gamma' \vdash \phi : *_l$ *with a derivation of depth d, then* $\Gamma, [t'/x]\Gamma' \vdash [t'/x]\phi : *_l$ *with a derivation of depth d,*

ii. *if* $\Gamma, x : \phi', \Gamma' \vdash t : \phi$ *with a derivation of depth d, then* $\Gamma, [t'/x]\Gamma' \vdash [t'/x]t : [t'/x]\phi$ *with a derivation of depth d, and*

iii. *if* $\Gamma, x : \phi', \Gamma'$ *Ok with a derivation of depth d, then* $\Gamma, [t'/x]\Gamma'$ *Ok with a derivation of depth d.*

Context weakening for the kinding and typing relations is defined by Lemma 48, and is used by the proof of the main substitution lemma.

▶ **Lemma 48** (Context Weakening for Kinding and Typing). *Assume* $\Gamma, \Gamma'', \Gamma'$ *Ok,* $\Gamma, \Gamma' \vdash \phi : *_p$ *and* $\Gamma, \Gamma' \vdash t : \phi$*. Then i.* $\Gamma, \Gamma'', \Gamma' \vdash \phi : *_p$ *and ii.* $\Gamma, \Gamma'', \Gamma' \vdash t : \phi$*.*

Finally, we have regularity which is used by the proof of type soundness. These are all the results about the kinding relation and well-formed contexts that we will need to conclude normalization, but we need syntactic inversion for the typing relation, because it is not a trivial result like it is for SSF$^+$.

▶ **Lemma 49** (Regularity). *If* $\Gamma \vdash t : \phi$ *then* $\Gamma \vdash \phi : *_p$ *for some p.*

▶ **Lemma 50** (Transitivity of Type Equality). *If* $\Gamma \vdash \phi_1 \approx \phi_2$ *and* $\Gamma \vdash \phi_2 \approx \phi_3$ *then* $\Gamma \vdash \phi_1 \approx \phi_3$*.*

▶ **Lemma 51** (Symmetry of Type Equality). *If* $\Gamma \vdash \phi \approx \phi'$ *then* $\Gamma \vdash \phi' \approx \phi$*.*

▶ **Lemma 52** (Substitution for Type Equality). *If* $\Gamma, x : \phi, \Gamma' \vdash \phi' \approx \phi''$ *and* $\Gamma \vdash t : \phi$ *then* $\Gamma, [t/x]\Gamma' \vdash [t/x]\phi' \approx [t/x]\phi''$*.*

▶ **Lemma 53.** *If* $\Gamma \vdash \phi \approx \Pi j : \phi_1.\phi_2$ *then there exists a term h and types* ϕ'_1 *and* ϕ'_2 *such that* $\phi \equiv \Pi h : \phi'_1.\phi'_2$*.*

▶ **Definition 54.** The following function constructs the set of redexes within a term:

$$
\begin{array}{lll}
\multicolumn{3}{l}{rset(x) = \emptyset} \\
\multicolumn{3}{l}{rset(join) = \emptyset} \\
\multicolumn{3}{l}{rset(\lambda x : \phi.t) = rset(t)} \\
\multicolumn{3}{l}{rset(\Lambda X : *_l.t) = rset(t)} \\
\multicolumn{3}{l}{rset(t_1\ t_2)} \\
= & rset(t_1, t_2) & \text{if } t_1 \text{ is not a } \lambda\text{-abstraction.} \\
= & \{t_1\ t_2\} \cup rset(t'_1, t_2) & \text{if } t_1 \equiv \lambda x : \phi.t'_1. \\
\multicolumn{3}{l}{rset(t''[\phi''])} \\
= & rset(t'') & \text{if } t'' \text{ is not a type absraction.} \\
= & \{t''[\phi'']\} \cup rset(t''') & \text{if } t'' \equiv \Lambda X : *_l.t'''.
\end{array}
$$

The extention of $rset$ to multiple arguments is defined as follows:

$$rset(t_1, \ldots, t_n) =^{def} rset(t_1) \cup \cdots \cup rset(t_n).$$

B.2 Properties of The Hereditary Substitution Function

We prove all the same proprites of the heredtiary substitution function as we did for the previous language. They only differ in the statement of some of the lemmas due to syntactic inversion. We simply list all the properties below. The defintion of $rset$ can be found in Appendix B.1 of this report. [8].

▶ **Lemma 55** (Properties of $ctype_\phi$).

i. *If* $ctype_\phi(x,t) = \phi'$ *then* $head(t) = x$ *and* ϕ' *is a subexpression of* ϕ.

ii. *If* $\Gamma, x : \phi, \Gamma' \vdash t : \phi'$ *and* $ctype_\phi(x,t) = \phi''$ *then* $\Gamma, x : \phi, \Gamma' \vdash \phi' \approx \phi''$.

iii. *If* $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$ *and* $head(t_1\ t_2) = x$ *then* $ctype_\phi(x,t_1) = \Pi j : \phi_1.\phi_2$ *for some term* j *and types* ϕ_1 *and* ϕ_2.

iv. *If* $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$, $\Gamma \vdash t : \phi$, $[t/x]^\phi t_1 \equiv \lambda y : \phi_1.q$, *and* t_1 *is not then there exists a type* ψ *such that* $ctype_\phi(x,t_1) = \psi$.

▶ **Lemma 56** (Total, Type Preserving, and Sound with Respect to Reduction)**.** *Suppose* $\Gamma \vdash t : \phi$ *and* $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$. *Then*

i. *there exists a term* t'' *and a type* ϕ'' *such that* $[t/x]^\phi t' = t''$, $\Gamma, [t/x]\Gamma' \vdash t'' : \phi''$, *and* $\Gamma, \Gamma' \vdash \phi'' \approx [t/x]\phi'$, *and*

ii. $[t/x]t' \rightsquigarrow^* [t/x]^\phi t'$.

▶ **Corollary 57.** *Suppose* $\Gamma \vdash t : \phi$ *and* $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$. *Then* $\Gamma, [t/x]\Gamma' \vdash [t/x]^\phi t' : [t/x]\phi'$.

▶ **Lemma 58** (Redex Preserving)**.** *If* $\Gamma \vdash t : \phi$, $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ *then* $|rset(t', t)| \geq |rset([t/x]^\phi t')|$.

▶ **Lemma 59** (Normality Preserving)**.** *If* $\Gamma \vdash n : \phi$ *and* $\Gamma, x : \phi' \vdash n' : \phi'$ *then there exists a normal term* n'' *such that* $[n/x]^\phi n' = n''$.

B.3 Proof of Lemma 45

This is a proof by structural induction on the kinding derivation of $\Gamma \vdash \phi : *_p$.

Case.

$$\frac{\Gamma(X) = *_p \qquad p \leq q \qquad \Gamma\ Ok}{\Gamma \vdash X : *_q}$$

By inversion of the kind-checking rule $\Gamma\ Ok$.

Case.

$$\frac{\Gamma \vdash \phi_1 : *_p \qquad \Gamma, x : \phi_1 \vdash \phi_2 : *_q}{\Gamma \vdash \Pi x : \phi_1.\phi_2 : *_{max(p,q)}}$$

By the induction hypothesis, $\Gamma \vdash \phi_1 : *_p$ implies $\Gamma\ Ok$, thus, after applying the rule $\Gamma\ Ok$.

Case.

$$\frac{\Gamma, X : *_q \vdash \phi : *_p}{\Gamma \vdash \forall X : *_q.\phi : *_{max(p,q)+1}}$$

By the induction hypothesis $\Gamma, X : *_p\ Ok$, and by inversion of the type-variable well-formed contexts rule $\Gamma\ Ok$.

Case.

$$\frac{\Gamma \vdash \phi : *_p \qquad \Gamma \vdash t_1 : \phi \qquad \Gamma \vdash t_2 : \phi}{\Gamma \vdash t_1 = t_2 : *_p}$$

By the induction hypothesis, $\Gamma \vdash \phi : *_p$ implies $\Gamma\ Ok$. Now since the above rule does not alter the context in anyway Γ remains Ok after applying the above rule.

B.4 Proof of Type Substitution for Kinding, Typing, and Context-Ok

This is a proof by induction on d. We prove the first part of the lemma, and then the second, and finally the third, doing a case analysis for each implication on the form of the derivation whose depth is being considered.

Case.

$$\frac{(\Gamma, X : *_p, \Gamma')(Y) = *_r \qquad r \leq s \qquad \Gamma, X : *_p, \Gamma'\ Ok}{\Gamma, X : *_p, \Gamma' \vdash Y : *_s}$$

By assumption $\Gamma \vdash \phi' : *_p$. We must consider whether or not $X \equiv Y$. If $X \equiv Y$ then $[\phi'/X]Y \equiv \phi'$, $r = p$, and $q = s$; this conclusion is equivalent to $\Gamma, [\phi'/X]\Gamma' \vdash \phi' : *_q$ and by the third part of the induction hypothesis
$\Gamma, [\phi'/X]\Gamma'\ Ok$. If $X \not\equiv Y$ then $[\phi'/X]Y \equiv Y$ and by the third part of the induction hypothesis $\Gamma, [\phi'/X]\Gamma'\ Ok$, hence, $\Gamma, [\phi'/X]\Gamma' \vdash Y : *_q$.

Case.

$$\frac{\Gamma, X : *_p, \Gamma' \vdash \phi_1 : *_r \qquad \Gamma, X : *_p, \Gamma', x : \phi_1 \vdash \phi_2 : *_s}{\Gamma, X : *_p, \Gamma' \vdash \Pi x : \phi_1.\phi_2 : *_{max(r,s)}}$$

Here $q = max(r, s)$ and by the first part of the induction hypothesis $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]\phi_1 : *_r$ and $\Gamma, [\phi'/X]\Gamma', x : [\phi'/X]\phi_1 \vdash [\phi'/X]\phi_2 : *_s$. We can now reapply the rule to get $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X](\Pi x : \phi_1.\phi_2) : *_q$.

Case.

$$\frac{\Gamma, X : *_q, \Gamma', Y : *_r \vdash \phi : *_s}{\Gamma, X : *_p, \Gamma' \vdash \forall Y : *_r.\phi : *_{max(r,s)+1}}$$

Here $q = max(r, s) + 1$ and by the first part of the induction hypothesis
$\Gamma, [\phi'/X]\Gamma', Y : *_r \vdash [\phi'/X]\phi : *_s$. We can reapply this rule to get $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]\forall Y : *_r.\phi : *_q$.

Case.

$$\frac{\Gamma, X : *_p, \Gamma' \vdash \phi : *_p \qquad \Gamma, X : *_p, \Gamma' \vdash t_1 : \phi \qquad \Gamma, X : *_p, \Gamma' \vdash t_2 : \phi}{\Gamma, X : *_p, \Gamma' \vdash t_1 = t_2 : *_p}$$

By the first part of the induction hypothesis, $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]\phi : *_p$ and by the second part, $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t_1 : [\phi'/X]\phi$ and $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t_2 : [\phi'/X]\phi$. Finally, by reapplying the equality type kind-checking rule we obtain, $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t_1 = [\phi'/X]t_2 : *_p$, which is equivalent to, $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X](t_1 = t_2) : *_p$.

We now show the second part of the lemma. The proof proceeds in the same way as the proof of part one.

Case. Suppose t is a variable.

$$\frac{(\Gamma, X : *_l, \Gamma')(y) = \phi \qquad \Gamma, X : *_l, \Gamma'\ Ok}{\Gamma, X : *_l, \Gamma' \vdash y : \phi}$$

Clearly, $[\phi'/X]y \rightsquigarrow y$ and $(\Gamma, [\phi'/X]\Gamma')(y) = [\phi'/X]\phi$, because the assumed context is well-formed. By the third part of the induction hypothesis,
$\Gamma, [\phi'/X]\Gamma'\ Ok$. So by applying the variable type-checking rule, $\Gamma, [\phi'/X]\Gamma' \vdash y : [\phi'/X]\phi$.

Case.

$$\frac{\Gamma, X : *_l, \Gamma', y : \phi_1 \vdash t : \phi_2}{\Gamma, X : *_l, \Gamma' \vdash \lambda y : \phi_1.t : \Pi y : \phi_1.\phi_2}$$

By the second part of the induction hypothesis, $\Gamma, [\phi'/X]\Gamma', y : [\phi'/X]\phi_1 \vdash [\phi'/X]t' : [\phi'/X]\phi_2$. So by applying the lambda type-checking rule, $\Gamma, [\phi'/X]\Gamma' \vdash \lambda y : [\phi'/X]\phi_1.[\phi'/X]t' : \Pi y : [\phi'/X]\phi_1.[\phi'/X]\phi_2$, which is equivalent to
$\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X](\lambda y : \phi_1.t') : [\phi'/X](\Pi y : \phi_1.\phi_2)$.

Case.

$$\frac{\Gamma, X : *_l, \Gamma', Y : *_p \vdash t : \phi}{\Gamma, X : *_l, \Gamma' \vdash \Lambda Y : *_p.t : \forall Y : *_p.\phi}$$

This case is similar to the previous case.

Case.

$$\frac{t_1 \downarrow t_2 \qquad \Gamma, X : *_l, \Gamma' \; Ok}{\Gamma, X : *_l, \Gamma' \vdash join : t_1 = t_2}$$

It is a property of full β-reduction that if $t_1 \downarrow t_2$ then $[\phi'/X]t_1 \downarrow [\phi'/X]t_2$. By the third part of the induction hypothesis, $\Gamma, [\phi'/X]\Gamma' \; Ok$. Now by applying the join type-checking rule, $\Gamma, [\phi'/X]\Gamma' \vdash join : [\phi'/X]t_1 = [\phi'/X]t_2$.

Case.

$$\frac{\Gamma, X : *_l, \Gamma' \vdash t_0 : t_1 = t_2 \qquad \Gamma, X : *_l, \Gamma' \vdash t : [t_1/y]\phi}{\Gamma, X : *_l, \Gamma' \vdash t : [t_2/y]\phi}$$

By the second part of the induction hypothesis, $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t_0 : [\phi'/X]t_1 = [\phi'/X]t_2$ and $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t : [\phi'/X][t_1/y]\phi$. The latter is equivalent to $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t : [[\phi'/X]t_1/y][\phi'/X]\phi$. Now by applying the conversion type-checking rule, $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t : [[\phi'/X]t_2/y][\phi'/X]\phi$.

Case.

$$\frac{\Gamma, X : *_l, \Gamma' \vdash t_1 : \Pi y : \phi_1.\phi_2 \qquad \Gamma, X : *_l, \Gamma' \vdash t_2 : \phi_1}{\Gamma, X : *_l, \Gamma' \vdash t_1\ t_2 : [t_2/y]\phi_2}$$

By the second part of the induction hypothesis, $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t_1 : \Pi y : [\phi'/X]\phi_1.[\phi'/X]\phi_2$ and $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t_2 : [\phi'/X]\phi_1$. By applying the application type-checking rule, $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t_1\ [\phi'/X]t_2 : [[\phi'/X]t_2/y]\phi_2$, which is equivalent to $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X](t_1\ t_2) : [\phi'/X][t_2/y]\phi_2$.

Case.

$$\frac{\Gamma, X : *_l, \Gamma' \vdash t : \forall Y : *_p.\phi_1 \qquad \Gamma, X : *_l, \Gamma' \vdash \phi_2 : *_p}{\Gamma, X : *_l, \Gamma' \vdash t[\phi_2] : [\phi_2/Y]\phi_1}$$

By the second part of the induction hypothesis, $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t : \forall Y : *_p.[\phi'/X]\phi_1$ and by by the first part, $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]\phi_2 : *_p$. Finally by applying the instantiation type-checking rule, $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X]t[[\phi'/X]\phi_2] : [[\phi'/X]\phi_2/Y][\phi'/X]\phi_1$, which is equivalent to $\Gamma, [\phi'/X]\Gamma' \vdash [\phi'/X](t[\phi_2]) : [\phi'/X][\phi_2/Y]\phi_1$.

We now show the third part of the lemma. The case were $d = 0$ cannot arise, since it requires the context to be empty. Suppose $d = n + 1$. We do a case analysis on the last rule applied in the derivation of $\Gamma, X : *_p, \Gamma'$.

Case. Suppose $\Gamma' = \Gamma'', Y : *_q$.

$$\frac{\Gamma, X : *_p, \Gamma''\ Ok}{\Gamma, X : *_p, \Gamma'', Y : *_q\ Ok}$$

By the third part of the induction hypothesis, $\Gamma, [\phi'/X]\Gamma''\ Ok$. Now, by reapplying the rule above $\Gamma, [\phi'/X]\Gamma'', Y : *_q\ Ok$, hence $\Gamma, [\phi'/X]\Gamma'\ Ok$, since $X \not\equiv Y$.

Case. Suppose $\Gamma' = \Gamma'', y : \phi$.

$$\frac{\Gamma, X : *_p, \Gamma'' \vdash \phi : *_q \qquad \Gamma, X : *_p, \Gamma''\ Ok}{\Gamma, X : *_p, \Gamma'', y : \phi\ Ok}$$

By the first part of the induction hypothesis, $\Gamma', [\phi'/X]\Gamma'' \vdash [\phi'/X]\phi : *_q$ and $\Gamma', [\phi'/X]\Gamma''\ Ok$. Thus, by reapplying the rule above $\Gamma, [\phi'/X]\Gamma'', x : [\phi'/X]\phi\ Ok$, therefore, $\Gamma, [\phi'/X]\Gamma'\ Ok$.

B.5 Proof of Term Substitution for Kinding, Typing, and Context-Ok

This is a proof by induction on d. We prove the i. first, and then the second and finally the third, doing a case analysis for each implication on the form of the derivation whose depth is being considered.

Case.

$$\frac{(\Gamma, X : *_p, \Gamma')(Y) = *_r \qquad r \leq s \qquad \Gamma, x : \phi', \Gamma'\ Ok}{\Gamma, x : \phi', \Gamma' \vdash Y : *_s}$$

Clearly, $[t'/x]Y \rightsquigarrow Y$, so $(\Gamma, [t'/x]\Gamma')(Y) = *_r$. By the second part of the induction hypothesis, $\Gamma, [t'/x]\Gamma'\ Ok$. By applying the variable kind-checking rule, $\Gamma, [t'/x]\Gamma' \vdash Y : *_s$.

Case.

$$\frac{\Gamma, x : \phi', \Gamma' \vdash \phi_1 : *_r \qquad \Gamma, x : \phi', \Gamma', y : \phi_1 \vdash \phi_2 : *_s}{\Gamma, x : \phi', \Gamma' \vdash \Pi y : \phi_1.\phi_2 : *_{max(r,s)}}$$

By the first part of the induction hypothesis, $\Gamma, [t'/x]\Gamma' \vdash [t'/x]\phi_1 : *_r$ and $\Gamma, [t'/x]\Gamma', y : [t'/x]\phi_1 \vdash [t'/x]\phi_2 : *_s$. By applying the Π-type type-checking rule, $\Gamma, [t'/x]\Gamma' \vdash \Pi y : [t'/x]\phi_1.[t'/x]\phi_2 : *_{max(r,s)}$, which is equivalent to $\Gamma, [t'/x]\Gamma' \vdash [t'/x](\Pi y : \phi_1.\phi_2) : *_{max(r,s)}$.

Case.

$$\frac{\Gamma, x : \phi', \Gamma', Y : *_r \vdash \phi : *_s}{\Gamma, x : \phi', \Gamma' \vdash \forall Y : *_r.\phi : *_{max(r,s)+1}}$$

By the first part of the induction hypothesis, $\Gamma, [t'/x]\Gamma', Y : *_r \vdash [t'/x]\phi : *_s$ and by applying the forall-type type-checking rule, $\Gamma, [t'/x]\Gamma' \vdash \forall Y : *_r.[t'/x]\phi : *_{max(r,s)+1}$. The latter is equivalent to
$\Gamma, [t'/x]\Gamma' \vdash \forall[t'/x](Y : *_r.\phi) : *_{max(r,s)+1}$.

Case.

$$\frac{\Gamma, x : \phi', \Gamma' \vdash \phi : *_p \qquad \Gamma, x : \phi', \Gamma' \vdash t_1 : \phi \qquad \Gamma, x : \phi', \Gamma' \vdash t_2 : \phi}{\Gamma, x : \phi', \Gamma' \vdash t_1 = t_2 : *_p}$$

By third part of the induction hypothesis, $\Gamma, [t'/x]\Gamma' \vdash [t'/x]t_1 : [t'/x]\phi$ and $\Gamma, [t'/x]\Gamma' \vdash [t'/x]t_2 : [t'/x]\phi$. By the first part of the induction hypothesis,$\Gamma, \Gamma' \vdash t_1 = t_2 : *_p$. Finally, by reapplying the type equality kind-checking rule,

We now prove the second part of the lemma. This part proceeds in the exact same way as the first.

Case. Suppose t is a variable.

$$\frac{(\Gamma, x : \phi', \Gamma')(y) = \phi \qquad \Gamma, x : \phi', \Gamma'\ Ok}{\Gamma, x : \phi', \Gamma' \vdash y : \phi}$$

Case. Let $y \equiv x$. Then $[t'/x]t \leadsto t'$. Thus, $\Gamma \vdash [t'/x]t : [t'/x]\phi$.

Case. Let y be distinct from x. By the third part of the induction hypothesis $\Gamma, [t'/x]\Gamma'\ Ok$ and since y is distinct from x, $(\Gamma, [t'/x]\Gamma')(y) = [t'/x]\phi$. Now after applying the variable type-checking rule we obtain $\Gamma', [t'/x]\Gamma' \vdash y : [t'/x]\phi$.

Case.

$$\frac{\Gamma, x : \phi', \Gamma', y : \phi_1 \vdash t : \phi_2}{\Gamma, x : \phi', \Gamma' \vdash \lambda y : \phi_1.t : \Pi y : \phi_1.\phi_2}$$

By the second part of the induction hypothesis, $\Gamma, [t'/x]\Gamma', y : [t'/x]\phi_1 \vdash [t'/x]t : [t'/x]\phi_2$. Now by reapplying the lambda type-checking rules we obtain, $\Gamma, [t'/x]\Gamma' \vdash \lambda y : [t'/x]\phi_1.t : \Pi y : [t'/x]\phi_1.[t'/x]\phi_2$, which is equivalent to $\Gamma, [t'/x]\Gamma' \vdash [t'/x](\lambda y : \phi_1.t) : [t'/x](\Pi y : \phi_1.\phi_2)$.

Case.

$$\frac{\Gamma, x : \phi', \Gamma', X : *_p \vdash t : \phi}{\Gamma, x : \phi', \Gamma' \vdash \Lambda X : *_p.t : \forall X : *_p.\phi}$$

This case is similar to the previous case.

Case.

$$\frac{t_1 \downarrow t_2 \qquad \Gamma, x : \phi', \Gamma'\ Ok}{\Gamma, x : \phi', \Gamma' \vdash join : t_1 = t_2}$$

It is a well known property of full β-reduction that if $t_1 \downarrow t_2$ then $[t'/x]t_1 \downarrow [t'/x]t_2$. By the third part of the induction hypothesis we know $\Gamma, [t'x/]\Gamma'\ Ok$. Now by applying the join type-checking rule we obtain $\Gamma, [t'/x]\Gamma' \vdash join : [t'x/]t_1 \downarrow [t'/x]t_2$.

Case.

$$\frac{\Gamma, x : \phi', \Gamma' \vdash t_0 : t_1 = t_2 \qquad \Gamma, x : \phi', \Gamma' \vdash t : [t_1/y]\phi}{\Gamma, x : \phi', \Gamma' \vdash t : [t_2/y]\phi}$$

By the second part of the induction hypothesis we know $\Gamma, [t'/x]\Gamma' \vdash [t'/x]t_0 : [t'/x]t_1 = [t'/x]t_2$ and $\Gamma, [t'/x]\Gamma' \vdash [t'/x]t : [t'/x][[t'/x]t_1/y]\phi$. Clearly,
$\Gamma, [t'/x]\Gamma' \vdash [t'/x]t : [t'/x][[t'/x]t_1/y]\phi$ is equivalent to $\Gamma, [t'/x]\Gamma' \vdash [t'/x]t : [[t'/x]t_1/y][t'/x]\phi$.
Now by applying the conv type-checking rule we obtain
$\Gamma, [t'/x]\Gamma' \vdash [t'/x]t : [[t'/x]t_2/y][t'/x]\phi$.

Case.

$$\frac{\Gamma, x : \phi', \Gamma' \vdash t_1 : \Pi y : \phi_1.\phi_2 \qquad \Gamma, x : \phi', \Gamma' \vdash t_2 : \phi_1}{\Gamma, x : \phi', \Gamma' \vdash t_1\ t_2 : [t_2/y]\phi_2}$$

By the second part of the induction hypothesis, $\Gamma, [t'/x]\Gamma' \vdash [t'/x]t_1 : \Pi y : [t'/x]\phi_1.[t'/x]\phi_2$ and $\Gamma, [t'/x]\Gamma' \vdash [t'/x]t_2 : [t'/x]\phi_1$. Now by applying the application type-checking rule we obtain,
$\Gamma, [t'/x]\Gamma' \vdash [t'/x](t_1\ t_2) : [[t'/x]t_2/y][t'/x]\phi_2$.

Case.

$$\frac{\Gamma, x : \phi', \Gamma' \vdash t : \forall X : *_l.\phi_1 \qquad \Gamma, x : \phi', \Gamma' \vdash \phi_2 : *_l}{\Gamma, x : \phi', \Gamma' \vdash t[\phi_2] : [\phi_2/X]\phi_1}$$

By the second part of the induction hypothesis, $\Gamma, [t'/x]\Gamma' \vdash [t'/x]t : \forall X : *_l.[t'/x]\phi_1$. By first part of the induction hypothesis, $\Gamma, [t'/x]\Gamma' \vdash [t'/x]\phi_2 : *_l$. Now by applying the type instantiation type-checking rule we obtain $\Gamma, [t'/x]\Gamma' \vdash [t'/x](t[\phi_2]) : [[t'/x]\phi_2/X][t'/x]\phi_1$.

We now show the third implication. The case were $d = 0$ cannot arise, since it requires the context to be empty. Suppose $d = n + 1$. We do a case analysis on the last rule applied in the derivation of $\Gamma, x : \phi', \Gamma'$.

Case. Suppose $\Gamma' = \Gamma'', Y : *_q$.

$$\frac{\Gamma, x : \phi', \Gamma''\ Ok}{\Gamma, x : \phi', \Gamma'', Y : *_q\ Ok}$$

By the third part of the induction hypothesis, $\Gamma, [t'/x]\Gamma''\ Ok$. Now, by reapplying the rule above $\Gamma, [t'/x]\Gamma'', Y : *_q\ Ok$, hence $\Gamma, [t'/x]\Gamma'\ Ok$.

Case. Suppose $\Gamma' = \Gamma'', y : \phi$.

$$\frac{\Gamma, x : \phi', \Gamma'' \vdash \phi : *_q \qquad \Gamma, x : \phi', \Gamma''\ Ok}{\Gamma, x : \phi', \Gamma'', y : \phi\ Ok}$$

By the first part of the induction hypothesis, $\Gamma', [t'/x]\Gamma'' \vdash [t'/x]\phi : *_q$ and $\Gamma', [t'/x]\Gamma''\ Ok$. Thus, by reapplying the rule above $\Gamma, [t'/x]\Gamma'', y : [t'/x]\phi\ Ok$, therefore, $\Gamma, [t'/x]\Gamma'\ Ok$.

B.6 Proof of Context Weakening for Kinding and Typing

We first prove part i. This is a proof by structural induction on the kinding derivation of $\Gamma, \Gamma' \vdash \phi : *_p$.

Case.

$$\frac{(\Gamma, \Gamma')(X) = *_p \qquad p \leq q \qquad \Gamma, \Gamma'\ Ok}{\Gamma, \Gamma' \vdash X : *_q}$$

If $(\Gamma, \Gamma')(X) = *_p$ then $(\Gamma, \Gamma'', \Gamma')(X) = *_p$, hence, by reapplying the type-variable kind-checking rule, $\Gamma, \Gamma'', \Gamma' \vdash \phi : *_p$.

Case.

$$\frac{\Gamma, \Gamma' \vdash \phi_1 : *_p \qquad \Gamma, \Gamma', x : \phi_1 \vdash \phi_2 : *_q}{\Gamma, \Gamma' \vdash \Pi x : \phi_1.\phi_2 : *_{max(p,q)}}$$

By the induction hypothesis $\Gamma, \Gamma'', \Gamma' \vdash \phi_1 : *_p$ and $\Gamma, \Gamma'', \Gamma', x : \phi_1 \vdash \phi_2 : *_q$, hence, by reapplying the Π-type kind-checking rule $\Gamma, \Gamma'', \Gamma'' \vdash \Pi x : \phi_1.\phi_2 : *_{max(p,q)}$.

Case.

$$\frac{\Gamma, \Gamma' \vdash \phi : *_p \qquad \Gamma, \Gamma' \vdash t_1 : \phi \qquad \Gamma, \Gamma' \vdash t_2 : \phi}{\Gamma, \Gamma' \vdash t_1 = t_2 : *_p}$$

By the induction hypothesis we know, $\Gamma, \Gamma'', \Gamma' \vdash \phi : *_p$. By assumption we know $\Gamma, \Gamma'', \Gamma'\ Ok$, $\Gamma, \Gamma' \vdash t_1 : \phi$, and $\Gamma, \Gamma' \vdash t_2 : \phi$, thus, by the second part of the induction hypothesis, $\Gamma, \Gamma'', \Gamma' \vdash t_1 : \phi$ and $\Gamma, \Gamma'', \Gamma' \vdash t_2 : \phi$. Now by applying the equality type kind-checking rule we obtain, $\Gamma, \Gamma'', \Gamma' \vdash t_1 = t_2 : *_p$.

Case.

$$\frac{\Gamma, \Gamma', X : *_q \vdash \phi' : *_p}{\Gamma, \Gamma' \vdash \forall X : *_q.\phi' : *_{max(p,q)+1}}$$

By the induction hypothesis $\Gamma, \Gamma'', \Gamma', X : *_p \vdash \phi : *_q$, hence, by reapplying the forall-type kind-checking rule $\Gamma, \Gamma'', \Gamma' \vdash \forall X : *_p.\phi : *_{max(p,q)+1}$.

We now prove part two.

This is a proof by induction on the form of the typing derivation of $\Gamma, \Gamma' \vdash t : \phi$.

Case.

$$\frac{(\Gamma, \Gamma')(x) = \phi \qquad \Gamma, \Gamma'\ Ok}{\Gamma, \Gamma' \vdash x : \phi}$$

Clearly, if $(\Gamma, \Gamma')(x) = \phi$ then $(\Gamma, \Gamma'', \Gamma')(x) = \phi$. By assumption, $\Gamma, \Gamma'', \Gamma'\ Ok$. Finally, by reapplying the variable type-checking rule, $\Gamma, \Gamma'', \Gamma' \vdash x : \phi$.

Case.

$$\frac{\Gamma, \Gamma', x : \phi_1 \vdash t : \phi_2}{\Gamma, \Gamma' \vdash \lambda x : \phi_1.t : \Pi x : \phi_1.\phi_2}$$

By the first induction hypothesis, $\Gamma, \Gamma'', \Gamma', x : \phi_1 \vdash t : \phi_2$ and by reapplying the lambda type-checking rule, $\Gamma, \Gamma'', \Gamma' \vdash \lambda x : \phi_1.t : \Pi x : \phi_1.\phi_2$.

Case.

$$\frac{\Gamma, \Gamma' \vdash t_1 : \Pi x : \phi_1.\phi_2 \qquad \Gamma, \Gamma' \vdash t_2 : \phi_1}{\Gamma, \Gamma' \vdash t_1\ t_2 : [t_2/x]\phi_2}$$

By the first induction hypothesis, $\Gamma, \Gamma'', \Gamma' \vdash t_1 : \Pi x : \phi_1.\phi_2$ and $\Gamma, \Gamma'', \Gamma' \vdash t_2 : \phi_1$. Now by reapplying the application type-checking rule, $\Gamma, \Gamma'', \Gamma' \vdash t_1\ t_2 : [t_2/x]\phi_2$.

Case.

$$\frac{\Gamma, \Gamma', X : *_p \vdash t : \phi}{\Gamma, \Gamma' \vdash \Lambda X : *_p.t : \forall X : *_p.\phi}$$

By the first induction hypothesis, $\Gamma, \Gamma'', \Gamma', X : *_p \vdash t : \phi$ and by reapplying the type abstraction type-checking rule, $\Gamma, \Gamma'', \Gamma' \vdash \Lambda X : *_p.t : \forall X : *_p.\phi$.

Case.

$$\frac{\Gamma, \Gamma' \vdash t : \forall X : *_l.\phi_1 \qquad \Gamma, \Gamma' \vdash \phi_2 : *_l}{\Gamma, \Gamma' \vdash t[\phi_2] : [\phi_2/X]\phi_1}$$

By the first induction hypothesis we know $\Gamma, \Gamma'', \Gamma' \vdash t : \forall X : *_l.\phi_1$ and by the second $\Gamma, \Gamma'', \Gamma' \vdash \phi_2 : *_l$. By applying the type instantiation type-checking rule, $\Gamma, \Gamma'', \Gamma' \vdash t[\phi_2] : [\phi_2/X]\phi_1$.

Case.

$$\frac{t_1 \downarrow t_2 \qquad \Gamma, \Gamma' \vdash t_1 : \phi \qquad \Gamma, \Gamma' \vdash t_2 : \phi \qquad \Gamma, \Gamma'\ Ok}{\Gamma, \Gamma' \vdash join : t_1 = t_2}$$

By the induction hypothesis, $\Gamma, \Gamma'', \Gamma' \vdash t_1 : \phi$ and $\Gamma, \Gamma'', \Gamma' \vdash t_2 : \phi$. We know by assumption that $\Gamma, \Gamma'', \Gamma'\ Ok$. By reapplying the join type type-checking rule, $\Gamma, \Gamma'', \Gamma' \vdash join : t_1 = t_2$.

Case.

$$\frac{\Gamma, \Gamma' \vdash t_0 : t_1 = t_2 \qquad \Gamma, \Gamma' \vdash t : [t_1/x]\phi}{\Gamma, \Gamma' \vdash t : [t_2/x]\phi}$$

By the induction hypothesis, $\Gamma, \Gamma'', \Gamma' \vdash t_0 : t_1 = t_2$ and $\Gamma, \Gamma'', \Gamma' \vdash t : [t_1/x]\phi$. By reapplying the conv type-checking rules, $\Gamma, \Gamma'', \Gamma' \vdash t : [t_2/x]\phi$.

B.7 Proof of Regularity

This proof is by structural induction on the derivation of $\Gamma \vdash t : \phi$.

Case.

$$\frac{\Gamma(x) = \phi \qquad \Gamma\ Ok}{\Gamma \vdash x : \phi}$$

By the definition of well-formedness contexts $\Gamma \vdash \phi : *_p$ for some p.

Case.

$$\frac{\Gamma, x : \phi_1 \vdash t : \phi_2}{\Gamma \vdash \lambda x : \phi_1.t : \Pi x : \phi_1.\phi_2}$$

By the induction hypothesis $\Gamma \vdash \phi_1 : *_p$, $\Gamma, x : \phi_1 \vdash \phi_2 : *_q$ and by applying the Π-type kind-checking rule we get $\Gamma \vdash \Pi x : \phi_1.\phi_2 : *_{max(p,q)}$.

Case.

$$\frac{t_1 \downarrow t_2 \qquad \Gamma \vdash t_1 : \phi \qquad \Gamma \vdash t_2 : \phi \qquad \Gamma\ Ok}{\Gamma \vdash join : t_1 = t_2}$$

By the induction hypothesis we know $\Gamma \vdash \phi : *_l$. Thus, by applying the equality type kind-checking rule, $\Gamma \vdash t_1 = t_2 : *_l$.

Case.

$$\frac{\Gamma \vdash t_0 : t_1 = t_2 \qquad \Gamma \vdash t : [t_1/x]\phi}{\Gamma \vdash t : [t_2/x]\phi}$$

By the induction hypothesis, $\Gamma \vdash t_1 = t_2 : *_p$ and $\Gamma \vdash [t_1/x]\phi : *_q$ which implies $\Gamma, x : \phi'' \vdash \phi' : *_q$. By inversion of the equality type kind-checking rule, $\Gamma \vdash t_1 : \phi''$ and $\Gamma \vdash t_2 : \phi''$. Thus, by Lemma 47, $\Gamma \vdash [t_2/x]\phi : *_q$.

Case.

$$\frac{\Gamma \vdash t_1 : \phi_1 \to \phi_2 \qquad \Gamma \vdash t_2 : \phi_1}{\Gamma \vdash t_1\ t_2 : \phi_2}$$

By the induction hypothesis $\Gamma \vdash \phi_1 \to \phi_2 : *r$ and $\Gamma \vdash \phi_1 : *_p$. By inversion of the arrow-type kind-checking rule $r = max(p, q)$, for some q, which implies $\Gamma \vdash \phi_2 : *_q$.

Case.

$$\frac{\Gamma, X : *_p \vdash t : \phi}{\Gamma \vdash \Lambda X : *_p.t : \forall X : *_q.\phi}$$

By the induction hypothesis $\Gamma, X : *_q \vdash \phi : *_p$. By applying the forall-type kind-checking rule $\Gamma \vdash \forall X.\phi : *_{max(p,q)+1}$.

Case.

$$\frac{\Gamma \vdash t : \forall X : *_p.\phi_1 \qquad \Gamma \vdash \phi_2 : *_p}{\Gamma \vdash t[\phi_2] : [\phi_2/X]\phi_1}$$

By assumption $\Gamma \vdash \phi_2 : *_r$. By the induction hypothesis $\Gamma \vdash \forall X : *_p.\phi_1 : *_s$ and by inversion of the forall-type kind-checking rule $r = max(p, q) + 1$, for some q, which implies $\Gamma, X : *_p \vdash \phi_1 : *_q$. Now, by Lemma 46, $\Gamma \vdash [\phi_2/X]\phi_1 : *_q$.

B.8 Proof of Transitivity of Type Equality

This is a proof by induction on the form of first assumed type equality with an inner induction on the form of the second.

Case.

$$\frac{\Gamma \vdash p : t_1 = t_2}{\Gamma \vdash [t_1/x]\phi \approx [t_2/x]\phi}\ \mathrm{TE}_{\mathrm{Q}_1}$$

We case split on the form of the final rule applied in the derivation of $\Gamma \vdash \phi_2 \approx \phi_3$.

Case.

$$\frac{\Gamma \vdash p' : t'_1 = t'_2}{\Gamma \vdash [t'_1/z]\phi \approx [t'_2/z]\phi}\ \mathrm{TE}_{\mathrm{Q}_1}$$

Trivial, because we are using the same type in all equalities.

Case.

$$\frac{\Gamma \vdash [t_2/x]\phi \approx [t_2/x]\phi_3 \qquad \Gamma \vdash p : t_2 = t_3}{\Gamma \vdash [t_2/x]\phi \approx [t_3/x]\phi_3}\ \mathrm{TE}_{\mathrm{Q}_2}$$

By the induction hypothesis we know $\Gamma \vdash [t_1/x]\phi \approx [t_2/x]\phi_3$ and by applying TEQ$_2$ using $\Gamma \vdash p : t_2 = t_3$ we obtain $\Gamma \vdash [t_1/x]\phi \approx [t_3/x]\phi_3$.

Case.

$$\frac{\Gamma \vdash [t_1/x]\phi_1 \approx [t_1/x]\phi_2 \qquad \Gamma \vdash p : t_1 = t_2}{\Gamma \vdash [t_1/x]\phi_1 \approx [t_2/x]\phi_2}\ \mathrm{TE}_{\mathrm{Q}_2}$$

Case.

$$\frac{\Gamma \vdash p' : t'_1 = t'_2}{\Gamma \vdash [t'_1/z]\phi_2 \approx [t'_2/z]\phi_2}\ \mathrm{TE}_{\mathrm{Q}_1}$$

Trivial.

Case.

$$\frac{\Gamma \vdash [t_2/x]\phi_2 \approx [t_2/x]\phi_3 \qquad \Gamma \vdash p : t_2 = t_3}{\Gamma \vdash [t_2/x]\phi_2 \approx [t_3/x]\phi_3}\ \mathrm{TE}_{\mathrm{Q}_2}$$

By the induction hypothesis we know $\Gamma \vdash [t_1/x]\phi_1 \approx [t_2/x]\phi_3$ and by applying TEq$_2$ using $\Gamma \vdash p : t_2 = t_3$ we obtain $\Gamma \vdash [t_1/x]\phi_1 \approx [t_3/x]\phi_3$.

B.9 Proof of Symmetry of Type Equality

This is a proof by induction on the assumed type equality.

Case.

$$\frac{\Gamma \vdash p : t_1 = t_2}{\Gamma \vdash [t_1/x]\phi \approx [t_2/x]\phi}\ \mathrm{TE}_{\mathrm{Q}_1}$$

It suffices to show that there exists a p' such that $\Gamma \vdash p' : t_2 = t_1$. We obtain this by the following typing derivation:

$$\frac{\Gamma \vdash p : t_1 = t_2 \qquad \Gamma \vdash \mathit{join} : [t_1/y](y = t_1)}{\Gamma \vdash \mathit{join} : t_2 = t_1}\ \textsc{Conv}$$

Therefore, by applying TE$_{\mathrm{Q}_1}$ using the previous proof we obtain $\Gamma \vdash [t_2/x]\phi \approx [t_1/x]\phi$.

Case.

$$\frac{\Gamma \vdash [t_1/x]\phi \approx [t_1/x]\phi' \qquad \Gamma \vdash p : t_1 = t_2}{\Gamma \vdash [t_1/x]\phi \approx [t_2/x]\phi'}\ \textsc{TEq}_2$$

By the induction hypothesis $\Gamma \vdash [t_1/x]\phi' \approx [t_1/x]\phi$. Clearly, if $\Gamma \vdash [t_1/x]\phi' \approx [t_1/x]\phi$ then $\Gamma, x : \phi'' \vdash \phi' \approx \phi$ for some type ϕ'' such that $\Gamma \vdash t_1 : \phi$ and $\Gamma \vdash t_2 : \phi$. We know such a type exists by inversion on $\Gamma \vdash p : t_1 = t_2$. Now by the substitution for type equality lemma using $\Gamma, x : \phi'' \vdash \phi' \approx \phi$ and $\Gamma \vdash t_2 : \phi$ we know $\Gamma \vdash [t_2/x]\phi' \approx [t_2/x]\phi$. We obtain our desired result by the following derivation:

$$\frac{\Gamma \vdash [t_2/x]\phi' \approx [t_2/x]\phi \qquad \dfrac{\Gamma \vdash p : t_1 = t_2 \qquad \Gamma \vdash \mathit{join} : [t_1/y](y = t_1)}{\Gamma \vdash \mathit{join} : t_2 = t_1}\ \textsc{Conv}}{\Gamma \vdash [t_2/x]\phi' \approx [t_1/x]\phi}\ \textsc{TEq}_2$$

B.10 Proof of Substitution for Type Equality

This is a proof by induction on the form of the assumed type-equality judgement.

Case.

$$\frac{\Gamma, x : \phi, \Gamma' \vdash p : t_1 = t_2}{\Gamma, x : \phi, \Gamma' \vdash [t_1/x]\phi \approx [t_2/x]\phi}\ \textsc{TEq}_1$$

By Lemma **??**, $\Gamma, [n/x]\Gamma' \vdash [n/x]p : [n/x](t_1 = t_2)$. Now by applying TEq$_1$, $\Gamma, [n/x]\Gamma' \vdash [[n/x]t_1/x][n/x]\phi \approx [[n/x]t_2/x][n/x]\phi$.

Case.

$$\frac{\Gamma, x : \phi, \Gamma' \vdash [t_1/y]\phi \approx [t_1/y]\phi' \qquad \Gamma, x : \phi, \Gamma' \vdash p : t_1 = t_2}{\Gamma, x : \phi, \Gamma' \vdash [t_1/y]\phi \approx [t_2/y]\phi'}\ \textsc{TEq}_2$$

By the induction hypothesis, $\Gamma, [n/x]\Gamma' \vdash [[n/x]t_1/y][n/x]\phi \approx [[n/x]t_1/y][n/x]\phi'$. By Lemma **??**, $\Gamma, [n/x]\Gamma' \vdash [n/x]p : [n/x]t_1 = [n/x]t_2$. Therefore by applying TEq$_2$, $\Gamma, [n/x]\Gamma' \vdash [n/x]t_1/y][n/x]\phi \approx [[n/x]t_2/y][n/x]\phi'$.

B.11 Proof of Type Syntactic Conversion

If $\Gamma \vdash t : \phi$ and $\Gamma \vdash \phi \approx \phi'$ then we know several things: $\phi \equiv [\bar{t}/\bar{x}]\phi''$, $\phi' \equiv [\bar{t'}/\bar{x'}]\phi''$, $\Gamma \vdash \bar{p} : \bar{t} = \bar{t'}$, and $\Gamma \vdash t : [\bar{t}/\bar{x}]\phi''$ for some type ϕ''. Suppose each vector has i elements. Then by applying the conversion type-checking rule i times with the appropriate proof from our vector of proofs we will obtain $\Gamma \vdash t : [\bar{t'}/\bar{x}]\phi''$. This last result is exactly, $\Gamma \vdash t : \phi'$.

B.12 Proof of Lemma 53

This is a proof by induction on the form of the assume type-equality derivation.

Case.

$$\frac{\Gamma \vdash p : t_1 = t_2}{\Gamma \vdash [t_1/x](\Pi j : \phi'_1.\phi'_2) \approx [t_2/x](\Pi j : \phi'_1.\phi'_2)}\ \textsc{TEq}_1$$

Trivial, because ϕ must also be a Π-type.

Case.

$$\frac{\Gamma \vdash [t_1/x]\phi' \approx [t_1/x](\Pi j : \phi'_1.\phi'_2) \qquad \Gamma \vdash p : t_1 = t_2}{\Gamma \vdash [t_1/x]\phi' \approx [t_2/x](\Pi j : \phi'_1.\phi'_2)}\ \text{TE}_{\text{Q}_2}$$

By the induction hypothesis $\phi \equiv [t_1/x]\phi' \equiv \Pi h : \psi_1.\psi_2$ for some term h and types ψ_1 and ψ_2.

B.13 Proof of Injectivity of Π-Types for Type Equality

This is a proof by induction on the form of the assumed typing derivation.

Case.

$$\frac{\Gamma \vdash p : t_1 = t_2}{\Gamma \vdash [t_1/x]\phi' \approx [t_2/x]\phi'}\ \text{TE}_{\text{Q}_1}$$

Trivial, becasue ϕ' and ϕ'' only differ by terms, which do not affect the ordering of types.

Case.

$$\frac{\Gamma \vdash [t_1/x]\phi' \approx [t_1/x]\phi'' \qquad \Gamma \vdash p : t_1 = t_2}{\Gamma \vdash [t_1/x]\phi' \approx [t_2/x]\phi''}\ \text{TE}_{\text{Q}_2}$$

By the induction hypothesis $\phi >_\Gamma [t_1/x]\phi''$, which implies that $\phi >_\Gamma [t_2/x]\phi''$.

B.14 Proof of Lemma 22

The possible form for ψ is only a Π-type. So it suffices to show that if $\Gamma \vdash \Pi y : \phi'_1.\phi'_2 \approx \Pi y : \phi_1.\phi_2$ and ψ is a subexpression of ϕ'' then $\phi'' >_\Gamma \phi_1$ and $\phi'' >_{\Gamma,y:\phi_1} \phi_2$.

It must be the case that $\phi'' >_\Gamma \phi'_1$ and $\phi'' >_{\Gamma,y:\phi_1} \phi'_2$, because ϕ'_1 and ϕ'_2 are both subexpressions of ϕ''. By injectivity of Π-types for typed equality we obtain $\Gamma \vdash \phi'_1 \approx \phi_1$ and $\Gamma, y : \phi_1 \vdash \phi'_2 \approx \phi_2$. Finally, by Lemma 21 we know $\phi'' >_\Gamma \phi_1$ and $\phi'' >_{\Gamma,y:\phi_1} \phi_2$.

B.15 Proof of Syntactic Inversion

We prove all cases by induction on the form of the typing relation.

Case. Part i.

Case.

$$\frac{\Gamma, x : \phi_1 \vdash t : \phi_2}{\Gamma \vdash \lambda x : \phi_1.t : \Pi x : \phi_1.\phi_2}$$

Trivial.

Case.

$$\frac{\Gamma \vdash p : t_1 = t_2 \qquad \Gamma \vdash \lambda x : \phi_1.t : [t_1/y]\phi'}{\Gamma \vdash \lambda x : \phi_1.t : [t_2/y]\phi'}$$

Here $\phi \equiv [t_2/y]\phi'$. By the induction hypothesis, where ϕ is $[t_1/y]\phi'$, there exists a type ϕ_2, such that $\Gamma, x : \phi_1 \vdash t : \phi_2$ and $\Gamma \vdash \Pi x : \phi_1.\phi_2 \approx [t_1/y]\phi'$, which implies that $\Gamma \vdash [t'_1/y](\Pi x : \phi'_1.\phi'_2) \approx [t_1/y]\phi'$ and $\Gamma \vdash p' : t'_1 = t_1$ for some terms t'_1 and p'. Hence, by TEq$_2$ $\Gamma \vdash [t_1/y](\Pi x : \phi'_1.\phi'_2) \approx [t_1/y]\phi'$ and by applying the same rule a second time, except using the proof $\Gamma \vdash p : t_1 = t_2$ we obtain $\Gamma \vdash [t_1/y](\Pi x : \phi'_1.\phi'_2) \approx [t_2/y]\phi'$. Finally, using TEq$_2$ a third time using $\Gamma \vdash p' : t'_1 = t_1$ we obtain $\Gamma \vdash [t'_1/y](\Pi x : \phi'_1.\phi'_2) \approx [t_2/y]\phi'$, which is equivalent to $\Gamma \vdash \Pi x : \phi_1.\phi_2 \approx [t_2/y]\phi'$.

Case. Part ii.

Case.

$$\frac{\Gamma \vdash t_1 : \Pi x : \phi_1.\phi_2 \qquad \Gamma \vdash t_2 : \phi_1}{\Gamma \vdash t_1\ t_2 : [t_2/x]\phi_2}$$

Trivial.

Case.

$$\frac{\Gamma \vdash p : t_1 = t_2 \qquad \Gamma \vdash t_1\ t_2 : [t_1/y]\phi}{\Gamma \vdash t_1\ t_2 : [t_2/y]\phi}$$

Similar to the previous case.

Case. Part iii.

Case.

$$\frac{\Gamma, X : *_l \vdash t : \phi}{\Gamma \vdash \Lambda X : *_l.t : \forall X : *_l.\phi}$$

Trivial.

Case.

$$\frac{\Gamma \vdash p : t_1 = t_2 \qquad \Gamma \vdash \Lambda X : *_l.t : [t_1/y]\phi''}{\Gamma \vdash \Lambda X : *_l.t : [t_2/y]\phi''}$$

Similar to the previous case.

Case. Part iv.

Case.

$$\frac{\Gamma \vdash t : \forall X : *_l.\phi_1 \qquad \Gamma \vdash \phi_2 : *_l}{\Gamma \vdash t[\phi_2] : [\phi_2/X]\phi_1}$$

Trivial.

Case.

$$\frac{\Gamma \vdash p : t_1 = t_2 \qquad \Gamma \vdash t[\phi_2] : [t_1/y]\phi'}{\Gamma \vdash t[\phi_2] : [t_2/y]\phi'}$$

Similar to the previous case.

Case. Part v.

Case.

$$\frac{t_1 \downarrow t_2 \qquad \Gamma \vdash t_1 : \phi \qquad \Gamma \vdash t_2 : \phi \qquad \Gamma\ Ok}{\Gamma \vdash join : t_1 = t_2}$$

Trivial.

Case.

$$\frac{\Gamma \vdash p : t'_1 = t'_2 \qquad \Gamma \vdash join : [t'_1/y]\phi')}{\Gamma \vdash join : [t'_2/y]\phi'}$$

Similar to the previous case.

B.16 Proof of Properties of $ctype_\phi$

We prove part one first. This is a proof by induction on the structure of t.

Case. Suppose $t \equiv x$. Then $ctype_\phi(x, x) = \phi$. Clearly, $head(x) = x$ and ϕ is a subexpression of itself.

Case. Suppose $t \equiv t_1\ t_2$. Then $ctype_\phi(x, t_1\ t_2) = \phi''$ when $ctype_\phi(x, t_1) = \Pi y : \phi'.\phi''$. Now $t > t_1$ so by the inducitOn hypothesis $head(t_1) = x$ and $\Pi y : \phi'.\phi''$ is a subexpression of ϕ. Therefore, $head(t_1\ t_2) = x$ and certainly ϕ'' is a subexpression of ϕ.

We now prove part two. This is also a proof by induction on the structure of t.

Case. Suppose $t \equiv x$. Then $ctype_\phi(x, x) = \phi$. Clearly, $\Gamma, x : \phi, \Gamma' \vdash \phi \approx \phi$.

Case. Suppose $t \equiv t_1\ t_2$. Then $ctype_\phi(x, t_1\ t_2) = [t_2/y]\phi_2$ when $ctype_\phi(x, t_1) = \Pi y : \phi_1.\phi_2$. By inversion on the assumption $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ there exists a term z and types ψ_1 and ψ_2 such that $\Gamma, x : \phi, \Gamma' \vdash t_1 : \Pi z : \psi_1.\psi_2$ and $\Gamma, x : \phi, \Gamma' \vdash \phi' \approx [t_2/z]\psi_2$. Now $t > t_1$ so by the induction hypothesis $\Gamma, x : \phi, \Gamma' \vdash \Pi z : \psi_1.\phi_2 \approx \Pi y : \phi_1.\phi_2$. Injectivity of equalities between Π-types yeilds $\Gamma, x : \phi, \Gamma' \vdash \psi_1 \approx \phi_1$, and $\Gamma, x : \phi, \Gamma', z : \psi_1 \vdash \psi_2 \approx [z/y]\phi_2$ and by substitution for types equality $\Gamma, x : \phi, \Gamma', y : \phi_1 \vdash [t_2/y]([y/z]\psi_2) \approx [t_2/y]\phi_2$. Now $\Gamma, x : \phi, \Gamma' \vdash \phi' \approx [t_2/z]\psi_2$ is equivalent to $\Gamma, x : \phi, \Gamma' \vdash \phi' \approx [t_2/y]([y/z]\psi_2)$. Therefore, by transitivity of types equality $\Gamma, x : \phi, \Gamma' \vdash \phi' \approx [t_2/y]\phi_2$.

Next we prove part three. This is a proof by induction on the structure of t_1.

Case. Suppose $t_1 \equiv x$. Then $ctype_\phi(x, x) = \phi$ and by inversion on the assumed typed derivation $\Gamma, x : \phi, \Gamma' \vdash x : \Pi j : \phi_1.\phi_2$ for some term j and types ϕ_1 and ϕ_2 which implies that $\phi \equiv \Pi j : \phi_1.\phi_2$.

Case. Suppose $t_1 \equiv t'_1\ t'_2$. Again by inversion on the assumed typing derivation there exists a term j and types ϕ_1 and ϕ_2 such that $\Gamma, x : \phi, \Gamma' \vdash t'_1\ t'_2 : \Pi j : \phi_1.\phi_2$. Now $ctype_\phi(x, t_1) = ctype_\phi(x, t'_1\ t'_2) = [t'_2/j]\phi_2$ when $ctype_\phi(x, t'_1) = \Pi h : \phi'_1.\phi'_2$. Now $head(t_1\ t_2) = x$ implies that $head(t'_1) = x$ so by the induction hypothesis there exists a term h and types ϕ'_1 and ϕ'_2 such that $ctype_\phi(x, t'_1) = \Pi h : \phi'_1.\phi'_2$.

Finally, we prove part four. This is a proof by induction on the structure of $t_1\ t_2$.

The only possiblities for the form of t_1 is x or a $\hat{t}_1\ \hat{t}_2$. All other forms would not result in $[t/x]^\phi t_1$ being a λ-abstraction and t_1 not. Suppose $t_1 \equiv x$. Then by inversion on the assumption $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$ there exists types ϕ_1 and ϕ_2 such that $\Gamma, x : \phi, \Gamma' \vdash t_1 : \Pi z : \phi_1.\phi_2$, $\Gamma, x : \phi, \Gamma', z : \phi_1 \vdash t_2 : \phi_2$, and $\Gamma, x : \phi, \Gamma' \vdash \phi' \approx [t_2/x]\phi_2$. Now in this case it must be that $\phi \equiv \Pi z : \phi_1.\phi_2$ and $ctype_\phi(x, x\ t_2) = [t_2/x]\phi_2$ when $ctype_\phi(x, x) = \phi \equiv \Pi z : \phi_1.\phi_2$ in this case.

Now suppose $t_1 \equiv (\hat{t}_1\ \hat{t}_2)$. Now knowing t'_1 to not be a λ-abstraction implies that $\hat{t}_1$ is also not a λ-abstraction or $[t/x]^\phi t_1$ would be an application instead of a λ-abstraction. So it must be the case that $[t/x]^\phi \hat{t}_1$ is a λ-abstraction and $\hat{t}_1$ is not. Since $t_1\ t_2 > t_1$ we can apply the induction hypothesis to obtain there exists a type ψ such that $ctype_\phi(x, \hat{t}_1) = \psi$.

Now by inversion on $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$ we know there exists a term z and types ϕ_1 and ϕ_2 such that $\Gamma, x : \phi, \Gamma' \vdash t_1 : \Pi z : \phi_1.\phi_2$ and $\Gamma, x : \phi, \Gamma', z : \phi_1 \vdash t_2 : \phi_2$, and $\Gamma, x : \phi, \Gamma' \vdash \phi' \approx [t_2/z]\phi_2$. We know $t_1 \equiv (\hat{t}_1\ \hat{t}_2)$ so by inversion on $\Gamma, x : \phi, \Gamma' \vdash t_1 : \Pi z : \phi_1.\phi_2$

we know there exists a term j and types ψ_1 and ψ_2 such that $\Gamma, x : \phi, \Gamma' \vdash \hat{t}_1 : \Pi j : \psi_1.\psi_2$, $\Gamma, x : \phi, \Gamma', j : \psi_1 \vdash \hat{t}_2 : \psi_2$, and $\Gamma, x : \phi, \Gamma' \vdash \Pi z : \phi_1.\phi_2 \approx [\hat{t}_2/j]\psi_2$.

By part two of Lemma 55 we know $\Gamma, x : \phi, \Gamma' \vdash \psi \approx \Pi j : \psi_1.\psi_2$ and by Lemma 53 $\psi \equiv \Pi h : \psi'_1.\psi'_2$ for some term h and types ψ'_1 and ψ'_2. At this point $ctype_\phi(x, t_1) = ctype_\phi(x, \hat{t}_1\ \hat{t}_2) = [\hat{t}_2/h]\psi'_2$ when $ctype_\phi(x, \hat{t}_1) = \Pi h : \psi'_1.\psi'_2$.

B.17 Proof of Total, Type Preserving, and Sound with Respect to Reduction

First, we prove part one. This is a proof by induction on the lexicorgraphic combination (ϕ, t') of $>_{\Gamma,\Gamma'}$ and the strict subexpression ordering. We case split on t'.

Case. Suppose t' is either x or a variable y distinct from x. Trivial in both cases.

Case. Suppose $t' \equiv join$. Trivial.

Case. Suppose $t' \equiv \lambda y : \phi_1.t'_1$. First, since t' is a λ-abstraction we know $\phi' \equiv \Pi y : \phi_1.\phi_2$ for some type ϕ_2. To make use of the induction hypothesis we first must show that t'_1 is typeable. Applying the inversion lemma to the assumption $\Gamma, x : \phi, \Gamma' \vdash \lambda y : \phi_1.t'_1 : \Pi y : \phi_1.\phi_2$ yeilds (i) $\Gamma, x : \phi, \Gamma', y : \phi_1 \vdash t'_1 : \phi'_2$ for some type ϕ'_2 such that (ii) $\Gamma, x : \phi, \Gamma' \vdash \Pi y : \phi_1.\phi'_2 \approx \Pi y : \phi_1.\phi_2$. Lastly, by applying the substitution for syntactic equality we obtain $\Gamma, [t/x]\Gamma' \vdash [t/x](\Pi y : \phi_1.\phi'_2) \approx [t/x](\Pi y : \phi_1.\phi_2)$. At this point we can apply the induction hypothesis.

We know $t' > t'_1$, so by the first part of the induction hypothesis there exists a term $\hat{t}'_1$ and type ψ such that $[t/x]^\phi t'_1 = \hat{t}'_1$, $\Gamma, [t/x]\Gamma', y : [t/x]\phi_1 \vdash \hat{t}'_1 : \psi$, and $\Gamma, [t/x]\Gamma', y : [t/x]\phi_2 \vdash \psi \approx [t/x]\phi'_2$. Applying syntactic conversion on $\Gamma, [t/x]\Gamma', y : [t/x]\phi_1 \vdash \hat{t}'_1 : \psi$ using $\Gamma, [t/x]\Gamma', y : [t/x]\phi_2 \vdash \psi \approx [t/x]\phi'_2$ we obtain $\Gamma, [t/x]\Gamma', y : [t/x]\phi_1 \vdash \hat{t}'_1 : [t/x]\phi'_2$ and by applying the λ-abstraction typing rule we obtain $\Gamma, [t/x]\Gamma' \vdash \lambda y : [t/x]\phi_1.\hat{t}'_1 : \Pi y : [t/x]\phi_1.[t/x]\phi'_2$ By the definition of the hereditary substitution function $[t/x]^\phi t' = \lambda y : \phi_1.[t/x]^\phi t'_1 = \lambda y : \phi_1.\hat{t}'_1$. It suffices to show that $\Gamma, [t/x]\Gamma' \vdash \Pi y : [t/x]\phi_1.[t/x]\phi'_2 \approx \Pi y : [t/x]\phi_1.[t/x]\phi_2$. We obtain this by simply applying the substitution for syntactic type equality to (iii) using t.

Case. Suppose $t' \equiv \Lambda X : *_l.t'_1$. Similar to the previous case.

Case. Suppose $t' \equiv t'_1\ t'_2$. Note that $\phi' \equiv [t'_2/y]\phi''$ for some variable y and type ϕ''. We first must show that there exists types ϕ_1 and ϕ_2 such that

$$\begin{array}{l}\Gamma, x : \phi, \Gamma' \vdash t'_1 : \Pi y : \phi_1.\phi_2,\\ \Gamma, x : \phi, \Gamma' \vdash t'_2 : \phi_1, \text{ and}\\ \Gamma, x : \phi, \Gamma' \vdash [t'_2/y]\phi'' \approx [t'_2/y]\phi_2.\end{array}$$

Applying the syntactic inversion lemma to the assumption $\Gamma, x : \phi, \Gamma' \vdash t'_1\ t'_2 : \phi'$ yeilds there exists types ϕ_1 and ϕ_2, and a term y such that

$$\begin{array}{l}\Gamma, x : \phi, \Gamma' \vdash t'_1 : \Pi y : \phi_1.\phi_2,\\ \Gamma, x : \phi, \Gamma' \vdash t'_2 : \phi_1, \text{ and}\\ \Gamma, x : \phi, \Gamma' \vdash [t'_2/y]\phi'' \approx [t'_2/y]\phi_2.\end{array}$$

Now $t' > t'_1$ and $t' > t'_2$. So by two applications of part one and two of the induction hypothesis there exists terms m_1 and m_2, and types ψ_1 and ψ_2 such that

$[t/x]^{\phi}t'_1 = m_1$,
$[t/x]t'_1 \rightsquigarrow^* m_1$
$[t/x]^{\phi}t'_2 = m_2$,
$[t/x]t'_2 \rightsquigarrow^* m_2$
$\Gamma, [t/x]\Gamma' \vdash m_1 : \psi_1$,
$\Gamma, [t/x]\Gamma' \vdash m_2 : \psi_2$,
$\Gamma, [t/x]\Gamma' \vdash \psi_1 \approx [t/x](\Pi y : \phi_1.\phi_2)$, and
$\Gamma, [t/x]\Gamma' \vdash \psi_2 \approx [t/x]\phi_1$.

Now $\Gamma, [t/x]\Gamma' \vdash \psi_1 \approx [t/x](\Pi y : \phi_1.\phi_2)$ is equivalent to $\Gamma, [t/x]\Gamma' \vdash \psi_1 \approx \Pi y : [t/x]\phi_1.[t/x]\phi_2$. By applying the syntactic conversion lemma to

$\Gamma, [t/x]\Gamma' \vdash m_1 : \psi_1$ and
$\Gamma, [t/x]\Gamma' \vdash m_2 : \psi_2$

using the previous type equality we obtain

(i) $\Gamma, [t/x]\Gamma' \vdash m_1 : \Pi y : [t/x]\phi_1.[t/x]\phi_2$, and
(ii) $\Gamma, [t/x]\Gamma' \vdash m_2 : [t/x]\phi_1$.

We case split on whether or not m_1 is a λ-abstraction and t'_1 is not, or $ctype_{\phi}(x, t'_1)$ is undefined. Suppose m_1 is not a λ-abstraction or $ctype_{\phi}(x, t'_1)$ is undefined. Then $[t/x]^{\phi}(t'_1\ t'_2) = ([t/x]^{\phi}t'_1)\ ([t/x]^{\phi}t'_2) = m_1\ m_2$. It suffices to show that

$\Gamma, [t/x]\Gamma' \vdash m_1\ m_2 : \phi'''$ and
$\Gamma, [t/x]\Gamma' \vdash \phi''' \approx [t/x]\phi'$

for some ϕ'''. By applying the application typing rule to (i) and (ii) we obtain

$\Gamma, [t/x]\Gamma' \vdash m_1\ m_2 : [m_2/y]([t/x]\phi_2)$, which is equivalent to
$\Gamma, [t/x]\Gamma' \vdash m_1\ m_2 : [t/x]([m_2/y]\phi_2)$.

At this point it suffices to show that $\Gamma, [t/x]\Gamma' \vdash [t/x]([m_2/y]\phi_2) \approx [t/x]([t'_2/y]\phi'')$. We can see from above that $\Gamma, x : \phi, \Gamma' \vdash [t'_2/y]\phi'' \approx [t'_2/y]\phi_2$ and by Lemma 52 $\Gamma, [t/x]\Gamma' \vdash [t/x]([t'_2/y]\phi'') \approx [t/x]([t'_2/y]\phi_2)$. By symmetry of $\Gamma, [t/x]\Gamma' \vdash [t/x]([t'_2/y]\phi'') \approx [t/x]([t'_2/y]\phi_2)$ we have $\Gamma, [t/x]\Gamma' \vdash [t/x]([t'_2/y]\phi_2) \approx [t/x]([t'_2/y]\phi'')$. Now we know from above that $[t/x]t'_2 \rightsquigarrow^* m_2$ so by applying the join typing rule we obtain $\Gamma, [t/x]\Gamma' \vdash join : [t/x]t'_2 = m_2$. Finally, by applying TE_{Q_2} we obtain $\Gamma, [t/x]\Gamma' \vdash [t/x]([m_2/y]\phi_2) \approx [t/x]([t'_2/y]\phi'')$.

Suppose $m_1 \equiv \lambda y : \phi_1.m'_1$ and t'_1 is not a λ-abstraction. By Lemma 55, Lemma 22, and Lemma 53 there exists a type ψ such that $ctype_{\phi}(x, t'_1) = \psi$, $\Gamma, x : \phi, \Gamma' \vdash \psi \equiv \Pi y : \phi_1.\phi_2$, ψ is a Π-type, ψ is a subexpression of ϕ, $\phi >_{\Gamma,\Gamma'} \phi_1$ and $\phi >_{\Gamma,\Gamma',y:\phi_1} \phi_2$. According to the definition of the hereditary substitution function $[t/x]^{\phi}(t'_1\ t'_2) = [m_2/y]^{\phi_1}m'_1$. Recall from above that we know m_2 and m_1 are typeable, but we have to show that m'_1 is typeable. First, note that $\Gamma, [t/x]\Gamma' \vdash m_1 : \Pi y : [t/x]\phi_1.[t/x]\phi_2$ is equivalent to $\Gamma, [t/x]\Gamma' \vdash m_1 : \Pi y : \phi_1.\phi_2$, x and m_1 have the same type. This implies that $\Gamma, [t/x]\Gamma' \vdash m_2 : \phi_2$. By applying the inverison lemma to $\Gamma, [t/x]\Gamma' \vdash m_1 : \Pi y : \phi_1.\phi_2$ we obtain the there exists a type ϕ'_2 such that

$\Gamma, [t/x]\Gamma', y : \phi_1 \vdash m'_1 : \phi'_2$ and
$\Gamma, [t/x]\Gamma' \vdash \Pi y : \phi_1.\phi_2 \approx \phi'_2$.

Now $\phi >_{\Gamma,\Gamma'} \phi_1$ so we can apply the induction hypothesis to obtain there exists a term m and a type ψ such that $[m_2/y]^{\phi_1}m'_1 = m$, $\Gamma, [t/x]\Gamma' \vdash m : \psi$, and $\Gamma, [t/x]\Gamma' \vdash \psi \approx [m_2/y]\phi_2$. By syntactic conversion using $\Gamma, [t/x]\Gamma' \vdash \psi \approx \phi_2$ we have $\Gamma, [t/x]\Gamma' \vdash m : [m_2/y]\phi_2$. It suffices to show that $\Gamma, [t/x]\Gamma' \vdash [m_2/y]\phi_2 \approx [t/x]([t'_2/y]\phi'')$. We know from above that $\Gamma, x : \phi, \Gamma' \vdash [t'_2/y]\phi'' \approx [t'_2/y]\phi_2$. By symetery and the substitution for syntactic type equality lemma using the previous type equality $\Gamma, [t/x]\Gamma' \vdash [t/x]([t'_2/y]\phi_2) \approx [t/x]([t'_2/y]\phi'')$. We know x is not free in ϕ_2 so the previous type equality is equivalent to $\Gamma, [t/x]\Gamma' \vdash [t'_2/y]\phi_2 \approx [t/x]([t'_2/y]\phi'')$.

Recall from above that $[t/x]t'_2 \rightsquigarrow^* m_2$ so by applying the join typing rule $\Gamma, [t/x]\Gamma' \vdash join : [t/x]t'_2 = m_2$. Finally, by applying TE_{Q_2} we obtain $\Gamma, [t/x]\Gamma' \vdash [m_2/y]\phi_2 \approx [t/x]([t'_2/y]\phi'')$.

Case. Suppose $t' \equiv t'_1[\phi'']$. Similar to the previous case.

Next we show part two. This is a proof by induction on the lexicorgraphic combination (ϕ, t') of $>_\Gamma$ and the strict subexpression ordering. We case split on the structure of t'.

Case. Suppose t' is a variable x or y distinct from x. Trivial in both cases.

Case. Suppose $t' \equiv join$. Trivial.

Case. Suppose $t' \equiv \lambda y : \phi_1.\hat{t}$. Then we know $\phi' \equiv \Pi y : \phi_1.\phi_2$ for some type ϕ_2 and $[t/x]^\phi(\lambda y : \phi_1.\hat{t}) = \lambda y : \phi_1.([t/x]^\phi\hat{t})$. Now $t' > \hat{t}$, but before we can apply the induction hypothesis we first must show that $\hat{t}$ is typeable. By inversion on the typing assumption of t' we know there exists a type ϕ'_2 such that $\Gamma, x : \phi, \Gamma', y : \phi_1 \vdash \hat{t} : \phi'_2$ and $\Gamma, x : \phi, \Gamma' \vdash \Pi y : \phi_1.\phi'_2 \approx \Pi y : \phi_1.\phi_2$. So we can now apply part one of the induction hypothesis to obtain $[t/x]^\phi\hat{t}$ has a result and by part two we know $[t/x]\hat{t} \rightsquigarrow^* [t/x]^\phi\hat{t}$. At this point we can see that since $\lambda y : \phi'.[t/x]\hat{t} \equiv [t/x](\lambda y : \phi'.\hat{t})$ and we may conclude that $\lambda y : \phi'.[t/x]\hat{t} \rightsquigarrow^* \lambda y : \phi'.[t/x]^\phi\hat{t}$.

Case. Suppose $t' \equiv \Lambda X : *_l.\hat{t}$. Similar to the previous case.

Case. Suppose $t' \equiv t'_1\ t'_2$. Now $t' > t'_1$ and $t' > t'_2$, but before we can apply the induction hypothesis we must first show that t'_1 and t'_2 are typeable. By inverison on the typing assumption of t' we know there exists a term y and types ϕ'_1 and ϕ'_2 such that

$$\begin{array}{l}\text{(i) } \Gamma, x : \phi, \Gamma' \vdash t'_1 : \Pi y : \phi'_1.\phi'_2,\\ \Gamma, x : \phi, \Gamma' \vdash t'_2 : \phi'_2, \text{ and}\\ \Gamma, x : \phi, \Gamma' \vdash [t'_2/y]\phi'_2 \approx [t'_2/y]\phi_2,\end{array}$$

where $\phi' \equiv [t'_2/y]\phi_2$. So now we can apply the induction hypothesis. By part one of the induction hypothesis we know there exists terms $\hat{t}'_1$ and $\hat{t}'_2$ such that $[t/x]^\phi t'_1 = \hat{t}'_1$ and $[t/x]^\phi t'_2 = \hat{t}'_2$ and by part two of the induction hypothesis we know $[t/x]t'_1 \rightsquigarrow^* \hat{t}'_1$ and $[t/x]t'_2 \rightsquigarrow^* \hat{t}'_2$.

Now we case split on whether or not $\hat{t}'_1$ is a λ-abstraction and t'_1 is not, or $ctype_\phi(x, t'_1)$ is undefined. Suppose $\hat{t}'_1$ is not a λ-abstraction, or $ctype_\phi(x, t'_1)$ is undefined. Then $[t/x]^\phi t' = ([t/x]^\phi t'_1)\ ([t/x]^\phi t'_2) \equiv \hat{t}'_1\ \hat{t}'_2$. Thus, $[t/x]t' \rightsquigarrow^* [t/x]^\phi t'$, because $[t/x]t' = ([t/x]t'_1)\ ([t/x]t'_2)$. So suppose $\hat{t}'_1 \equiv \lambda y : \phi'_1.\hat{t}''_1$ and t'_1 is not a λ-abstraction. By Lemma 55, Lemma 22, and Lemma 53 there exists a type ψ such that $ctype_\phi(x, t'_1) = \psi$, $\Gamma, x : \phi, \Gamma' \vdash \psi \equiv \Pi y : \phi'_1.\phi'_2$, ψ is a Π-type, ψ is a subexpression of ϕ, $\phi >_{\Gamma,\Gamma'} \phi'_1$ and $\phi >_{\Gamma,\Gamma',y:\phi_1} \phi'_2$. Then by the definiton of the hereditary substitution function $[t/x]^\phi(t'_1\ t'_2) = [\hat{t}'_2/y]^{\phi'_1}\hat{t}''_1$. Again we must show that $\hat{t}''_1$ is typeable before using the induction hypothesis. This is a simple consequence of applying inversion to (i) above. Thus, we can apply part one of the induction hypothesis to obtain $[\hat{t}'_2/y]^{\phi'_1}\hat{t}''_1$ has a result and by part two of the induction hypothesis to obtain $[\hat{t}'_2/y]\hat{t}''_1 \rightsquigarrow^* [\hat{t}'_2/y]^{\phi'_1}\hat{t}''_1$. Now by knowing that $(\lambda y : \phi'_1.\hat{t}''_1)\ t'_2 \rightsquigarrow [\hat{t}'_2/y]\hat{t}''_1$ and by the previous fact we know $(\lambda y : \phi'_1.\hat{t}''_1)\ t'_2 \rightsquigarrow^* [\hat{t}'_2/y]^{\phi'_1}\hat{t}''_1$. We now make use of the well known result of full β-reduction. The result is stated as

$$\frac{\begin{array}{l}a \rightsquigarrow^* a'\\ b \rightsquigarrow^* b' \qquad a'\ b' \rightsquigarrow^* c\end{array}}{a\ b \rightsquigarrow^* c}$$

where a, a', b, b', and c are all terms. We apply this result by instantiating a, a', b, b', and c with $[t/x]t'_1$, $\hat{t}'_1$, $[t/x]t'_2$, $\hat{t}'_2$, and $[\hat{t}'_2/y]^{\phi'_1}\hat{t}''_1$ respectively. Therefore, $[t/x](t'_1\ t'_2) \rightsquigarrow^* [\hat{t}'_2/y]^{\phi'_1}\hat{t}''_1$.

Case. Suppose $t' \equiv t'_1[\phi'']$. Then $\phi' \equiv [\phi''/X]\psi$. Now by inversion on the assumed typing result of t' we know there exists a type ψ' and a term X such that $\Gamma, x : \phi, \Gamma' \vdash t'_1 : \forall X : *_l.\psi'$, $\Gamma, x : \phi, \Gamma' \vdash \psi' : *_l$, and $\Gamma, x : \phi, \Gamma' \vdash [\phi''/X]\psi \approx [\phi''/X]\psi'$. Since $t' > t'_1$ and t'_1 is typeable we can apply the induction hypothesis to obtain $[t/x]t'_1 \rightsquigarrow^* [t'/x]^\phi t'_1$. We case split on whether or not $[t'/x]^\phi t'_1$ is a type abstraction and t'_1 is not. The case where it is not is trivial so we only consider the case where $[t'/x]^\phi t'_1 \equiv \Lambda X : *_l.s'$. Then $[t'/x]^\phi t' = [\phi'/X]s'$. Now we have $[t/x]t'_1 \rightsquigarrow^* [t'/x]^\phi t'_1$ and $[t/x](t'_1[\phi]) \equiv ([t/x]t'_1)[\phi] \rightsquigarrow^* ([t'/x]^\phi t'_1)[\phi] \rightsquigarrow [\phi/X]s'$. Thus, $[t/x]t' \rightsquigarrow^* [t'/x]^\phi t'$.

B.18 Proof of Corollary 57

We know by Lemma 56 there exists a term t'' and a type ϕ'' such that $[t/x]^\phi t' = t''$, $\Gamma, \Gamma' \vdash t'' : \phi''$, and $\Gamma, \Gamma' \vdash \phi'' \approx [t/x]\phi'$. So by Lemma 16 $\Gamma, \Gamma' \vdash t'' : [t/x]\phi'$.

B.19 Proof of Redex Preservation

This is a proof by induction on the lexicorgraphic combination (ϕ, t') of $>_\Gamma$ and the strict subexpression ordering. We case split on the structure of t' and we do not state explicit typing results when applying the induction hypothesis. All such results are simple applications of the inversion lemma.

Case. Let $t' \equiv x$ or $t' \equiv y$ where y is distinct from x. Trivial.

Case. Suppose $t' \equiv join$. Trivial.

Case. Let $t' \equiv \lambda x : \phi_1.t''$. Then $[t/x]^\phi t' \equiv \lambda x : \phi_1.[t/x]^\phi t''$. Now

$$\begin{array}{rcl} rset(\lambda x : \phi_1.t'', t) & = & rset(\lambda x : \phi_1.t'') \cup rset(t) \\ & = & rset(t'') \cup rset(t) \\ & = & rset(t'', t). \end{array}$$

We know that $t' > t''$ by the strict subexpression ordering, hence by the induction hypothesis $|rset(t'', t)| \geq |rset([t/x]^\phi t'')|$ which implies $|rset(t', t)| \geq |rset([t/x]^\phi t')|$.

Case. Let $t' \equiv \Lambda X : *_l.t''$. Similar to the previous case.

Case. Let $t' \equiv t'_1\ t'_2$. First consider when t'_1 is not a λ-abstraction. Then

$$rset(t'_1\ t'_2, t) = rset(t'_1, t'_2, t)$$

Clearly, $t' > t'_i$ for $i \in \{1, 2\}$, hence, by the induction hypothesis $|rset(t'_i, t)| \geq |rset([t/x]^\phi t'_i)|$. We have two cases to consider. That is whether or not $[t/x]^\phi t'_1$ is a λ-abstraction or not. Suppose so. Then by Lemma 4 $ctype_\phi(x, t'_1) = \psi$ and by inversion on $\Gamma, x : \phi, \Gamma' \vdash t'_1\ t'_2 : \phi'$ there exists a term j and types ϕ_1 and ϕ_2 such that $\Gamma, x : \phi, \Gamma' \vdash t_1 : \Pi j : \phi_1.\phi_2, \Gamma, x : \phi, \Gamma', j : \phi_1 \vdash t_2 : \phi_2$ and $\Gamma, x : \phi, \Gamma' \vdash \phi' \approx [t_2/j]\phi_2$. Again, by Lemma 4 there exists a type ψ such that $ctype_\phi(x, t'_1) = \psi$, ψ is a subexpression of ϕ, and $\Gamma, x : \phi, \Gamma' \vdash \psi \approx \Pi j : \phi_1.\phi_2$. By Lemma 53 ψ is a Π-type and by Lemma 22 $\phi >_{\Gamma,\Gamma'} \phi_1$ and $\phi >_{\Gamma, j:\phi_1} \phi_2$. So by the definition of the hereditary substitution function $[t/x]^\phi t'_1\ t'_2 = [([t/x]^\phi t'_2)/y]^{\phi_1} t''_1$, where $[t/x]^\phi t'_1 = \lambda y : \phi_1.t''_1$. Hence,

$$|rset([t/x]^\phi t'_1\ t'_2)| = |rset([([t/x]^\phi t'_2)/y]^{\phi_1} t''_1)|.$$

Now $\phi >_{\Gamma,\Gamma'} \phi_1$ so by the induction hypothesis

$$\begin{array}{rcl} |rset([([t/x]^\phi t'_2)/y]^{\phi_1} t''_1)| & = & |rset([t/x]^\phi t'_2, t''_1)| \\ & = & |rset(t'_2, t''_1, t)| \\ & \geq & |rset(t'_2, [t/x]^\phi t'_1, t)| \\ & = & |rset(t'_2, t'_1, t)| \\ & = & |rset(t'_1, t'_2, t)|. \end{array}$$

Suppose $[t/x]^\phi t'_1$ is not a λ-abstractions or $ctype_\phi(x, t'_1)$ is undefined. Then

$$\begin{array}{rcl} rset([t/x]^\phi(t'_1\ t'_2)) & = & rset([t/x]^\phi t'_1\ [t/x]^\phi t'_2) \\ & = & rset([t/x]^\phi t'_1, [t/x]^\phi t'_2). \\ & \leq & rset(t'_1, t'_2, t). \end{array}$$

Next suppose $t'_1 \equiv \lambda y : \phi_1.t''_1$. Then

$$rset((\lambda y : \phi_1.t_1'')\ t_2', t) \quad = \quad \{(\lambda y : \phi_1.t_1'')\ t_2'\} \cup rset(t_1'', t_2', t).$$

By the definition of the hereditary substitution function,

$$\begin{array}{rcl} rset([t/x]^{\phi}(\lambda y : \phi_1.t_1'')\ t_2') & = & rset([t/x]^{\phi}(\lambda y : \phi_1.t_1'')\ [t/x]^{\phi}t_2') \\ & = & rset((\lambda y : \phi_1.[t/x]^{\phi}t_1'')\ [t/x]^{\phi}t_2') \\ & = & \{(\lambda y : \phi_1.[t/x]^{\phi}t_1'')\ [t/x]^{\phi}t_2'\} \cup rset([t/x]^{\phi}t_1'') \cup rset([t/x]^{\phi}t_2'). \end{array}$$

Since $t' > t_1''$ and $t' > t_2'$ we can apply the induction hypothesis to obtain, $|rset(t_1'', t)| \geq |rset([t/x]^{\phi}t_1'')|$ and $|rset(t_2', t)| \geq |rset([t/x]^{\phi}t_2')|$. Therefore,
$|\{(\lambda y : \phi_1.t_1'')\ t_2'\} \cup rset(t_1'', t) \cup rset(t_2', t)| \geq |\{(\lambda y : \phi_1.[t/x]^{\phi}t_1'')\ [t/x]^{\phi}t_2'\} \cup rset([t/x]^{\phi}t_1'') \cup rset([t/x]^{\phi}t_2')|$.

Case. Suppose $t' \equiv t_1'[\phi'']$. It suffices to show that $|rset(t, t')| = |rset([t/x]^{\phi}t')|$. Now

$$\begin{array}{rcl} |rset(t, t')| & = & |rset(t, t_1'[\phi''])| \\ & = & |rset(t) \cup rset(t_1'[\phi''])| \\ & = & |rset(t) \cup rset(t_1')| \\ & = & |rset(t, t_1')|. \end{array}$$

and

$$|rset[t/x]^{\phi}t')| = |rset([t/x]^{\phi}(t_1'[\phi'']))|.$$

We have several cases to consider. Suppose t_1' and $[t/x]^{\phi}t_1'$ are not type abstractions. Then

$$\begin{array}{rcl} |rset([t/x]^{\phi}(t_1'[\phi'']))| & = & |rset(([t/x]^{\phi}t_1')[\phi''])| \\ & = & |rset([t/x]^{\phi}t_1')|. \end{array}$$

We can see that $t' > t_1'$ so by the induction hypothesis

$$\begin{array}{rcl} |rset([t/x]^{\phi}t_1')| & \leq & |rset(t, t_1')| \\ & = & |rset(t, t')|. \end{array}$$

Suppose $t_1' \equiv \Lambda X : *_l.t_1''$. Then

$$\begin{array}{rcl} |rset(t, t')| & = & |rset(t, t_1'[\phi''])| \\ & = & |\{t_1'[\phi'']\} \cup rset(t, t_1'')| \end{array}$$

and

$$\begin{array}{rcl} |rset([t/x]^{\phi}t')| & = & |rset([t/x]^{\phi}(t_1'[\phi''])| \\ & = & |rset((\Lambda X : *_l.[t/x]^{\phi}t_1'')[\phi''])| \\ & = & |\{(\Lambda X : *_l.[t/x]^{\phi}t_1'')[\phi'']\} \cup rset([t/x]^{\phi}t_1'')|. \end{array}$$

Again, $t' > t_1'$ so by the induciton hypothesis $|rset([t/x]^{\phi}t_1'')| \leq |rset(t, t_1'')$. Thus, $|rset(t, t')| \leq |rset([t/x]^{\phi}t')|$.

Suppose $t_1' \equiv x[\phi'']$ and $t \equiv \lambda X : *_l.t''$. Then

$$\begin{array}{rcl} |rset(t, t')| & = & |rset(t)| \\ & = & |rset(t'')| \end{array}$$

and

$$\begin{array}{rcl} |rset([t/x]^{\phi}t')| & = & |rset([\phi''/X]t'')| \\ & = & |rset(t''). \end{array}$$

Therefore, $|rset(t, t')| = |rset([t/x]^{\phi}t')|$.

B.20 Proof of Normality Preservation

By Lemma 56 we know there exists a term n'' such that $[n/x]^{\phi}n' = n''$ and by Lemma 58 $|rset(n', n)| \geq |rset([n/x]^{\phi}n')|$. Hence, $|rset(n', n)| \geq |rset(t)|$, but $|rset(n', n)| = 0$. Therefore, $|rset(t)| = 0$ which implies n'' is normal.

B.21 Proof of Type Substitution for the Interpretation of Types

By the definition of the interpretation of types, $\Gamma, X : *_l, \Gamma' \vdash n : \phi'$ and by Lemma 46, $\Gamma, [\phi/X]\Gamma' \vdash [\phi/X]n : [\phi/X]\phi'$. Finally, by the definition of the interpretation of types, $[\phi/X]n \in [\![[\phi/X]\phi']\!]_{\Gamma,[\phi/X]\Gamma'}$.

B.22 Proof of Semantic Equality

We first prove the left to right containment. Suppose $t \rightsquigarrow^* n \in [\![[t_1/x]\phi]\!]_\Gamma$. Then by the definition of the interpretation of types, $\Gamma \vdash n : [t_1/x]\phi$. By assumption we know $\Gamma \vdash p : t_1 = t_2$, hence, by applying the conversion type-checking rule $\Gamma \vdash n : [t_2/x]\phi$. Finally, by the definition of the interpretation of types, $n \in [\![[t_2/x]\phi]\!]_\Gamma$. Therefore, $t \in [\![[t_2/x]\phi]\!]_\Gamma$. The opposite direction is similar.

B.23 Proof of Lemma 44

Assume $\phi >_\Gamma \phi'$ for some types ϕ and ϕ'. We case split on the form of ϕ. Clearly, ϕ is not a type variable.

Case. Suppose $\phi \equiv \Pi x : \phi_1.\phi_2$. Then ϕ' must be of the form ϕ_1 or $[t/x]\phi_2$, for some term $\Gamma \vdash t : \phi_1$. In both cases we have two cases to consider; either ϕ and ϕ' have the same level or they do not. Consider the first form and suppose they have the same level. Then it is clear that $depth(\phi) > depth(\phi')$. Now consider the latter form and suppose ϕ and ϕ' have the same level. Then clearly $depth(\phi) > depth(\phi')$. In either form if the level of ϕ and ϕ' are different, then the level of ϕ is larger than the level of ϕ'. In all cases $(l, d) > (l', d')$.

Case. Suppose $\phi \equiv \forall X : *_l.\phi_1$. Then ϕ' must be of the form $[\phi_2/X]\phi_1$ for some type $\Gamma \vdash \phi_2 : *_l$. It is obvious that the level of ϕ is always larger than the level of ϕ'. Hence, $(l, d) > (l', d')$.

B.24 Proof of Substitution for the Interpretation of Types

It suffices to show that $\Gamma, [n/x]\Gamma' \vdash [n/x]^\phi n' : [n/x]\phi'$ and $[n/x]^\phi n'$ is normal. By Corollary 57 we know $\Gamma, [n/x]\Gamma' \vdash [n/x]^\phi n' : [n/x]\phi'$ and by Lemma 59 $[n/x]^\phi n'$ is normal. Therefore, $[n/x]^\phi n' \in [\![[n/x]\phi']\!]_{\Gamma,[n/x]\Gamma'}$.

B.25 Proof of Type Soundness

This is a proof by induction on the structure of the typing derivation of t.

Case.

$$\frac{\Gamma(x) = \phi \qquad \Gamma\ Ok}{\Gamma \vdash x : \phi}$$

By regularity $\Gamma \vdash \phi : *_l$ for some l, hence $[\![\phi]\!]_\Gamma$ is nonempty. Clearly, $x \in [\![\phi]\!]_\Gamma$ by the definition of the interpretation of types.

Case.

$$\frac{\Gamma, x : \phi_1 \vdash t : \phi_2}{\Gamma \vdash \lambda x : \phi_1.t : \Pi x : \phi_1.\phi_2}$$

By the induction hypothesis and the definition of the interpretation of types $t \in [\![\phi_2]\!]_{\Gamma,x:\phi_1}$, $t \rightsquigarrow^! n \in [\![\phi_2]\!]_{\Gamma,x:\phi_1}$ and $\Gamma, x : \phi_1 \vdash n : \phi_2$. Thus, by applying the λ-abstraction type-checking rule, $\Gamma \vdash \lambda x : \phi_1.n : \Pi x : \phi_1.\phi_2$, hence by the definition of the interpretation of types $\lambda x : \phi_1.n \in [\![\Pi x : \phi_1.\phi_2]\!]_\Gamma$. Since $\lambda x : \phi_1.t \rightsquigarrow^! \lambda x : \phi_1.n \in [\![\Pi x : \phi_1.\phi_2]\!]_\Gamma$ we know by the definition of the interpretation of types $\lambda x : \phi_1.t \in [\![\Pi x : \phi_1.\phi_2]\!]_\Gamma$.

Case.

$$\frac{\Gamma \vdash t_1 : \Pi x : \phi_1.\phi_2 \qquad \Gamma \vdash t_2 : \phi_1}{\Gamma \vdash t_1\ t_2 : [t_2/x]\phi_2}$$

It suffices to show that there exists a normal term n such that $t_1\ t_2 \rightsquigarrow^! n \in [\![[t_2/x]\phi_2]\!]_\Gamma$. By the induction hypothesis and the definition of the interpretation of types $t_1 \rightsquigarrow^! n_1 \in [\![\Pi x : \phi_1.\phi_2]\!]_\Gamma$, $\Gamma \vdash n_1 : \Pi x : \phi_1.\phi_2$, $t_2 \rightsquigarrow^! n_2 \in [\![\phi_1]\!]_\Gamma$, and $\Gamma \vdash n_2 : \phi_1$. Clearly,

$$\begin{array}{lcl} t_1\ t_2 & \rightsquigarrow^* & n_1\ n_2 \\ & = & [n_1/z](z\ n_2), \end{array}$$

for some fresh variable $z \notin FV(n_1, n_2, \phi_1, \phi_2, x)$. By Lemma 56, Lemma 59, and Corollary 57 $[n_1/x](z\ n_2) \rightsquigarrow^* [n_1/x]^{\phi_1}(z\ n_2)$, $[n_1/x]^{\phi_1}(z\ n_2)$ is normal, and $\Gamma \vdash [n_1/x]^{\phi_1}(z\ n_2) : [n_2/x]\phi_2$. Thus, $t_1\ t_2 \rightsquigarrow^! [n_1/x]^{\phi_1}(z\ n_2)$. It suffices to show that $\Gamma \vdash [n_1/x]^{\phi_1}(z\ n_2) : [t_2/x]\phi_2$. This is justified by the following typing derivation:

$$\frac{\dfrac{\begin{array}{c}\Gamma \vdash t_2 : \phi_1 \\ \Gamma \vdash n_2 : \phi_1 \qquad n_2 \downarrow t_2\end{array}}{\Gamma \vdash join : n_2 = t_2}\ \textsc{Join} \qquad \Gamma \vdash [n_1/x]^{\phi_1}(z\ n_2) : [n_2/x]\phi_2}{\Gamma \vdash [n_1/x]^{\phi_1}(z\ n_2) : [t_2/x]\phi_2}\ \textsc{Conv}$$

Therefore, $[n_1/x]^{\phi_1}(z\ n_2) \in [\![[t_2/x]\phi_2]\!]_\Gamma$ which implies that $t_1\ t_2 \in [\![[t_2/x]\phi_2]\!]_\Gamma$.

Case.

$$\frac{t_1 \downarrow t_2 \qquad \Gamma\ Ok}{\Gamma \vdash join : t_1 = t_2}$$

Clearly, $join \in [\![t_1 = t_2]\!]_\Gamma$ by the definition of the interpretation of types.

Case.

$$\frac{\Gamma \vdash t_0 : t_1 = t_2 \qquad \Gamma \vdash t : [t_1/x]\phi}{\Gamma \vdash t : [t_2/x]\phi}$$

By the induction hypothesis, $t \in [\![[t_1/x]\phi]\!]_\Gamma$. By the definition of the interpretation of types, $t \rightsquigarrow^! n \in [\![[t_1/x]\phi]\!]_\Gamma$. By assumption we know, $\Gamma \vdash t_0 : t_1 = t_2$. Thus, by Lemma 24, $[\![[t_1/x]\phi]\!]_\Gamma = [\![[t_2/x]\phi]\!]_\Gamma$. Therefore, $n \in [\![[t_2/x]\phi]\!]_\Gamma$, hence, by the definition of the interpretation of types, $t \in [\![[t_2/x]\phi]\!]_\Gamma$.

Case.

$$\frac{\Gamma, X : *_p \vdash t : \phi}{\Gamma \vdash \Lambda X : *_p.t : \forall X : *_p.\phi}$$

By the induction hypothesis, $t \in [\![\phi]\!]_{\Gamma, X:*_p}$, so by the definition of the interpretation of types, $t \rightsquigarrow^! n \in [\![\phi]\!]_{\Gamma, X:*_p}$ and $\Gamma, X : *_p \vdash n : \phi$. We can apply the Λ-abstraction type-checking rule to obtain $\Gamma \vdash \Lambda X : *_p.n : \forall X : *_p.\phi$, thus $\Lambda X : *_p.n \in [\![\forall X : *_p.\phi]\!]_\Gamma$. Since $\Lambda X : *_p.t \rightsquigarrow^! \Lambda X : *_p.n$ by definition of the interpretation of types $\Lambda X : *_p.t \in [\![\forall X : *_p.\phi]\!]_\Gamma$.

Case.

$$\frac{\Gamma \vdash t : \forall X : *_l.\phi_1 \qquad \Gamma \vdash \phi_2 : *_l}{\Gamma \vdash t[\phi_2] : [\phi_2/X]\phi_1}$$

By the induction hypothesis $t \in [\![\forall X : *_l.\phi_1]\!]_\Gamma$, so by the definition of the interpretation of types $t \rightsquigarrow^! n \in [\![\forall X : *_l.\phi_1]\!]_\Gamma$ and $\Gamma \vdash n : \forall X : *_l.\phi_1$. We do a case split on whether or not n is a Λ-abstraction. We can apply the type-instantiation type-checking rule to obtain $\Gamma \vdash n[\phi_2] : [\phi_2/X]\phi_1$ and by the definition of the interpretation of types $n[\phi_2] \in [\![[\phi_2/X]\phi_1]\!]_\Gamma$. Therefore, $t \in [\![[\phi_2/X]\phi_1]\!]_\Gamma$. Suppose $n \equiv \Lambda X : *_l.n'$. Then $t[\phi_2] \rightsquigarrow^* (\Lambda X : *_l.n')[\phi_2] \rightsquigarrow [\phi_2/X]n'$.

By semantic inversion $n' \in [\![\phi_1]\!]_{\Gamma,X:*_l}$. Therefore, by Lemma 23 $[\phi_2/X]n' \in [\![[\phi_2/X]\phi_1]\!]_\Gamma$ and $t[\phi_2] \in [\![[\phi_2/X]\phi_1]\!]_\Gamma$, since $t[\phi_2] \downarrow [\phi_2/X]n'$.

C Proofs of Results Pertaining to STLC$^=$

C.1 Miscellaneous Definitions and Results

▶ **Lemma 60** (Weakening for Typing). *If $\Gamma \vdash t : \phi$ then $\Gamma, \Gamma' \vdash t : \phi$ for any context Γ' that does not overlap with Γ.*

Proof. By straightforward induction on the assumed typing derivation. ◀

We define a well-founded ordering on types in the following definition. We do not explicitly prove that this is well-founded, because it is simply the subexpression ordering for types, which is well known to be well-founded.

▶ **Definition 61.** The ordering $>_\Gamma$ is defined as the least relation satisfying the universal closures of the following formulas:

$$\begin{array}{lll} \phi_1 \to \phi_2 & >_\Gamma & \phi_1 \\ \phi_1 \to \phi_2 & >_\Gamma & \phi_2 \end{array} \qquad \begin{array}{lll} \phi_1 = \phi_2 & >_\Gamma & \phi_1 \\ \phi_1 = \phi_2 & >_\Gamma & \phi_2 \end{array}$$

▶ **Definition 62.** The following function constructs the set of redexes within a term:

$$\begin{array}{lll} rset(x) = \emptyset & & \\ rset(\lambda x : \phi.t) = rset(t) & & \\ rset(t_1\ t_2) & & \\ \quad = & rset(t_1, t_2) & \text{if } t_1 \text{ is not a } \lambda\text{-abstraction.} \\ \quad = & \{t_1\ t_2\} \cup rset(t_1', t_2) & \text{if } t_1 \equiv \lambda x : \phi.t_1'. \end{array}$$

The extention of $rset$ to multiple arguments is defined as follows:

$$rset(t_1, \ldots, t_n) =^{def} rset(t_1) \cup \cdots \cup rset(t_n).$$

C.2 Properties of the Hereditary Substitution Function

The following are the properties of the $ctype$ function and the hereditary substitution function for STLC$^=$.

▶ **Lemma 63** (Properties of $ctype_\phi$).

i. *If $ctype_\phi(x, t) = \phi'$ then $head(t) = x$ and ϕ' is a subexpression of ϕ.*
ii. *If $\Gamma, x : \phi, \Gamma' \vdash t : \phi'$ and $ctype_\phi(x, t) = \phi''$ then there exists a term p such that $\Gamma, x : \phi, \Gamma' \vdash p : \phi' = \phi''$.*
iii. *If $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$, $\Gamma \vdash t : \phi$, $[t/x]^\phi t_1 \equiv \lambda y : \phi_1.q$, and t_1 is not then there exists a type ψ such that $ctype_\phi(x, t_1) = \psi$.*

▶ **Lemma 64** (Total and Type Preserving). *Suppose $\Gamma \vdash t : \phi$ and $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$. Then there exists a term t'' such that $[t/x]^\phi t' = t''$ and $\Gamma, \Gamma' \vdash t'' : \phi'$.*

▶ **Lemma 65** (Redex Preserving). *If $\Gamma \vdash t : \phi$ and $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ then $|rset(t', t)| \geq |rset([t/x]^\phi t')|$.*

▶ **Lemma 66** (Normality Preserving). *If $\Gamma \vdash v : \phi$ and $\Gamma, x : \phi' \vdash v' : \phi'$ then there exists a value v'' such that $[v/x]^\phi v' = v''$.*

▶ **Lemma 67** (Soundness with Respect to Reduction). *If* $\Gamma \vdash t : \phi$ *and* $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ *then* $[t/x]t' \rightsquigarrow_\beta^* [t/x]^\phi t'$.

▶ **Corollary 68.** *If* $\Gamma \vdash t : \phi$ *and* $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi_2$ *then* $([t/x]^\phi t_1)\ ([t/x]^\phi t_2) \rightsquigarrow_\beta^* [t/x]^\phi (t_1\ t_2)$.

C.3 Proof of Syntactic Inversion

This is a proof by induction on the form of the assumed typing derivation.

Part i. We have two cases to consider.

Case.

$$\frac{\Gamma \vdash \phi_1 \qquad \Gamma, x : \phi_1 \vdash t : \phi_2}{\Gamma \vdash \lambda x.t : \phi_1 \to \phi_2}\ \textsc{Lam}$$

Trivial.

Case.

$$\frac{\Gamma \vdash \lambda x.t : [\phi_1'/X](\phi_1 \to \phi_2) \qquad \Gamma \vdash t' : \phi_1 = \phi_2'}{\Gamma \vdash \lambda x.t : [\phi_2'/X](\phi_1 \to \phi_2)}\ \textsc{Conv}$$

By the induction hypothesis we know $\Gamma, x : [\phi_1'/X]\phi_1 \vdash t : [\phi_1'/X]\phi_2$ and by an application of the above rule we obtain $\Gamma, x : [\phi_1'/X]\phi_1 \vdash t : [\phi_2'/X]\phi_2$. It suffices to show that $\Gamma, x : [\phi_2'/X]\phi_1 \vdash t : [\phi_2'/X]\phi_2$. This is a simply consequence of applying Lemma **??** to $\Gamma, x : [\phi_2'/X]\phi_1 \vdash t : [\phi_2'/X]\phi_2$ using $\Gamma \vdash t' : \phi_1 = \phi_2'$.

Part ii. Again, we have two cases to consider.

Case.

$$\frac{\Gamma \vdash t_1 : \phi_1 \to \phi_2 \qquad \Gamma \vdash t_2 : \phi_1}{\Gamma \vdash t_1\ t_2 : \phi_2}\ \textsc{App}$$

Trivial.

Case.

$$\frac{\Gamma \vdash t : [\phi_1/X]\phi \qquad \Gamma \vdash t' : \phi_1 = \phi_2}{\Gamma \vdash t : [\phi_2/X]\phi}\ \textsc{Conv}$$

By the induction hypothesis there exists a type ϕ_1', such that, $\Gamma \vdash t_1 : \phi_1' \to [\phi_1/X]\phi$ and $\Gamma \vdash t_2 : \phi_1'$. At this point all we have to do is apply the rule above to $\Gamma \vdash t_1 : \phi_1' \to [\phi_1/X]\phi$ to obtain $\Gamma \vdash t_1 : \phi_1' \to [\phi_2/X]\phi$.

C.4 Proof of Properties of $ctype_\phi$

We prove part one first. This is a proof by induction on the structure of t.

Case. Suppose $t \equiv x$. Then $ctype_\phi(x, x) = \phi$. Clearly, $head(x) = x$ and ϕ is a subexpression of itself.

Case. Suppose $t \equiv t_1\ t_2$. Then $ctype_\phi(x, t_1\ t_2) = \phi''$ when $ctype_\phi(x, t_1) = \phi' \to \phi''$. Now $t > t_1$ so by the induciton hypothesis $head(t_1) = x$ and $\phi' \to \phi''$ is a subexpression of ϕ. Therefore, $head(t_1\ t_2) = x$ and certainly ϕ'' is a subexpression of ϕ.

We now prove part two. This is also a proof by induction on the structure of t.

Case. Suppose $t \equiv x$. Then $ctype_\phi(x, x) = \phi$. Clearly, $\phi \equiv \phi$.

Case. Suppose $t \equiv t_1\ t_2$. Then $ctype_\phi(x, t_1\ t_2) = \phi_2$ when $ctype_\phi(x, t_1) = \phi_1 \to \phi_2$. By inversion on the assumed typing derivation we know there exists type ϕ'' such that $\Gamma, x : \phi, \Gamma' \vdash t_1 : \phi'' \to \phi'$. Now $t > t_1$ so by the induciton hypothesis $\phi_1 \to \phi_2 \equiv \phi'' \to \phi'$. Therefore, $\phi_1 \equiv \phi''$ and $\phi_2 \equiv \phi'$.

Next we prove part three. This is a proof by induction on the structure of $t_1\ t_2$.

The only possiblities for the form of t_1 is x or a $\hat{t}_1\ \hat{t}_2$. All other forms would not result in $[t/x]^\phi t_1$ being a λ-abstraction and t_1 not. If $t_1 \equiv x$ then there exist a type ϕ'' such that $\phi \equiv \phi'' \to \phi'$ and $ctype_\phi(x, x\ t_2) = \phi'$ when $ctype_\phi(x, x) = \phi \equiv \phi'' \to \phi'$ in this case. We know ϕ'' to exist by inversion on $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$.

Now suppose $t_1 \equiv (\hat{t}_1\ \hat{t}_2)$. Now knowing t'_1 to not be a λ-abstraction implies that $\hat{t}_1$ is also not a λ-abstraction or $[t/x]^\phi t_1$ would be an application instead of a λ-abstraction. So it must be the case that $[t/x]^\phi \hat{t}_1$ is a λ-abstraction and $\hat{t}_1$ is not. Since $t_1\ t_2 > t_1$ we can apply the induction hypothesis to obtain there exists a type ψ such that $ctype_\phi(x, \hat{t}_1) = \psi$. Now by inversion on $\Gamma, x : \phi, \Gamma' \vdash t_1\ t_2 : \phi'$ we know there exists a type ϕ'' such that $\Gamma, x : \phi, \Gamma' \vdash t_1 : \phi'' \to \phi'$. We know $t_1 \equiv (\hat{t}_1\ \hat{t}_2)$ so by inversion on $\Gamma, x : \phi, \Gamma' \vdash t_1 : \phi'' \to \phi'$ we know there exists a type ψ'' such that $\Gamma, x : \phi, \Gamma' \vdash \hat{t}_1 : \psi'' \to (\phi'' \to \phi')$. By part two of Lemma 63 we know $\psi \equiv \psi'' \to (\phi'' \to \phi')$ and $ctype_\phi(x, t_1) = ctype_\phi(x, \hat{t}_1\ \hat{t}_2) = \phi'' \to \phi'$ when $ctype_\phi(x, \hat{t}_1) = \psi'' \to (\phi'' \to \phi')$, which holds because we know $ctype_\phi(x, \hat{t}_1) = \psi$.

C.5 Proof of Total and Type Preserving

This is a proof by induction on the lexicorgraphic combination (ϕ, t') of $>_{\Gamma,\Gamma'}$ and the strict subexpression ordering. We case split on t'.

Case. Suppose t' is either x or a variable y distinct from x. Trivial in both cases.

Case. Suppose $t' \equiv \lambda y : \phi_1.t'_1$. By inversion on the typing judgement we know there exists a type ϕ_2 such that $\Gamma, x : \phi, \Gamma', y : \phi_1 \vdash t'_1 : \phi_2$. We also know $t' > t'_1$, hence we can apply the induction hypothesis to obtain $[t/x]^\phi t'_1 = \hat{t}'_1$ and $\Gamma, \Gamma', y : \phi_1 \vdash \hat{t} : \phi_2$ for some term $\hat{t}'_1$. By the definition of the hereditary substitution function $[t/x]^\phi t' = \lambda y : \phi_1.[t/x]^\phi t'_1 = \lambda y : \phi_1.\hat{t}'_1$. It suffices to show that $\Gamma, \Gamma' \vdash \lambda y : \phi_1.\hat{t}'_1 : \phi_1 \to \phi_2$. By simply applying the λ-abstraction typing rule using $\Gamma, \Gamma', y : \phi_1 \vdash \hat{t} : \phi_2$ we obtain $\Gamma, \Gamma' \vdash \lambda y : \phi_1.\hat{t}'_1 : \phi_1 \to \phi_2$.

Case. Suppose $t' \equiv t'_1\ t'_2$. By inversion we know $\Gamma, x : \phi, \Gamma' \vdash t'_1 : \phi'' \to \phi'$ and $\Gamma, x : \phi, \Gamma' \vdash t'_2 : \phi''$ for some type ϕ''. Clearly, $t' > t'_i$ for $i \in \{1, 2\}$. Thus, by the induction hypothesis there exists terms m_1 and m_2 such that $[t/x]^\phi t'_i = m_i$, $\Gamma, \Gamma' \vdash m_1 : \phi'' \to \phi'$ and $\Gamma, \Gamma' \vdash m_2 : \phi''$ for $i \in \{1, 2\}$. We case split on whether or not m_1 is a λ-abstraction, t'_1 is not, or $ctype_\phi(x, t'_1)$ is undefined. We only consider the non-trivial cases when $m_1 \equiv \lambda y : \phi''.m'_1$ and t'_1 is not a λ-abstraction. Now by Lemma 63 it is the case that there exists a ψ such that $ctype_\phi(x, t'_1) = \psi$, $\psi \equiv \phi'' \to \phi'$, and ψ is a subexpression of ϕ, hence $\phi >_{\Gamma,\Gamma'} \phi''$. Then $[t/x]^\phi(t'_1\ t'_2) = [m_2/y]^{\phi''} m'_1$. Therefore, by the induction hypothesis there exists a term m such that $[m_2/y]^{\phi''} m'_1 = m$ and $\Gamma, \Gamma' \vdash m : \phi''$.

C.6 Proof of Redex Preservervation

This is a proof by induction on the lexicorgraphic combination (ϕ, t') of $>_{\Gamma,\Gamma'}$ and the strict subexpression ordering. We case split on the structure of t'. We do not explicitly state typind results that are simple conseqences of inversion.

Case. Let $t' \equiv x$ or $t' \equiv y$ where y is distinct from x. Trivial.

Case. Let $t' \equiv \lambda x : \phi_1.t''$. Then $[t/x]^{\phi}t' \equiv \lambda x : \phi_1.[t/x]^{\phi}t''$. Now

$$\begin{array}{lll} rset(\lambda x : \phi_1.t'', t) & = & rset(\lambda x : \phi_1.t'') \cup rset(t) \\ & = & rset(t'') \cup rset(t) \\ & = & rset(t'', t). \end{array}$$

We know that $t' > t''$ by the strict subexpression ordering, hence by the induction hypothesis $|rset(t'', t)| \geq |rset([t/x]^{\phi}t'')|$ which implies $|rset(t', t)| \geq |rset([t/x]^{\phi}t')|$.

Case. Let $t' \equiv t'_1\ t'_2$. First consider when t'_1 is not a λ-abstraction. Then

$$rset(t'_1\ t'_2, t) = rset(t'_1, t'_2, t)$$

Clearly, $t' > t'_i$ for $i \in \{1, 2\}$, hence, by the induction hypothesis $|rset(t'_i, t)| \geq |rset([t/x]^{\phi}t'_i)|$. We have two cases to consider. That is whether or not $[t/x]^{\phi}t'_1$ is a λ-abstraction or not. Suppose so. Then by Lemma 63 $ctype_{\phi}(x.t'_1) = \psi$ and by inversion on $\Gamma, x : \phi, \Gamma' \vdash t'_1\ t'_2 : \phi'$ there exists a type ϕ'' such that $\Gamma, x : \phi, \Gamma' \vdash t_1 : \phi'' \to \phi'$. Again, by Lemma 63 $\psi \equiv \phi'' \to \phi'$. Thus, $ctype_{\phi}(x, t'_1) = \phi'' \to \phi'$ and $\phi'' \to \phi'$ is a subexpression of ϕ. So by the definition of the hereditary substitution function $[t/x]^{\phi}t'_1\ t'_2 = [([t/x]^{\phi}t'_2)/y]^{\phi''}t''_1$, where $[t/x]^{\phi}t'_1 = \lambda y : \phi''.t''_1$. Hence,

$$|rset([t/x]^{\phi}t'_1\ t'_2)| = |rset([([t/x]^{\phi}t'_2)/y]^{\phi''}t''_1)|.$$

Now $\phi >_{\Gamma,\Gamma'} \phi''$ so by the induction hypothesis

$$\begin{array}{lll} |rset([([t/x]^{\phi}t'_2)/y]^{\phi''}t''_1)| & = & |rset([t/x]^{\phi}t'_2, t''_1)| \\ & = & |rset(t'_2, t''_1, t)| \\ & \geq & |rset(t'_2, [t/x]^{\phi}t'_1, t)| \\ & = & |rset(t'_2, t'_1, t)| \\ & = & |rset(t'_1, t'_2, t)|. \end{array}$$

Suppose $[t/x]^{\phi}t'_1$ is not a λ-abstractions or $ctype_{\phi}(x, t'_1)$ is undefined. Then

$$\begin{array}{lll} rset([t/x]^{\phi}(t'_1\ t'_2)) & = & rset([t/x]^{\phi}t'_1\ [t/x]^{\phi}t'_2) \\ & = & rset([t/x]^{\phi}t'_1, [t/x]^{\phi}t'_2). \\ & \leq & rset(t'_1, t'_2, t). \end{array}$$

Next suppose $t'_1 \equiv \lambda y : \phi_1.t''_1$. Then

$$rset((\lambda y : \phi_1.t''_1)\ t'_2, t) \quad = \quad \{(\lambda y : \phi_1.t''_1)\ t'_2\} \cup rset(t''_1, t'_2, t).$$

By the definition of the hereditary substitution function,

$$\begin{array}{lll} rset([t/x]^{\phi}(\lambda y : \phi_1.t''_1)\ t'_2) & = & rset([t/x]^{\phi}(\lambda y : \phi_1.t''_1)\ [t/x]^{\phi}t'_2) \\ & = & rset((\lambda y : \phi_1.[t/x]^{\phi}t''_1)\ [t/x]^{\phi}t'_2) \\ & = & \{(\lambda y : \phi_1.[t/x]^{\phi}t''_1)\ [t/x]^{\phi}t'_2\} \cup rset([t/x]^{\phi}t''_1) \cup rset([t/x]^{\phi}t'_2). \end{array}$$

Since $t' > t''_1$ and $t' > t'_2$ we can apply the induction hypothesis to obtain, $|rset(t''_1, t)| \geq |rset([t/x]^{\phi}t''_1)|$ and $|rset(t'_2, t)| \geq |rset([t/x]^{\phi}t'_2)|$. Therefore,
$|\{(\lambda y : \phi_1.t''_1)\ t'_2\} \cup rset(t''_1, t) \cup rset(t'_2, t)| \geq |\{(\lambda y : \phi_1.[t/x]^{\phi}t''_1)\ [t/x]^{\phi}t'_2\} \cup rset([t/x]^{\phi}t''_1) \cup rset([t/x]^{\phi}t'_2)|$.

C.7 Proof of Normality Preservation

By Lemma 64 we know there exists a term n'' such that $[n/x]^{\phi}n' = t$ and by Lemma 65 $|rset(n', n)| \geq |rset([n/x]^{\phi}n')|$. Hence, $|rset(n', n)| \geq |rset(t)|$, but $|rset(n', n)| = 0$. Therefore, $|rset(t)| = 0$ which implies n'' has no redexes. Therefore, n'' is normal.

C.8 Proof of Soundness with Respect to Reduction

This is a proof by induction on the lexicorgraphic combination (ϕ, t') of $>_{\Gamma}$ and the strict subexpression ordering. We case split on the structure of t'. When applying the induction hypothesis we must show that the input terms to the substitution and the hereditary substitution functions are typeable. We do not explicitly state typing results that are simple conseqences of inversion.

Case. Suppose t' is a variable x or y distinct from x. Trivial in both cases.

Case. Suppose $t' \equiv \lambda y : \phi'.\hat{t}$. Then $[t/x]^{\phi}(\lambda y : \phi'.\hat{t}) = \lambda y : \phi'.([t/x]^{\phi}\hat{t})$. Now $t' > \hat{t}$ so we can apply the induction hypothesis to obtain $[t/x]\hat{t} \rightsquigarrow^*_{\beta} [t/x]^{\phi}\hat{t}$. At this point we can see that since $\lambda y : \phi'.[t/x]\hat{t} \equiv [t/x](\lambda y : \phi'.\hat{t})$ and we may conclude that $\lambda y : \phi'.[t/x]\hat{t} \rightsquigarrow^*_{\beta} \lambda y : \phi'.[t/x]^{\phi}\hat{t}$.

Case. Suppose $t' \equiv t'_1\ t'_2$. By Lemma 64 there exists terms $\hat{t}'_1$ and $\hat{t}'_2$ such that $[t/x]^{\phi}t'_1 = \hat{t}'_1$ and $[t/x]^{\phi}t'_2 = \hat{t}'_2$. Since $t' > t'_1$ and $t' > t'_2$ we can apply the induction hypothesis to obtain $[t/x]t'_1 \rightsquigarrow^*_{\beta} \hat{t}'_1$ and $[t/x]t'_2 \rightsquigarrow^*_{\beta} \hat{t}'_2$. Now we case split on whether or not $\hat{t}'_1$ is a λ-abstraction and t'_1 is not, $ctype_{\phi}(x, t'_1)$ is undefined, or $\hat{t}'_1$ is not a λ-abstraction. If $ctype_{\phi}(x, t'_1)$ is undefined or $\hat{t}'_1$ is not a λ-abstraction then $[t/x]^{\phi}t' = ([t/x]^{\phi}t'_1)\ ([t/x]^{\phi}t'_2) \equiv \hat{t}'_1\ \hat{t}'_2$. Thus, $[t/x]t' \rightsquigarrow^*_{\beta} [t/x]^{\phi}t'$, because $[t/x]t' = ([t/x]t'_1)\ ([t/x]t'_2)$. So suppose $\hat{t}'_1 \equiv \lambda y : \phi'.\hat{t}''_1$ and t'_1 is not a λ-abstraction. By Lemma 63 there exists a type ψ such that $ctype_{\phi}(x, t'_1) = \psi$, $\psi \equiv \phi'' \to \phi'$, and ψ is a subexpression of ϕ, where by inversion on $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ there exists a type ϕ'' such that $\Gamma, x : \phi, \Gamma' \vdash t'_1 : \phi'' \to \phi'$. Then by the definiton of the hereditary substitution function $[t/x]^{\phi}(t'_1\ t'_2) = [\hat{t}'_2/y]^{\phi'}\hat{t}''_1$. Now we know $\phi >_{\Gamma,\Gamma'} \phi'$ so we can apply the induction hypothesis to obtain $[\hat{t}'_2/y]\hat{t}''_1 \rightsquigarrow^*_{\beta} [\hat{t}'_2/y]^{\phi'}\hat{t}''_1$. Now by knowing that $(\lambda y : \phi'.\hat{t}''_1)\ t'_2 \rightsquigarrow_{\beta} [\hat{t}'_2/y]\hat{t}''_1$ and by the previous fact we know $(\lambda y : \phi'.\hat{t}''_1)\ t'_2 \rightsquigarrow^*_{\beta} [\hat{t}'_2/y]^{\phi'}\hat{t}''_1$. We now make use of the well known result of full β-reduction. The result is stated as

$$\frac{\begin{array}{l} a \rightsquigarrow^*_{\beta} a' \\ b \rightsquigarrow^*_{\beta} b' \qquad a'\ b' \rightsquigarrow^*_{\beta} c \end{array}}{a\ b \rightsquigarrow^*_{\beta} c}$$

where a, a', b, b', and c are all terms. We apply this result by instantiating a, a', b, b', and c with $[t/x]t'_1$, $\hat{t}'_1$, $[t/x]t'_2$, $\hat{t}'_2$, and $[\hat{t}'_2/y]^{\phi'}\hat{t}''_1$ respectively. Therefore, $[t/x](t'_1\ t'_2) \rightsquigarrow^*_{\beta} [\hat{t}'_2/y]^{\phi'}\hat{t}''_1$.

C.9 Proof of Corollary 68

We have two cases to consider.

Case. Suppose $([t/x]^{\phi}t_1)$ is not a λ-abstraction or $([t/x]^{\phi}t_1)$ and t_1 are λ-abstractions, or $ctype_{\phi}(x, t_1)$ is undefined. Then $([t/x]^{\phi}t_1)\ ([t/x]^{\phi}t_2) = [t/x]^{\phi}(t_1\ t_2)$.

Case. Suppose $([t/x]^{\phi}t_1) \equiv \lambda y : \phi_1.s'_1$ for some y and s'_1, and $ctype_{\phi}(x, t_1) = \phi_1 \to \phi_2$. Then $([t/x]^{\phi}t_1)\ ([t/x]^{\phi}t_2) = (\lambda y : \phi_1.s'_1)\ ([t/x]^{\phi}t_2) \rightsquigarrow [([t/x]^{\phi}t_2)/y]s'_1$. By Lemma 67 $[([t/x]^{\phi}t_2)/y]s'_1 \rightsquigarrow^*_{\beta} [([t/x]^{\phi}t_2)/y]^{\phi_1}s'_1$, but in this case $[t/x]^{\phi}(t_1\ t_2) = [([t/x]^{\phi}t_2)/y]^{\phi_1}s'_1$. Therefore, $([t/x]^{\phi}t_1)\ ([t/x]^{\phi}t_2) \rightsquigarrow^*_{\beta} [t/x]^{\phi}(t_1\ t_2)$.

C.10 Proof of Semantic Equality

We proceed by induction on the form of n.

Case. Suppose $n \equiv x$. Then by the definition of the interpretation of types, there exists a type ϕ' and a proof p' such that $\Gamma \vdash p' : [\phi_1/X]\phi = \phi'$, $\Gamma \vdash x : \phi'$ and $\Gamma(x) = \phi'$. It suffices to show $\Gamma \vdash p' : [\phi_2/X]\phi = \phi'$. We know $\Gamma \vdash p' : [\phi_1/X]\phi = \phi'$ and $\Gamma \vdash p : \phi_1 = \phi_2$. Thus, by applying Conv, $\Gamma \vdash p' : [\phi_2/X]\phi = \phi'$. Therefore, $n \in [\![[\phi_2/X]\phi]\!]_{\Gamma}$.

Case. Let $n \equiv \lambda x.t$. By the definition of the interpretation of types, $\Gamma \vdash \lambda x.t : [\phi_1/X]\phi$ and there exists a types ϕ'_1 and ϕ'_2 and a proof p' such that $\Gamma \vdash p' : [\phi_1/X]\phi = \phi'_1 \to \phi'_2$ and if $\Gamma, x : \phi'_1$ is consistent then $t \in [\![\phi'_2]\!]_{\Gamma, x:\phi'_1}$. It suffices to show that $\Gamma \vdash \lambda x.t : [\phi_2/X]\phi$ and $\Gamma \vdash p' : [\phi_2/X]\phi = \phi'_1 \to \phi'_2$. Both of these can be obtained by applying Conv using the assumed proof p. Therefore, $n \in [\![[\phi_2/X]\phi]\!]_{\Gamma}$.

Case. Let $n \equiv join$. Then by the definition of the interpretation of types, $\Gamma \vdash join : [\phi_1/X]\phi$ and there exists a type ϕ' and ϕ'' and a proof p', such that $\Gamma \vdash p' : [\phi_1/X]\phi = (\phi' = \phi'')$ and for all substitutions $\sigma : \Gamma$ we have $\sigma\ \phi' \equiv \sigma\ \phi''$. By applying Conv to $\Gamma \vdash join : [\phi_1/X]\phi$ we obtain $\Gamma \vdash join : [\phi_2/X]\phi$ and by applying it again using $\Gamma \vdash p' : [\phi_1/X]\phi = (\phi' = \phi'')$ we obtain $\Gamma \vdash p' : [\phi_2/X]\phi = (\phi' = \phi'')$. It suffices to show that there exists a type $\hat{\phi}$ and $\hat{\phi}'$ and a proof $\hat{p}$, such that $\Gamma \vdash \hat{p} : [\phi_2/X]\phi = (\hat{\phi} = \hat{\phi}')$ and for all substitutions $\sigma : \Gamma$ we have $\sigma\ \hat{\phi} \equiv \sigma\ \hat{\phi}'$. Choose ϕ', ϕ'', and p' for $\hat{\phi}$, $\hat{\phi}'$, $\hat{p}$ respectively.

Case. Let $n \equiv n_1\ n_2$. By the definition of the interpretation of types, $\Gamma \vdash n_1\ n_2 : [\phi_1/X]\phi$ and there exists a type ϕ' such that $n_1 \in [\![\phi' \to [\phi_1/X]\phi]\!]_\Gamma$ and $n_2 \in [\![\phi']\!]_\Gamma$. By applying Conv we obtain $\Gamma \vdash n_1\ n_2 : [\phi_2/X]\phi$. It suffices to show that $n_1 \in [\![\phi' \to [\phi_2/X]\phi]\!]_\Gamma$. We can see that $[\![\phi' \to [\phi_1/X]\phi]\!]_\Gamma \equiv [\![[\phi_1/X](\phi' \to \phi)]\!]_\Gamma$. By the induction hypothesis, $n_1 \in [\![[\phi_2/X](\phi' \to \phi)]\!]_\Gamma \equiv [\![\phi' \to [\phi_2/X]\phi]\!]_\Gamma$. Thus, $n \in [\![[\phi_2/X]\phi]\!]_\Gamma$.

C.11 Proof of Weakening for the Interpretation of Types

Suppose Γ' is a context which does not overlap with Γ. We need to consider the cases when Γ, Γ' is consistent and when it is not consistent. Suppose it is. Then the result follows by straightforward induction on the form of v and weakening for typing.

Suppose Γ, Γ' is inconsistent. Then either Γ is inconsistent or Γ' is inconsistent. Suppose the former. Then the result easily follows by induction on the form of v and weakening for typing. Finally, suppose Γ is consistent and Γ' is inconsistent. By assumption we know $v \in [\![\phi]\!]_\Gamma$. We prove this case by induction on the form of v.

Case. Suppose v is a variable. Trivial.

Case. Suppose $v \equiv join$. Then by the definition of the interpretation of types we know $\Gamma \vdash join : \phi$ and there exists types ϕ', ϕ'', and a term p such that $\Gamma \vdash p : \phi = (\phi' = \phi'')$ and $\forall\sigma : \Gamma.\sigma\phi' \equiv \sigma\phi''$. It suffices to show that $\Gamma, \Gamma' \vdash join : \phi$ and there exists types ψ', ψ'', and a term p' such that $\Gamma, \Gamma' \vdash p' : \phi = (\psi' = \psi'')$ and $\forall\sigma' : \Gamma, \Gamma'.\sigma'\psi' \equiv \sigma'\psi''$. The first two results hold by weakening for typing on the previous facts. The third result holds by first choosing ϕ' and ϕ'' for ψ' and ψ'' and then taking the substitution σ above and then combining it with the identity substitution $\sigma'' : \Gamma'$. That is choose $\sigma' = \sigma \cup \sigma'$. We choose the ideenity because we know Γ and Γ' do not overlap so ϕ' and ϕ'' cannot possibly depend on variables in Γ'. Therefore, $v \in [\![\phi]\!]_{\Gamma,\Gamma'}$.

Case. Suppose $v \equiv \lambda x.t$. Now by assumption Γ is consistent and Γ' is inconsistent hence Γ, Γ' is inconsistent. Thus, all we must conclude to obtain our result is $\Gamma, \Gamma' \vdash \lambda x.t : \phi \wedge \exists(\phi_1, \phi_2, p).(\Gamma, \Gamma' \vdash p : \phi = \phi_1 \to \phi_2)$. These follow from our assumption that $v \in [\![\phi]\!]_\Gamma$ and weakening for typing.

Case. Suppose $v \equiv s\ v'$. The results holds by using the induction hypothesis, the assumption that $v \in [\![\phi]\!]_\Gamma$, and weakening for typing.

C.12 Proof of Hereditary Substitution for the Interpretation of Types

This is a proof by induction on our usual ordering (ϕ, v').

Case. Suppose v' is a variable. Trivial.

Case. Suppose $v' \equiv s\ v''$. By assumption $s\ v'' \in [\![\phi']\!]_{\Gamma,x:\phi,\Gamma'}$ hence by the definition of the interpretation of types $s \in [\![\phi'' \to \phi']\!]_{\Gamma,x:\phi,\Gamma'}$ and $v'' \in [\![\phi'']\!]_{\Gamma,x:\phi,\Gamma'}$ for some type ϕ''. By the induction hypothesis we know $[v/x]^\phi s \in [\![\phi'' \to \phi']\!]_{\Gamma,\Gamma'}$ and $[v/x]^\phi v'' \in [\![\phi'']\!]_{\Gamma,\Gamma'}$. We have two cases to consider.

Case. Suppose $([v/x]^\phi s)$ is not a λ-abstraction or $([v/x]^\phi s)$ and s are λ-abstractions, or $ctype_\phi(x, s)$ is undefined. Then $([v/x]^\phi s)\ ([v/x]^\phi v'')$ is a value and $([v/x]^\phi s)\ ([v/x]^\phi v'') \in [\![\phi']\!]_{\Gamma,\Gamma'}$.

Case. Suppose $([v/x]^{\phi} s) \equiv \lambda y : \phi''.s'$ for some y and s', and $ctype_{\phi}(x, s) = \phi'' \to \phi'$. Then $([v/x]^{\phi} s)\ ([v/x]^{\phi} v'') \rightsquigarrow_{\beta} [([v/x]^{\phi} v'')/y]s'$. By Lemma 67 $[([v/x]^{\phi} v'')/y]s' \rightsquigarrow^{*}_{\beta} [([v/x]^{\phi} v'')/y]^{\phi''} s'$ and by Lemma 63 $\phi'' \to \phi'$ is a subexpression of ϕ hence $\phi >_{\Gamma,\Gamma'} \phi''$. Thus, by the induction hypothesis $[([v/x]^{\phi} v'')/y]^{\phi''} s' \in [\![\phi']\!]_{\Gamma,\Gamma'}$.

Case. Suppose $v' \equiv \lambda x.t'$. By assumption $\lambda y.t' \in [\![\phi']\!]_{\Gamma,x:\phi,\Gamma'}$. From this we know several facts, first that $\Gamma, x : \phi, \Gamma' \vdash \lambda y.t' : \phi'$, and second that there exists a term p and types ϕ_1 and ϕ_2 such that $\Gamma, x : \phi, \Gamma' \vdash p : \phi' = \phi_1 \to \phi_2$ and if $Con(\Gamma, x : \phi, \Gamma')$ holds then $t' \in [\![\phi_1]\!]_{\Gamma,x:\phi,\Gamma,y:\phi_1}$. Regardless of the consistency of $\Gamma, x : \phi, \Gamma'$ we must first show that $\Gamma, \Gamma' \vdash [t/x]^{\phi} v' : \phi'$ and there exists a proof and types p', ϕ'_1 and ϕ'_2 such that $\Gamma, \Gamma' \vdash p' : \phi' = \phi'_1 \to \phi'_2$. Both of these results are obtainable from Lemma 63. To obtain the second we can choose the term p and types ϕ_1 and ϕ_2 from above and apply the hereditary substitution function to p to obtain $\Gamma, \Gamma' \vdash [t/x]^{\phi} p : \phi' = \phi_1 \to \phi_2$. To conclude this case have two final cases to consider, whether or not $\Gamma, x : \phi, \Gamma'$ is consistent.

Clearly, if $\Gamma, x : \phi, \Gamma'$ is inconsistent then we know $[t/x]^{\phi}(\lambda y.t') = \lambda y.[t/x]^{\phi} t' \in [\![\phi']\!]_{\Gamma,\Gamma'}$, becuase $\lambda y.[t/x]^{\phi} t'$ is a value, $\Gamma, \Gamma' \vdash [t/x]^{\phi} v' : \phi'$, and $\Gamma, \Gamma' \vdash [t/x]^{\phi} p : \phi' = \phi_1 \to \phi_2$.

It suffices to show that $[t/x]^{\phi} t' \in [\![\phi_2]\!]_{\Gamma,\Gamma',y:\phi_1}$ when $\Gamma, x : \phi, \Gamma'$ is consistent. Suppose $\Gamma, x : \phi, \Gamma'$ is consistent. Then by the definition of the interpretation of types we know $t' \in [\![\phi_2]\!]_{\Gamma,x:\phi,\Gamma',y:\phi_1}$. Since t' is a strict subexpression of v' we can apply the induction hypothesis to obtain $[t/x]^{\phi} t' \in [\![\phi_2]\!]_{\Gamma,\Gamma',y:\phi_1}$.

Finally, all cases lead to the conclusion that $[t/x]^{\phi} v' \in [\![\phi']\!]_{\Gamma,\Gamma'}$.

C.13 Proof of Type Soundness

This is a proof by induction on the form of the assumed typing derivation. We implicitly use the fact that for each case $\Gamma \vdash t : \phi$ implies there exists a proof p such that $\Gamma \vdash p : \phi = \phi$ throughout the entire proof.

Case.

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

By assumption we know $\Gamma \vdash x : \phi$ and $\Gamma(x) = \phi$, thus by the definition of the interpretation of types, $x \in [\![\phi]\!]_{\Gamma}$.

Case.

$$\frac{\Gamma \vdash \phi_1 \qquad \Gamma, x : \phi_1 \vdash t : \phi_2}{\Gamma \vdash \lambda x.t : \phi_1 \to \phi_2}$$

We have two cases to consider, when $\Gamma, x : \phi_1$ is consistent and when it is not. Consider the latter. Then by the definition of the interpretation of types, $\lambda x.t \in [\![\phi_1 \to \phi_2]\!]_{\Gamma}$. Now assume $\Gamma, x : \phi_1$ is consistent. Then by the definition of the interpretation of types and the induction hypothesis, $t \in [\![\phi_2]\!]_{\Gamma,x:\phi_1}$. Finally, by the definition of the interpretation of types, $\lambda x.t \in [\![\phi_1 \to \phi_2]\!]_{\Gamma}$.

Case.

$$\frac{\Gamma \vdash t_1 : \phi_1 \to \phi_2 \qquad \Gamma \vdash t_2 : \phi_1}{\Gamma \vdash t_1\ t_2 : \phi_2}$$

By the induction hypothesis, $t_1 [\![\phi_1 \to \phi_2]\!]_\Gamma$ and $t_1 [\![\phi_1]\!]_\Gamma$. Thus, there exists values v_1 and v_2 such that $t_1 \rightsquigarrow_\beta^* v_1 \in [\![\phi_1 \to \phi_2]\!]_\Gamma$ and $t_2 \rightsquigarrow_\beta^* v_2 \in [\![\phi_1]\!]_\Gamma$. Choose a variable x fresh in both t_1 and t_2. Then we know that by the definition of the interpretation of types $(x\ v_2) \in [\![\phi_2]\!]_{\Gamma, x:\phi_1 \to \phi_2}$. Now we know the following:

$$\begin{array}{lll} [t_1/x](x\ t_2) & \rightsquigarrow_\beta^* & [v_1/x](x\ v_2) \\ & \rightsquigarrow_\beta^* & [v_1/x]^{\phi_1 \to \phi_2}(x\ v_2). \end{array}$$

The previous fact holds by β-reduction and Lemma 67. Finally, by Lemma 32 $[v_1/x]^{\phi_1 \to \phi_2}(x\ v_2) \in [\![\phi_2]\!]_\Gamma$. Therefore, $t_1\ t_2 \in [\![\phi_2]\!]_\Gamma$.

Case.

$$\frac{}{\Gamma \vdash join : \phi = \phi}$$

Trivial.

Case.

$$\frac{\Gamma \vdash t : [\phi_1/X]\phi \qquad \Gamma \vdash t' : \phi_1 = \phi_2}{\Gamma \vdash t : [\phi_2/X]\phi}$$

By the induction hypothesis, $t \in [\![[\phi_1/X]\phi]\!]_\Gamma$. We know by assumption that $\Gamma \vdash t' : \phi_1 = \phi_2$, thus by Lemma 30, $t \in [\![[\phi_2/X]\phi]\!]_\Gamma$.