Exploring the Reach of Hereditary Substitution

Harley Eades and Aaron Stump

Computer Science
The University of Iowa

UPenn PL Club 2012

Introduction

- Tait-Girard's reducibility is the most often used proof technique for proving normalization.
  - Complex.
    - Type soundness theorem requires universal quantification over all substitutions.
    - Requires mutual recursion.
- Hereditary substitution shows promise of being less complex than reducibility.
  - No universal quantification needed in the statement of the type soundness theorem.
  - In general not dependent on mutual recursion.
  - One major draw back: we are unsure what systems hereditary substitution can be applied to.
    - This is the focus of our work.

Introduction

- Type Theories:
  - Stratified System F^{+}
  - Dependent Stratified System $F^{=}$
  - Simply Typed λ-Calculus$^{=}$, and
  - $\lambda\Delta$-Calculus.
- The hereditary substitution function.
  - Well-founded ordering on types.
  - Properties of the hereditary substitution function.
- Concluding normalization.
  - The interpretation of types.
  - Hereditary Substitution for the interpretation of types.
  - Type soundness.

Stratified System F^+ (SSF^+)

- SSF^+ is an extension of the system Stratified System F first analyzed by D. Leivant and N. Danner.
- Syntax for kinds, types, and terms:

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

Stratified System F^+ (SSF^+)

- Kind assignment rules:

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

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

Stratified System F^+ (SSF^+)

- The type assignment rules:

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

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

$$\frac{\Gamma \vdash t : \phi_1 \quad \Gamma \vdash \phi_2 : *_p}{\Gamma \vdash inl(t) : \phi_1 + \phi_2} \qquad \frac{\Gamma \vdash t : \phi_2 \quad \Gamma \vdash \phi_1 : *_p}{\Gamma \vdash inr(t) : \phi_1 + \phi_2} \qquad \frac{\Gamma \vdash t : \phi_1 + \phi_2 \quad \Gamma, x : \phi_1 \vdash t_1 : \psi \quad \Gamma, x : \phi_2 \vdash t_2 : \psi}{\Gamma \vdash \text{case } t \text{ of } x.t_1, x.t_2 : \psi}$$

Stratified System F^+ (SSF^+)

- The reduction rules:

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

- Commuting Conversions:

$$
\begin{array}{l}
(\text{case } t \text{ of } x.t_1, x.t_2)\ t' \\
\quad \rightsquigarrow \text{case } t \text{ of } x.(t_1\ t'), x.(t_2\ t')
\end{array}
$$

$$
\begin{array}{l}
\text{case } (\text{case } t \text{ of } x.t_1, x.t_2) \text{ of } y.s_1, y.s_2 \\
\quad \rightsquigarrow \text{case } t\ \text{ of } x.(\text{case } t_1 \text{ of } y.s_1, y.s_2), \\
\qquad\qquad\qquad\quad x.(\text{case } t_1 \text{ of } y.s_1, y.s_2)
\end{array}
$$

Stratified System F^+ (SSF^+)

- The reduction rules:

Redex		
$(\Lambda X : *_p.t)[\phi]$	$\rightsquigarrow$	$[\phi/X]t$
$(\lambda x : \phi.t)t'$	$\rightsquigarrow$	$[t'/x]t$
case $inl(t)$ of $x.t_1,x.t_2$	$\rightsquigarrow$	$[t/x]t_1$
case $inr(t)$ of $x.t_1,x.t_2$	$\rightsquigarrow$	$[t/x]t_2$

- Commuting Conversions:

Structural redex

(case t of $x.t_1,x.t_2$) t'
$\rightsquigarrow$ case t of $x.(t_1\ t'),x.(t_2\ t')$

case (case t of $x.t_1,x.t_2$) of $y.s_1,y.s_2$
$\rightsquigarrow$ case t of $x.$(case t_1 of $y.s_1,y.s_2$),
$x.$(case t_1 of $y.s_1,y.s_2$)

Well-founded ordering on types

Definition (well-founded ordering on types)

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

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

Theorem ($>_\Gamma$ is well-founded)

The ordering $>_\Gamma$ is well-founded on types ϕ such that $\Gamma \vdash \phi : *_l$ for some l.

Hereditary Substitution

- Syntax: $[t/x]^A t' = t''$.
- Usual termination order: (A, t').

- Like ordinary capture avoiding substitution.
- Except, if the substitution introduces a redex, then that redex is recursively reduced.
  - Example: $[\lambda z : \mathrm{b}.z/x]^{\mathrm{b}\to\mathrm{b}}(x\,y)(\approx ((\lambda z : \mathrm{b}.z)\,y \approx [y/z]^{\mathrm{b}}z) = y$.

- The constructive content of normalization proofs dating all the way back to Prawitz (1965).
- First made explicit by K. Watkins for simple types and R. Adams for dependent types.

The hereditary substitution function for SSF$^+$

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

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

$ctype_\phi(x, t[\phi']) = [\phi'/X]\phi''$
Where $ctype_\phi(x, t) = \forall X : *_l.\phi''$.

Lemma (Properties of $ctype_\phi$)

If $\Gamma, x : \phi, \Gamma' \vdash t : \phi'$ and $ctype_\phi(x, t) = \phi''$ then $head(t) = x$, $\phi' \equiv \phi''$, and $\phi' \leq_\Gamma \phi$.

The hereditary substitution function for SSF$^+$

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

$app_\phi\ (\lambda x : \phi'.t_1)\ t_2 = [t_2/x]^{\phi'}\ t_1$
$app_\phi\ (\text{case } t_0 \text{ of } x.t_1, x.t_2)\ t = \text{case } t_0 \text{ of } x.(app_\phi\ t_1\ t), x.(app_\phi\ t_2\ t)$

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

$rcase_\phi\ inl(t')\ y\ t_1\ t_2 = [t'/y]^{\phi_1}\ t_1$
$rcase_\phi\ inr(t')\ y\ t_1\ t_2 = [t'/y]^{\phi_2}\ t_2$

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

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

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

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 $([t/x]^{\phi} t_1) \equiv \lambda y : \phi''.s'_1$ 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 $[t/x]^{\phi} t_1 \equiv \text{case } w \text{ of } y.r, y.s$ 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 $[t/x]^{\phi} t' \equiv \Lambda X : *_l.s'_1$, for some X, s'_1 and $\Gamma \vdash \phi' : *_q$, such that, $q \leq l$ and $ctype_{\phi}(x, t') = \forall X : *_l.\phi''$.

$[t/x]^{\phi}(\text{case } t_0 \text{ of } y.t_1, y.t_2) = \text{case } ([t/x]^{\phi} t_0) \text{ of } y.([t/x]^{\phi} t_1), y.([t/x]^{\phi} t_2)$

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 $ctype_{\phi}(x, t_0) = \phi_1 + \phi_2$.

The ctype$_\phi$ *properties*

Lemma (Properties of ctype$_\phi$*)*

i. *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* $ctype_\phi(x, t_1) = \psi$.

ii. *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* $ctype_\phi(x, t_1) = \psi$.

iii. *If* $\Gamma, x : \phi, \Gamma' \vdash t'[\phi''] : \phi'$, $\Gamma \vdash t : \phi$, $[t/x]^\phi t' = \Lambda X : *_l.t''$, *and* t' *is not then there exists a type* ψ *such that* $ctype_\phi(x, t') = \psi$.

iv. *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* $ctype_\phi(x, t_0) = \psi$.

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

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

Properties of the hereditary substitution function

Lemma (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 (Redex Preserving)

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

Examples: rset and commuting conversions

- Structural redexes are not preserved by the hereditary substitution function in general.

  Let

  $t \equiv inl(a)$, such that $a : \phi_1 \vdash t : \phi_1 + \phi_2$ and
  $t' \equiv \text{case (case } x \text{ of } z.z,z.z) \text{ of } y.y,y.y$.

  So

  $[t/x]^{\phi_1+\phi_2} t' =$
  $\text{case } ([t/x]^{\phi_1+\phi_2}(\text{case } x \text{ of } z.z,z.z)) \text{ of } y.([t/x]^{\phi_1+\phi_2} y),y.([t/x]^{\phi_1+\phi_2} y)$.

  Now

  $[t/x]^{\phi_1+\phi_2}(\text{case } x \text{ of } z.z,z.z) =$
  $\text{rcase}_{\phi_1+\phi_2}\ [t/x]^{\phi_1+\phi_2} x\ [t/x]^{\phi_1+\phi_2} z\ [t/x]^{\phi_1+\phi_2} z$,

  because

  $[t/x]^{\phi_1+\phi_2} x = inl(a)$, x is not an inject-left term, and
  $ctype_{\phi_1+\phi_2}(x, x) = \phi_1 + \phi_2$.

  Finally,

  $[t/x]^{\phi_1+\phi_2}(\text{case } x \text{ of } z.z,z.z) = [a/z]^{\phi_1} z = a$, which implies,
  $[t/x]^{\phi_1+\phi_2} t' = \text{case } a \text{ of } y.y,y.y$.

Properties of the hereditary substitution function

Lemma (Normality Preserving)

If $\Gamma \vdash n : \phi$ and $\Gamma, x : \phi' \vdash n' : \phi'$ then there exists a normal term n'' such that $[n/x]^{\phi} n' = n''$.

Lemma (Soundness with Respect to Reduction)

If $\Gamma \vdash t : \phi$ and $\Gamma, x : \phi, \Gamma' \vdash t' : \phi'$ then $[t/x]t' \rightsquigarrow^* [t/x]^{\phi} t'$.

Concluding normalization

Definition

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

Lemma (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 Totality 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 Normality Preservation $\hat{n}$ is normal. Therefore, $[n/x]^\phi n' = \hat{n} \in [\![\phi']\!]_{\Gamma, \Gamma'}$. $\square$

Concluding normalization

Theorem (Type Soundness)

If $\Gamma \vdash t : \phi$ then $t \in [\![\phi]\!]_\Gamma$.

Corollary (Normalization)

If $\Gamma \vdash t : \phi$ then $t \rightsquigarrow^! n$.

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

- $\text{DSSF}^=$ is another extension of the system Stratified System F.
- Syntax for kinds, types, and terms:

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

- Reduction Rules:

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

Dependent Stratified System $F^{=}$ ($DSSF^{=}$)

- Kind assignment rules:

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

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

- Type assignment rules:

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

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

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

Syntactic Inversion for DSSF$^=$

- For the proof of normalization for DSSF$^=$ our semantics are the same as for SSF$^+$.
- Must have semantic inversion for the proof of normalization.
  - I.e. $\lambda x : \phi_1.t \in [\![\Pi x : \phi_1.\phi_2]\!]_\Gamma \implies t \in [\![\phi_2]\!]_{\Gamma, x:\phi_1}$ and $\Gamma \vdash p : t_1 = t_2 \implies [\![[t_1/x]\phi]\!]_\Gamma = [\![[t_2/x]\phi]\!]_\Gamma$.
- Because, our semantics are sets of typeable normal forms.
  - Syntactic inversion $\implies$ Semantic inversion.
- Syntactic inversion turns out to be non-trivial.

Syntactic Inversion for DSSF$^{=}$

- Type syntactic equality:

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

Lemma (Type Syntactic Conversion)

If $\Gamma \vdash t : \phi$ *and* $\Gamma \vdash \phi \approx \phi'$ *then* $\Gamma \vdash t : \phi'$.

Lemma (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* $\Gamma, y : \phi_1 \vdash \phi_2 \approx \phi'_2$.

Syntactic Inversion for DSSF$^{=}$

Lemma (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 $\exists (t_1, t_2, \phi')$.
$t_1 \downarrow t_2 \wedge \Gamma \vdash t_1 : \phi' \wedge \Gamma \vdash t_2 : \phi' \wedge \Gamma \vdash \phi \approx t_1 = t_2 \wedge \Gamma\ Ok$.

Well-founded ordering on types for DSSF$^{=}$

Definition

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

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

Theorem (Well-Founded Ordering)

The ordering $>_\Gamma$ is well-founded on types ϕ such that $\Gamma \vdash \phi : *_l$ for some l.

Lemma

If $\Gamma \vdash \phi' \approx \phi''$ and $\phi >_\Gamma \phi'$ then $\phi >_\Gamma \phi''$.

The hereditary substitution function for DSSF$^{=}$

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

$[t/x]^{\phi} y = y$
  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, $q \leq l$ and t' is not a type abstraction.

Concluding normalization

- The interpretation of types are the same as for SSF$^+$.

Lemma (Semantic Equality)

If $\Gamma \vdash p : t_1 = t_2$ *then* $[\![[t_1/x]\phi]\!]_\Gamma = [\![[t_2/x]\phi]\!]_\Gamma$.

Lemma (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 (Type Soundness)

If $\Gamma \vdash t : \phi$ *then* $t \in [\![\phi]\!]_\Gamma$.

Corollary (Normalization)

If $\Gamma \vdash t : \phi$ *then* $t \leadsto^! n$.

The simply typed λ-Calculus$^{=}$ (STLC$^{=}$)

- An extension of STLC with a primitive notion of equality between types.
- Syntax:

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

- Reduction:

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

The simply typed λ-Calculus$^=$ (STLC$^=$)

- Kinding rules:

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

- Typing rules:

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

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

Why CBV and not full β-reduction?

Example

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 .

The hereditary substitution function

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

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

Where y is a variable distinct from x.

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

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

$[t/x]^{\phi}(t_1\ t_2) = ([t/x]^{\phi} t_1)\ ([t/x]^{\phi} t_2)$

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 $ctype_{\phi}(x, t_1) = \phi_1 \to \phi_2$.

The interpretation of types

Definition

The interpretation of types $[\![\phi]\!]$ is defined as follows:

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

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

Concluding normalization

Lemma (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 (Hereditary Substitution for the Interpretation of Types)

If $v' \in [\![\phi']\!]_{\Gamma,x:\phi,\Gamma'}$ *and* $v \in [\![\phi]\!]_\Gamma$ *then* $[v/x]^\phi v' \in [\![\phi']\!]_{\Gamma,\Gamma'}$.

Theorem (Type Soundness)

If $\Gamma \vdash t : \phi$ *then* $t \in [\![\phi]\!]_\Gamma$.

Corollary (Normalization)

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

The $\lambda\Delta$-Calculus

- A type theory correspoding to classical natural deduction.
- Originally defined by J. Rehof and M. Sørensen in 1994.
- Provably equalivalent to M. Parigot's $\lambda\mu$-Calculus.
- The bases of classical pure type systems (G. Barthe, J. Hatcliff, M. Sørensen 1997).

The $\lambda\Delta$-Calculus

- Syntax:

$$\begin{array}{rcl} T, A, B, C & ::= & \perp \mid b \mid A \to B \\ t & ::= & x \mid \lambda x : T.t \mid \Delta x : T.t \mid t_1\, t_2 \\ n, m & ::= & \lambda x : T.n \mid \Delta x : T.n \mid h \\ h & ::= & x \mid h\, n \end{array}$$

We denote the set of all terms $\mathcal{T}$ and the set of all types Ψ.

- Reduction:

$$\frac{}{(\lambda x : T.t)\, t' \rightsquigarrow [t'/x]t}\ \textsc{Beta}$$

$$\frac{\begin{array}{c} y \text{ fresh in } t \text{ and } t' \\ z \text{ fresh in } t \text{ and } t' \end{array}}{(\Delta x : \neg(T_1 \to T_2).t)\, t' \rightsquigarrow \Delta y : \neg T_2.[\lambda z : T_1 \to T_2.(y\,(z\, t'))/x]t}\ \textsc{StructRed}$$

The $\lambda\Delta$-Calculus

- Typing Rules:

$$\frac{}{\Gamma, x : A \vdash x : A}\ \textsc{Ax} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A.t : A \to B}\ \textsc{Lam}$$

$$\frac{\begin{array}{c}\Gamma \vdash t_2 : A \\ \Gamma \vdash t_1 : A \to B\end{array}}{\Gamma \vdash t_1\ t_2 : B}\ \textsc{App} \qquad \frac{\Gamma, x : \neg A \vdash t : \perp}{\Gamma \vdash \Delta x : \neg A.t : A}\ \textsc{Delta}$$

An Intuition of the Problems Involved

- Recall how hereditary substitution works for β-reduction:
  $[\lambda z : \mathsf{b}.z/x]^{\mathsf{b}\to\mathsf{b}}(x\,y)(\approx ((\lambda z : \mathsf{b}.z)\,y \approx [y/z]^{\mathsf{b}} z) = y$

An Intuition of the Problems Involved

- Recall how hereditary substitution works for β-reduction:
  $[\lambda z : \mathsf{b}.z/x]^{\mathsf{b} \to \mathsf{b}}(x\, y)(\approx ((\lambda z : \mathsf{b}.z)\, y \approx [y/z]^{\mathsf{b}} z) = y$

- The naive solution for structural reduction:
  $[\Delta x : \neg(A'' \to A').(x\, q)/z]^{(A'' \to A')}(z\, r) = \Delta y : \neg A'.[(\lambda u : A'' \to A'.(y\, (u\, r)))/x]^{\neg(A'' \to A')}(x\, q)$

An Intuition of the Problems Involved

- Recall how hereditary substitution works for β-reduction:
  $[\lambda z : \mathsf{b}.z/x]^{\mathsf{b}\to\mathsf{b}}(x\,y)(\approx ((\lambda z : \mathsf{b}.z)\,y \approx [y/z]^{\mathsf{b}} z) = y$
- The naive solution for structural reduction:
  $[\Delta x : \neg(A'' \to A').(x\,q)/z]^{(A''\to A')}(z\,r) = \Delta y : \neg A'.[(\lambda u : A'' \to A'.(y\,(u\,r)))/x]^{\neg(A''\to A')}(x\,q)$
  - The cut type actually increased!
- The problem: The usual termination order (A, t') no longer works.
  - How do we fix this?

A Look at Structural Reduction

Consider: $((\Delta x : \neg(A'' \to A').t)\, t') \rightsquigarrow \Delta y : \neg A'.[(\lambda u : A'' \to A'.(y\,(u\, t')))/x]t$

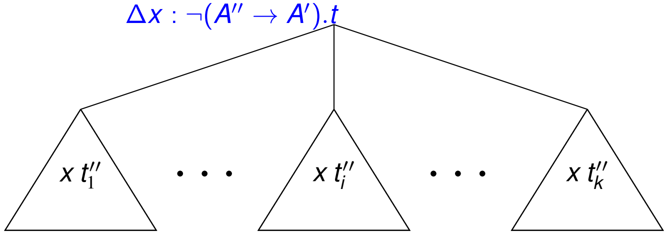
A Look at Structural Reduction

Consider: $((\Delta x : \neg(A'' \to A').t)\, t') \rightsquigarrow \Delta y : \neg A'.[(\lambda u : A'' \to A'.(y\,(u\,t')))/x]t$

A Look at Structural Reduction

Consider: $((\Delta x : \neg(A'' \to A').t)\, t') \rightsquigarrow \Delta y : \neg A'.[(\lambda u : A'' \to A'.(y\,(u\,t')))/x]t$

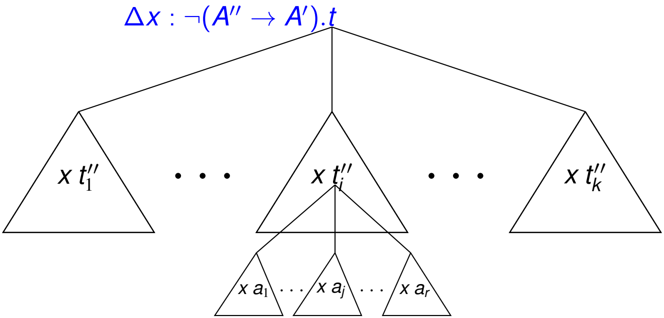
When redexes are created:



A Look at Structural Reduction

Consider: $((\Delta x : \neg(A'' \to A').t)\, t') \rightsquigarrow \Delta y : \neg A'.[(\lambda u : A'' \to A'.(y\,(u\,t')))/x]t$

When redexes are created:



Is Further Reduction the Answer?

- Consider the previous example:
  $[\Delta x : \neg(A'' \to A').(x\, q)/z]^{(A'' \to A')}(z\, r) = \Delta y : \neg A'.[(\lambda u : A'' \to A'.(y\,(u\, r)))/x]^{\neg(A'' \to A')}(x\, q)$

- Recursively reducing the redexes introduced by substituting the linear λ-abstraction:
  $[\Delta x : \neg(A'' \to A').(x\, q)/z]^{(A'' \to A')}(z\, r) = \Delta y : \neg A'.(y\, [q/u]^{(A'' \to A')}(u\, r))$

Is Further Reduction the Answer?

- Consider the previous example:
  $[\Delta x : \neg(A'' \to A').(x\, q)/z]^{(A'' \to A')}(z\, r) = \Delta y : \neg A'.[(\lambda u : A'' \to A'.(y\,(u\, r)))/x]^{\neg(A'' \to A')}(x\, q)$
- Recursively reducing the redexes introduced by substituting the linear λ-abstraction:
  $[\Delta x : \neg(A'' \to A').(x\, q)/z]^{(A'' \to A')}(z\, r) = \Delta y : \neg A'.(y\,[q/u]^{(A'' \to A')}(u\, r))$
  - The cut type stayed the same.
  - But the term we are substituting has decreased.
  - Is this always the case? Basically, it is!

The Final Solution

- The term we are substituting either decrease structurally or decreases contextually.
  - Structural decrease: $\forall t, t'.t < t'$ if t' is a strict subexpression of t.
  - Contextual decrease: A term is considered larger than itself with a hole.
    - $\forall \mathcal{C}, t.C < t$ if $\exists s.\mathcal{C}[s] \equiv t$.

- Using this insight the hereditary substitution function is defineable using the ordering (A, t, t').

Hereditary Substitution

Type: $\mathcal{T} \cup \mathcal{E} \to \mathcal{T} \to \Psi \to \mathcal{T} \cup \mathcal{E} \to \mathcal{T} \cup \mathcal{E}$
Total using the ordering: (A, t, t')

$[t/x]^A \Box = \Box$

$[t/x]^A x = t$

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

$[t/x]^A(\lambda y : A'.t') = \lambda y : A'.([t/x]^A t')$
Where $\mathsf{FV}(t) \cap \mathsf{FV}(t') = \emptyset$.

$[t/x]^A(\Delta y : A'.t') = \Delta y : A'.([t/x]^A t')$
Where $\mathsf{FV}(t) \cap \mathsf{FV}(t') = \emptyset$.

$[t/x]^A(t_1\ t_2) = ([t/x]^A t_1)\,([t/x]^A t_2)$
Where $([t/x]^A t_1)$ is not a λ-abstraction or Δ-abstraction, or both $([t/x]^A t_1)$ and t_1 are λ-abstractions or Δ-abstractions, or $\mathsf{ctype}_A(x, t_1)$ is undefined.

$[t/x]^A(t_1\ t_2) = [s_2'/y]^{A''} s_1'$
Where $([t/x]^A t_1) = \lambda y : A''.s_1'$ for some y, s_1' and A'', $[t/x]^A t_2 = s_2'$, and $\mathsf{ctype}_A(x, t_1) = A'' \to A'$.

$[t/x]^A(t_1\ t_2) = \Delta z : \neg A'.([\lambda u : A'' \to A'.(z\,(u\,s_2))/y]s_1)$
Where $([t/x]^A t_1) = \Delta y : \neg(A'' \to A').s_1$ for some y, s_1, A'', and there does not exists any context of s_1 equal to $\mathcal{C}[y\ s_1']$ for some term s_1', $([t/x]^A t_2) = s_2$ for some s_2, z and u are fresh variables of type A' and $A'' \to A'$ respectively, and $\mathsf{ctype}_A(x, t_1) = A'' \to A'$.

$[t/x]^A(t_1\ t_2) = \Delta z : \neg A'.[\lambda u : A'' \to A'.(z\,(u\,s_2))/y](\mathsf{fill}\,\mathcal{C}[\overrightarrow{\Box_i}]\ \overrightarrow{\mathcal{C}[z\,([s_1/q]^{A'' \to A'}(q\,s_2))]})$
Where $([t/x]^A t_1) = \Delta y : \neg(A'' \to A').\mathcal{C}[\overrightarrow{(y\,s_1)_i}]$ for some i, y, s_1 and A'', $([t/x]^A t_2) = s_2$ for some s_2, z and r are fresh variables of type A' and A'' respectively, and $\mathsf{ctype}_A(x, t_1) = A'' \to A'$.

Type: $\mathcal{T} \cup \mathcal{E} \to \mathcal{T} \to \Psi \to \mathcal{T} \cup \mathcal{E} \to \mathcal{T} \cup \mathcal{E}$
Total using the ordering: (A, t, t')

$[t/x]^A \Box = \Box$

$[t/x]^A x = t$

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

$[t/x]^A (\lambda y : A'.t') = \lambda y : A'.([t/x]^A t')$
Where $\mathsf{FV}(t) \cap \mathsf{FV}(t') = \emptyset$.

$[t/x]^A (\Delta y : A'.t') = \Delta y : A'.([t/x]^A t')$
Where $\mathsf{FV}(t) \cap \mathsf{FV}(t') = \emptyset$.

$[t/x]^A (t_1\ t_2) = ([t/x]^A t_1)\ ([t/x]^A t_2)$
Where $([t/x]^A t_1)$ is not a λ-abstraction or Δ-abstraction, or both $([t/x]^A t_1)$ and t_1 are λ-abstractions or Δ-abstractions, or $\mathsf{ctype}_A(x, t_1)$ is undefined.

$[t/x]^A (t_1\ t_2) = [s_2'/y]^{A''} s_1'$
Where $([t/x]^A t_1) = \lambda y : A''.s_1'$ for some y, s_1' and A'', $[t/x]^A t_2 = s_2'$, and $\mathsf{ctype}_A(x, t_1) = A'' \to A'$.

$[t/x]^A (t_1\ t_2) = \Delta z : \neg A'.([\lambda u : A'' \to A'.(z\,(u\,s_2))/y] s_1)$
Where $([t/x]^A t_1) = \Delta y : \neg(A'' \to A').s_1$ for some y, s_1, A'', and there does not exists any context of s_1 equal to $\mathcal{C}[y\ s_1']$ for some term s_1', $([t/x]^A t_2) = s_2$ for some s_2, z and u are fresh variables of type A' and $A'' \to A'$ respectively, and $\mathsf{ctype}_A(x, t_1) = A'' \to A'$.

$[t/x]^A (t_1\ t_2) = \Delta z : \neg A'.[\lambda u : A'' \to A'.(z\,(u\,s_2))/y](\mathsf{fill}\,\mathcal{C}[\overrightarrow{\Box_i}]\ \overrightarrow{\mathcal{C}[z\,([s_1/q]^{A'' \to A'}(q\,s_2))]})$
Where $([t/x]^A t_1) = \Delta y : \neg(A'' \to A').\mathcal{C}[\overrightarrow{(y\,s_1)_i}]$ for some i, y, s_1 and A'', $([t/x]^A t_2) = s_2$ for some s_2, z and r are fresh variables of type A' and A'' respectively, and $\mathsf{ctype}_A(x, t_1) = A'' \to A'$.

Hereditary Substitution

Type: $\mathcal{T} \cup \mathcal{E} \to \mathcal{T} \to \Psi \to \mathcal{T} \cup \mathcal{E} \to \mathcal{T} \cup \mathcal{E}$
Total using the ordering: (A, t, t')

$[t/x]^A\Box = \Box$

$[t/x]^A x = t$

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

$[t/x]^A(\lambda y : A'.t') = \lambda y : A'.([t/x]^A t')$
Where $\mathsf{FV}(t) \cap \mathsf{FV}(t') = \emptyset$.

$[t/x]^A(\Delta y : A'.t') = \Delta y : A'.([t/x]^A t')$
Where $\mathsf{FV}(t) \cap \mathsf{FV}(t') = \emptyset$.

$[t/x]^A(t_1\ t_2) = ([t/x]^A t_1)\,([t/x]^A t_2)$
Where $([t/x]^A t_1)$ is not a λ-abstraction or Δ-abstraction, or both $([t/x]^A t_1)$ and t_1 are λ-abstractions or Δ-abstractions, or $\mathsf{ctype}_A(x, t_1)$ is undefined.

$[t/x]^A(t_1\ t_2) = [s_2'/y]^{A''} s_1'$
Where $([t/x]^A t_1) = \lambda y : A''.s_1'$ for some y, s_1' and A'', $[t/x]^A t_2 = s_2'$, and $\mathsf{ctype}_A(x, t_1) = A'' \to A'$.

$[t/x]^A(t_1\ t_2) = \Delta z : \neg A'.([\lambda u : A'' \to A'.(z\,(u\,s_2))/y]s_1)$
Where $([t/x]^A t_1) = \Delta y : \neg(A'' \to A').s_1$ for some y, s_1, A'', and there does not exists any context of s_1 equal to $\mathcal{C}[y\ s_1']$ for some term s_1', $([t/x]^A t_2) = s_2$ for some s_2, z and u are fresh variables of type A' and $A'' \to A'$ respectively, and $\mathsf{ctype}_A(x, t_1) = A'' \to A'$.

$[t/x]^A(t_1\ t_2) = \Delta z : \neg A'.[\lambda u : A'' \to A'.(z\,(u\,s_2))/y](\mathsf{fill}\,\mathcal{C}[\overrightarrow{\Box_i}]\,\overrightarrow{\mathcal{C}[z\,([s_1/q]^{A''\to A'}(q\,s_2))]})$
Where $([t/x]^A t_1) = \Delta y : \neg(A'' \to A').\mathcal{C}[\overrightarrow{(y\,s_1)_i}]$ for some i, y, s_1 and A'', $([t/x]^A t_2) = s_2$ for some s_2, z and r are fresh variables of type A' and A'' respectively, and $\mathsf{ctype}_A(x, t_1) = A'' \to A'$.

Hereditary Substitution: Handling Structural Reduction

- Case when no further redexes are created:

$[t/x]^A (t_1\ t_2) = \Delta z : \neg A'.([\lambda u : A'' \to A'.(y\,(u\,s_2))/y]s_1)$

Where $([t/x]^A t_1) = \Delta y : \neg(A'' \to A').s_1$ for some y, s_1, A'', and there does not exists any context of s_1 equal to $\mathcal{C}[y\ s_1']$ for some term s_1' , $([t/x]^A t_2) = s_2$ for some s_2, z and u are fresh variables of type A' and $A'' \to A'$ respectively, and $\mathrm{ctype}_A(x, t_1) = A'' \to A'$.

Hereditary Substitution: Handling Structural Reduction

- Case when no further redexes are created:

  $[t/x]^A(t_1\ t_2) = \Delta z : \neg A'.([\lambda u : A'' \to A'.(y\,(u\,s_2))/y]s_1)$

  Where $([t/x]^A t_1) = \Delta y : \neg(A'' \to A').s_1$ for some y, s_1, A'', and there does not exists any context of s_1 equal to $\mathcal{C}[y\ s'_1]$ for some term s'_1 , $([t/x]^A t_2) = s_2$ for some s_2, z and u are fresh variables of type A' and $A'' \to A'$ respectively, and $\mathrm{ctype}_A(x, t_1) = A'' \to A'$.

- Case when structural reduction will introduce more redexes:

  $[t/x]^A(t_1\ t_2) = \Delta z : \neg A'.[\lambda u : A'' \to A'.(z\,(u\,s_2))/y](\mathrm{fill}\,\mathcal{C}[\overrightarrow{\square_i}]\ \overrightarrow{\mathcal{C}[z\,([s_1/q]^{A'' \to A'}(q\,s_2))]})$

  Where $([t/x]^A t_1) = \Delta y : \neg(A'' \to A').\mathcal{C}[\overrightarrow{(y\,s_1)_i}]$ for some i, y, s_1 and A'', $([t/x]^A t_2) = s_2$ for some s_2, z and r are fresh variables of type A' and A'' respectively, and $\mathrm{ctype}_A(x, t_1) = A'' \to A'$.
  - Do not substitute the linear lambda-abstractions, but reduce them right away.
  - $\overrightarrow{\mathcal{C}[t]}$: Expands the context into a list of lists of subcontexts.
  - If $A \equiv A'' \to A'$ then we know $t_1 \equiv x$ and $t \equiv \Delta y : \neg(A'' \to A').\mathcal{C}[\overrightarrow{(y\,s_1)_i}]$.
    - Hence $s_1 < t$.

Concluding Normalization

Definition

The interpretation of types $[\![T]\!]_\Gamma$ is defined by:

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

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

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

Lemma (Hereditary Substitution for the Interpretation of Types)

If $n \in [\![T]\!]_\Gamma$ and $n' \in [\![T']\!]_{\Gamma,x:T,\Gamma'}$, then $[n/x]^T n' \in [\![T']\!]_{\Gamma,\Gamma'}$.

Theorem (Type Soundness)

If $\Gamma \vdash t : T$ then $t \in [\![T]\!]_\Gamma$.

Concluding remarks

- We have analyzed several systems.
  - Simply Typed λ-Calculus$^=$
    - An extension of STLC with a primitive notion of equality between types.
  - Stratified System F (SSF)
  - Stratified System F$^+$
    - An extension of SSF with sum types and commuting conversions.
  - Dependent Stratified System F
    - An extension of SSF with dependent function types and a primitive notion of equality between terms.
  - Stratified System Fω
    - An extension of SSF with type-level computation.
  - The $\lambda\Delta$-Calculus.
    - An extension of STLC with the law of double negation.
- Future work.
  - Can we do this for system T and system F?
- Thank you all for listening.