Exploring the Reach of Hereditary Substitution

Harley Eades and Aaron Stump

Computer Science
The University of Iowa

TYPES 2011

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

- Stratified System F^{+}.
- The hereditary substitution function.
  - Well-founded ordering on types.
  - Properties of the hereditary substitution function.
- Concluding normalization.
  - The interpretation of types.
  - 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:

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

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:

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

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

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

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:

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

$$
\begin{array}{l}
(\text{case } t \text{ of } x.t_1, x.t_2)[\phi] \\
\quad \rightsquigarrow (\text{case } t \text{ of } x.(t_1[\phi]), x.(t_2[\phi])
\end{array}
$$

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

Stratified System F^+ (SSF^+)

- The reduction rules:

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

- Commuting Conversions:

Structural redex

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

(case t of $x.t_1$,$x.t_2$)$[\phi]$
$\leadsto$ (case t of $x.(t_1[\phi])$,$x.(t_2[\phi])$)

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

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

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 function [Watkins et al., 2004]

- Syntax: $[t/x]^{\phi}t' = t''$.
- Like ordinary capture avoiding substitution.
- Except, if the substitution introduces a redex, then that redex is recursively reduced.
  - Example: $[(\lambda z : b.z)/x]^{b \to b}(x\ y)$ $(\rightsquigarrow (\lambda z : b.z)y \rightsquigarrow [y/z]^{b}z) = y$.

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 $ctype_{\phi}(x, t_1) = \phi'' \rightarrow \phi'$.

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

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 $ctype_{\phi}(x, t_1) = \phi'' \rightarrow \phi'$.

$[t/x]^{\phi}(t'[\phi']) = ([t/x]^{\phi} t')[\phi']$

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 $ctype_{\phi}(x, t') = \forall X : *_l.\phi''$.

$[t/x]^{\phi}(\text{case } t_0 \text{ of } y.t_1, y.t_2) =$
case $([t/x]^{\phi} t_0)$ 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
$t' \equiv \text{case } (\text{case } x \text{ of } z.z,z.z) \text{ of } y.y,y.y$.

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* $[t/x]t' \leadsto^* [t/x]^{\phi} t'$.

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, $[n/x]^\phi n' = \hat{n} \in [\![\phi']\!]_{\Gamma,\Gamma'}$. □

Concluding normalization

Theorem (Type Soundness)

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

Corollary (Normalization)

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

Concluding remarks

- We have analyzed several systems.
  - Simply Typed λ-Calculus (STLC)
  - 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.
- Future work.
  - Extend to higher ordinals. Goal: System T.
- Thank you all of you for listening.