Hereditary Substitution for Classical Natural Deduction

Harley Eades

Trellys Meeting 01/2012

The University of Iowa

What is this talk about?

- Goals: future and achieved
- Introduction to the $\lambda\mu$-Calculus
- Introduction to hereditary substitution
- Conclude normalization

Goals of this Project

- Future:
  - Explicitly state the hereditary substitution function for the $\lambda\mu$-Calculus.
  - Use the hereditary substitution function to prove WN.
- Achieved:
  - Proven WN of the $\lambda\mu$-Calculus using implicit hereditary substitution.

The $\lambda\mu$-Calculus

- Originally defined by M. Parigot in 1992.
  - Corresponds to Free Deduction.
  - Two types of variables.
  - Types were full sequents.
  - Multiple Conclusions. Even no conclusion!

The $\lambda\mu$-Calculus

- Originally defined by M. Parigot in 1992.
  - Corresponds to Free Deduction.
  - Two types of variables.
  - Types were full sequents.
  - Multiple Conclusions. Even no conclusion!
- A few equivalent systems.
  - Easy: restrict Parigot's version to exactly one conclusion.
  - Easy: restrict Parigot's version to formulas for types.
  - Not so easy: restrict Parigot's version to one type of variables.

The $\lambda\mu$-Calculus

- Originally defined by M. Parigot in 1992.
  - Corresponds to Free Deduction.
  - Two types of variables.
  - Types were full sequents.
  - Multiple Conclusions. Even no conclusion!
- A few equivalent systems.
  - Easy: restrict Parigot's version to exactly one conclusion.
  - Easy: restrict Parigot's version to formulas for types.
  - Not so easy: restrict Parigot's version to one type of variables.
- Can we have our cake and eat it too?

The $\lambda\Delta$-Calculus.

The $\lambda\mu$-Calculus

- The $\lambda\mu$-Calculus.
  - Defined in J. Rehof's Ph.D. thesis in 1994.

The $\lambda\mu$-Calculus

Definition (Syntax)

$$
\begin{array}{lll}
\textit{term},\ t & ::= & \\
& | & x \\
& | & \lambda x.t \\
& | & \mu x.t \\
& | & t_1\ t_2
\end{array}
\qquad
\begin{array}{lll}
\textit{type},\ T,\ A,\ B,\ C & ::= & \\
& | & \perp \\
& | & \mathrm{b} \\
& | & A \to B
\end{array}
$$

- One additional definition: $\neg A =^{def} A \to \perp$.

The $\lambda\mu$-Calculus

Definition (Typing)

$$\boxed{\Gamma \vdash t : T}$$

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

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

The $\lambda\mu$-Calculus

Definition (Operational Semantics)

$\boxed{t \rightsquigarrow t'}$

$$\frac{}{(\lambda x : T.t)\ t' \rightsquigarrow [t'/x]t}\ \textsc{Beta} \qquad \frac{\begin{array}{c} y \text{ fresh in } t \text{ and } t' \\ z \text{ fresh in } t \text{ and } t' \end{array}}{(\mu x.t)\ t' \rightsquigarrow \mu y.[\lambda z : T.(y\ (z\ t'))/x]t}\ \textsc{StructRed}$$

- To justify the StructRed rule consider its typing:

$$\frac{\Gamma \vdash t' : A \qquad \Gamma \vdash \mu x.t : A \to B}{\Gamma \vdash (\mu x.t)\, t' : B}\ \textsc{App} \qquad\qquad \frac{\Gamma, x : \neg(A \to B) \vdash t : \perp}{\Gamma \vdash \mu x.t : A \to B}\ \textsc{Mu}$$

$$(\mu x.t)\ t' \quad \rightsquigarrow \quad \mu y.[\ \lambda z : A \to B.(y\,(z\,t'))/x]t$$

- To justify the StructRed rule consider its typing:

$$\frac{\Gamma \vdash t' : A \qquad \Gamma \vdash \mu x.t : A \to B}{\Gamma \vdash (\mu x.t)\, t' : B}\ \textsc{App} \qquad \frac{\Gamma, x : \neg(A \to B) \vdash t : \perp}{\Gamma \vdash \mu x.t : A \to B}\ \textsc{Mu}$$

$$(\mu x.t)\ t' \quad \rightsquigarrow \quad \mu y.[\ \lambda z : A \to B.(y\,(z\,t'))/x]t$$

The $\lambda\mu$-Calculus: Typing of the Structural Redex

- To justify the StructRed rule consider its typing:

$$\frac{\Gamma \vdash t' : A \qquad \Gamma \vdash \mu x.t : A \to B}{\Gamma \vdash (\mu x.t)\, t' : B}\ \textsc{App} \qquad\qquad \frac{\Gamma, x : \neg(A \to B) \vdash t : \perp}{\Gamma \vdash \mu x.t : A \to B}\ \textsc{Mu}$$

$$(\mu x.t)\, t' \quad\rightsquigarrow\quad \mu y.[\ \lambda z : A \to B.(y\,(z\, t'))/x]t$$

- To justify the StructRed rule consider its typing:

$$\frac{\Gamma \vdash t' : A \quad \Gamma \vdash \mu x.t : A \to B}{\Gamma \vdash (\mu x.t)\, t' : B}\ \textsc{App} \qquad \frac{\Gamma, x : \neg(A \to B) \vdash t : \perp}{\Gamma \vdash \mu x.t : A \to B}\ \textsc{Mu}$$

$$(\mu x.t)\, t' \quad \rightsquigarrow \quad \mu y.[\ \lambda z : A \to B.(y\,(z\,t'))/x]t$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, y : \neg B, z : A \to B \vdash z : A \to B}\ \textsc{Ax} \quad \Gamma, y : \neg B, z : A \to B \vdash t' : A}{\Gamma, y : \neg B, z : A \to B \vdash (z\,t') : B}\ \textsc{App} \quad \dfrac{}{\Gamma, y : \neg B, z : A \to B \vdash y : \neg B}\ \textsc{Ax}}{\Gamma, y : \neg B, z : A \to B \vdash y\,(z\,t') : \perp}\ \textsc{App}}{\Gamma, y : \neg B \vdash \lambda z : A \to B.(y\,(z\,t')) : \neg(A \to B)}\ \textsc{Lam}$$

The $\lambda\mu$-Calculus: The μ-Abstraction as a Control Operator

- The μ-abstraction really is a control operator.
  - We know from Felleisen's thesis that the control operator $\mathcal{F}$ has the following operational semantics:

$$\begin{array}{llll}
(1) & \mathcal{F}(t) & \rightsquigarrow & t(\lambda x.x) \\
(2) & \mathcal{F}(t)\,t' & \rightsquigarrow & \mathcal{F}(\lambda x.(t(\lambda y.(x(y\,t'))))) \\
(3) & t\,\mathcal{F}(t') & \rightsquigarrow & \mathcal{F}(\lambda x.(t'(\lambda y.(x\,(t\,y))))), \\
 & & & \text{where } t \text{ is a value.}
\end{array}$$

  - $\mathcal{F}(\lambda x.t)$ has the same computational behavior as $\mu x.t$.

The $\lambda\mu$-Calculus: The μ-Abstraction as a Control Operator

- In fact we can embed the $\lambda\mu$-calculus in Felleisen's calculus.

Definition (Embedding)

$$\begin{array}{lcl} |\mu x.t| & = & \mathcal{F}(\lambda x.|t|) \\ |\lambda x : A.t| & = & \lambda x.|t| \\ |t_1\ t_2| & = & |t_1|\ |t_2| \end{array}$$

The $\lambda\mu$-Calculus: The μ-Abstraction as a Control Operator

- In fact we can embed the $\lambda\mu$-calculus in Felleisen's calculus.

Definition (Embedding)

$$\begin{array}{lcl} |\mu x.t| & = & \mathcal{F}(\lambda x.|t|) \\ |\lambda x : A.t| & = & \lambda x.|t| \\ |t_1\ t_2| & = & |t_1|\,|t_2| \end{array}$$

- Once we have the embedding we have the following equivalence:

$$\begin{array}{lcl} |(\mu x.t)\ t'| & = & \mathcal{F}(\lambda x.|t|)\,|t'| \\ & \rightsquigarrow_2 & \mathcal{F}(\lambda y.((\lambda x.|t|)\,(\lambda z.(y\,(z\,|t'|))))) \\ & \rightsquigarrow_\beta & \mathcal{F}(\lambda y.([(\lambda z.(y\,(z\,|t'|)))/x]|t|)) \\ & = & |\mu y.([(\lambda z : A \to B.(y\,(z\,t')))/x]t)| \end{array}$$

The $\lambda\mu$-Calculus: The μ-Abstraction as a Control Operator

- In fact we can embed the $\lambda\mu$-calculus in Felleisen's calculus.

Definition (Embedding)

$$\begin{array}{lcl} |\mu x.t| & = & \mathcal{F}(\lambda x.|t|) \\ |\lambda x : A.t| & = & \lambda x.|t| \\ |t_1\ t_2| & = & |t_1|\ |t_2| \end{array}$$

- Once we have the embedding we have the following equivalence:

$$\begin{array}{lcl} |(\mu x.t)\ t'| & = & \mathcal{F}(\lambda x.|t|)\ |t'| \\ & \rightsquigarrow_2 & \mathcal{F}(\lambda y.((\lambda x.|t|)\ (\lambda z.(y\ (z\ |t'|))))) \\ & \rightsquigarrow_\beta & \mathcal{F}(\lambda y.([(\lambda z.(y\ (z\ |t'|)))/x]|t|)) \\ & = & |\mu y.([(\lambda z : A \to B.(y\ (z\ t')))/x]t)| \end{array}$$

- Finally, we have:

$$\begin{array}{lcl} |\lambda x.(\mu k.k\ (x\ (\lambda y.(\mu d.x\ y))))| & = & \lambda x.\mathcal{F}(\lambda k.k\ (x\ (\lambda y.\mathcal{F}(\lambda d.x\ y)))) \\ & =^{def} & call/cc \end{array}$$

History up to Hereditary Substitution

- Hereditary substitution lies at the heart of combinatorial proofs of normalization of logics and type theories.
  - Intuitionistic natural deduction by Prawitz:1965.
  - WN of the Simply Typed λ-Calculus (STLC) by Girard et al.:1989.
  - Joachimski and Matthes:1999 show WN and SN of STLC.
  - These are just a few examples.
- The hereditary substitution function was first made explicit by Watkins:2004 and Adams:2004.

Hereditary Substitution

- It has the following syntax: $[t/x]^T t' = t''$.
- Main idea:
  - Like ordinary capture avoiding substitution.
  - Except, if the substitution introduces a redex, then that redex is recursively reduced.

Hereditary Substitution

- It has the following syntax: $[t/x]^T t' = t''$.
- Main idea:
  - Like ordinary capture avoiding substitution.
  - Except, if the substitution introduces a redex, then that redex is recursively reduced.
- Main properties:
  Assuming $\Gamma \vdash t : T \wedge \Gamma, x : T \vdash t' : T'$.

Hereditary Substitution

- It has the following syntax: $[t/x]^T t' = t''$.
- Main idea:
  - Like ordinary capture avoiding substitution.
  - Except, if the substitution introduces a redex, then that redex is recursively reduced.
- Main properties:
  Assuming $\Gamma \vdash t : T \land \Gamma, x : T \vdash t' : T'$.
  - Total: $\exists t''.[t/x]^T t' = t''$

Hereditary Substitution

- It has the following syntax: $[t/x]^T t' = t''$.
- Main idea:
  - Like ordinary capture avoiding substitution.
  - Except, if the substitution introduces a redex, then that redex is recursively reduced.
- Main properties:
  Assuming $\Gamma \vdash t : T \wedge \Gamma, x : T \vdash t' : T'$.
  - Total: $\exists t''.[t/x]^T t' = t''$
  - Type preserving: $\Gamma \vdash [t/x]^T t' : T'$.

Hereditary Substitution

- It has the following syntax: $[t/x]^T t' = t''$.
- Main idea:
  - Like ordinary capture avoiding substitution.
  - Except, if the substitution introduces a redex, then that redex is recursively reduced.
- Main properties:
  Assuming $\Gamma \vdash t : T \land \Gamma, x : T \vdash t' : T'$.
  - Total: $\exists t''.[t/x]^T t' = t''$
  - Type preserving: $\Gamma \vdash [t/x]^T t' : T'$.
  - Normality preserving: $[n/x]^T n' = n''$.

Hereditary Substitution

- It has the following syntax: $[t/x]^T t' = t''$.
- Main idea:
  - Like ordinary capture avoiding substitution.
  - Except, if the substitution introduces a redex, then that redex is recursively reduced.
- Main properties:
  Assuming $\Gamma \vdash t : T \wedge \Gamma, x : T \vdash t' : T'$.
  - Total: $\exists t''.[t/x]^T t' = t''$
  - Type preserving: $\Gamma \vdash [t/x]^T t' : T'$.
  - Normality preserving: $[n/x]^T n' = n''$.
  - Consistent w.r.t. reduction: $[t/x]t' \rightsquigarrow^* [t/x]^T t'$.

Hereditary Substitution

- It has the following syntax: $[t/x]^T t' = t''$.
- Main idea:
  - Like ordinary capture avoiding substitution.
  - Except, if the substitution introduces a redex, then that redex is recursively reduced.
- Main properties:
  Assuming $\Gamma \vdash t : T \wedge \Gamma, x : T \vdash t' : T'$.
  - Total: $\exists t''.[t/x]^T t' = t''$
  - Type preserving: $\Gamma \vdash [t/x]^T t' : T'$.
  - Normality preserving: $[n/x]^T n' = n''$.
  - Consistent w.r.t. reduction: $[t/x]t' \rightsquigarrow^* [t/x]^T t'$.
- Each of the above properties can be proven with a simple lexicographic combination: (T, t').

Concluding Normalization

Definition (Ordering)

$$\begin{array}{lcl} T_1 \to T_2 & >_\Gamma & T_1 \\ T_1 \to T_2 & >_\Gamma & T_2 \end{array}$$

- Well-founded ✓

Concluding Normalization

Definition ($ctype_T$)

$ctype_T(x, x) = T$

$ctype_T(x, t_1\ t_2) = T''$
  Where $ctype_T(x, t_1) = T' \to T''$.

- $ctype_T$ has nice properties:
  - If $ctype_T(x, t) = T'$ then $head(t) = x$ and T' is a subexpression of T.
  - Furthermore, t has type T'.
  - If $t' \equiv t_1\ t_2$, $[t/x]t_1 \rightsquigarrow^* \lambda y : T_1.t_1'$ and t_1 is not, then $ctype_T(x, t_1)$ is defined.
  - If $t' \equiv t_1\ t_2$, $[t/x]t_1 \rightsquigarrow^* \mu y.t_1'$ and t_1 is not, then $ctype_T(x, t_1)$ is defined.

Concluding Normalization

Definition (Interpretation of Types)

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

Extended to non-normal terms:

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

Concluding Normalization

Lemma (Substitution for the Interpretation of Types)

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

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

- Lemma statement: Carefully crafted.
- Proof: Mutual induction using the ordering (T, n').
- Proof: First part only requires second part of the ordering.

A Peek at the Proof

Case. Suppose $n' \equiv h\, n''$.

1. IH(2): $[n/x]h \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$

2. IH(2): $[n/x]n'' \in [\![T'']\!]_{\Gamma,\Gamma'}$

3. $[n/x]h \rightsquigarrow^! m_1 \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$

4. $[n/x]n'' \rightsquigarrow^! m_2 \in [\![T'']\!]_{\Gamma,\Gamma'}$

TS. $m_1\, m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$

5. Suppose $m_1 \not\equiv \lambda y : T''.m'$ and $m_1 \not\equiv \mu y.m'$.

Case. Suppose $n' \equiv h\, n''$.

1. IH(2): $[n/x]h \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$
2. IH(2): $[n/x]n'' \in [\![T'']\!]_{\Gamma,\Gamma'}$
3. $[n/x]h \rightsquigarrow^! m_1 \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$
4. $[n/x]n'' \rightsquigarrow^! m_2 \in [\![T'']\!]_{\Gamma,\Gamma'}$

TS. $m_1\, m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$

5. Suppose $m_1 \not\equiv \lambda y : T''.m'$ and $m_1 \not\equiv \mu y.m'$.

A Peek at the Proof

Case. Suppose $n' \equiv h\, n''$.

1. IH(2): $[n/x]h \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$ ✓
2. IH(2): $[n/x]n'' \in [\![T'']\!]_{\Gamma,\Gamma'}$
3. $[n/x]h \rightsquigarrow^! m_1 \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$
4. $[n/x]n'' \rightsquigarrow^! m_2 \in [\![T'']\!]_{\Gamma,\Gamma'}$

TS. $m_1\, m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$

5. Suppose $m_1 \not\equiv \lambda y : T''.m'$ and $m_1 \not\equiv \mu y.m'$.

A Peek at the Proof

Case. Suppose $n' \equiv h\, n''$.

1. IH(2): $[n/x]h \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$ ✓

2. IH(2): $[n/x]n'' \in [\![T'']\!]_{\Gamma,\Gamma'}$ ✓

3. $[n/x]h \rightsquigarrow^! m_1 \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$

4. $[n/x]n'' \rightsquigarrow^! m_2 \in [\![T'']\!]_{\Gamma,\Gamma'}$

TS. $m_1\, m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$

5. Suppose $m_1 \not\equiv \lambda y : T''.m'$ and $m_1 \not\equiv \mu y.m'$.

A Peek at the Proof

Case. Suppose $n' \equiv h\, n''$.

1. IH(2): $[n/x]h \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$ ✓
2. IH(2): $[n/x]n'' \in [\![T'']\!]_{\Gamma,\Gamma'}$ ✓
3. $[n/x]h \rightsquigarrow^! m_1 \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$ ✓
4. $[n/x]n'' \rightsquigarrow^! m_2 \in [\![T'']\!]_{\Gamma,\Gamma'}$

TS. $m_1\, m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$

5. Suppose $m_1 \not\equiv \lambda y : T''.m'$ and $m_1 \not\equiv \mu y.m'$.

A Peek at the Proof

Case. Suppose $n' \equiv h\, n''$.

1. IH(2): $[n/x]h \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$ ✓

2. IH(2): $[n/x]n'' \in [\![T'']\!]_{\Gamma,\Gamma'}$ ✓

3. $[n/x]h \leadsto^! m_1 \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$ ✓

4. $[n/x]n'' \leadsto^! m_2 \in [\![T'']\!]_{\Gamma,\Gamma'}$ ✓

TS. $m_1\, m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$

5. Suppose $m_1 \not\equiv \lambda y : T''.m'$ and $m_1 \not\equiv \mu y.m'$.

A Peek at the Proof

Case. Suppose $n' \equiv h\,n''$.

1. IH(2): $[n/x]h \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$ ✓
2. IH(2): $[n/x]n'' \in [\![T'']\!]_{\Gamma,\Gamma'}$ ✓
3. $[n/x]h \rightsquigarrow^! m_1 \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$ ✓
4. $[n/x]n'' \rightsquigarrow^! m_2 \in [\![T'']\!]_{\Gamma,\Gamma'}$ ✓

TS. $m_1\ m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$

5. Suppose $m_1 \not\equiv \lambda y : T''.m'$ and $m_1 \not\equiv \mu y.m'$.

A Peek at the Proof

Case. Suppose $n' \equiv h\, n''$.

1. IH(2): $[n/x]h \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$ ✓
2. IH(2): $[n/x]n'' \in [\![T'']\!]_{\Gamma,\Gamma'}$ ✓
3. $[n/x]h \rightsquigarrow^{!} m_1 \in [\![T'' \to T']\!]_{\Gamma,\Gamma'}$ ✓
4. $[n/x]n'' \rightsquigarrow^{!} m_2 \in [\![T'']\!]_{\Gamma,\Gamma'}$ ✓

TS. $m_1\, m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$

5. Suppose $m_1 \not\equiv \lambda y : T''.m'$ and $m_1 \not\equiv \mu y.m'$. ✓

6. Suppose $m_1 \equiv \lambda y : T''.m'$.

   6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$

   6.2. $[n/x]n' \rightsquigarrow^* m_1\ m_2 \equiv (\lambda y : T''.m')\ m_2 \rightsquigarrow [m_2/y]m'$

   TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

   6.3. h is not a λ-abstraction and m_1 is a λ-abstraction

   6.4. $ctype_T$ Props: $ctype_T(x, h) = A$

   6.5. $ctype_T$ Props: $A \equiv T'' \rightarrow T'$

   6.6. $ctype_T$ Props: A is a subexpression of T

   6.8. $(T, m_1) > (T'', [m_2/y]m')$

   6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

6. Suppose $m_1 \equiv \lambda y : T''.m'$.
   - 6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$
   - 6.2. $[n/x]n' \rightsquigarrow^* m_1\, m_2 \equiv (\lambda y : T''.m')\, m_2 \rightsquigarrow [m_2/y]m'$
   - TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 6.3. h is not a λ-abstraction and m_1 is a λ-abstraction
   - 6.4. $ctype_T$ Props: $ctype_T(x, h) = A$
   - 6.5. $ctype_T$ Props: $A \equiv T'' \rightarrow T'$
   - 6.6. $ctype_T$ Props: A is a subexpression of T
   - 6.8. $(T, m_1) > (T'', [m_2/y]m')$
   - 6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

6. Suppose $m_1 \equiv \lambda y : T''.m'$.
   - 6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$ ✓
   - 6.2. $[n/x]n' \rightsquigarrow^* m_1\ m_2 \equiv (\lambda y : T''.m')\ m_2 \rightsquigarrow [m_2/y]m'$
   - TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 6.3. h is not a λ-abstraction and m_1 is a λ-abstraction
   - 6.4. $ctype_T$ Props: $ctype_T(x, h) = A$
   - 6.5. $ctype_T$ Props: $A \equiv T'' \to T'$
   - 6.6. $ctype_T$ Props: A is a subexpression of T
   - 6.8. $(T, m_1) > (T'', [m_2/y]m')$
   - 6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

6. Suppose $m_1 \equiv \lambda y : T''.m'$.
   - 6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$ ✓
   - 6.2. $[n/x]n' \rightsquigarrow^* m_1\ m_2 \equiv (\lambda y : T''.m')\ m_2 \rightsquigarrow [m_2/y]m'$ ✓
   - TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 6.3. h is not a λ-abstraction and m_1 is a λ-abstraction
   - 6.4. $ctype_T$ Props: $ctype_T(x, h) = A$
   - 6.5. $ctype_T$ Props: $A \equiv T'' \to T'$
   - 6.6. $ctype_T$ Props: A is a subexpression of T
   - 6.8. $(T, m_1) > (T'', [m_2/y]m')$
   - 6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

6. Suppose $m_1 \equiv \lambda y : T''.m'$.
   6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$ ✓

   6.2. $[n/x]n' \leadsto^* m_1\ m_2 \equiv (\lambda y : T''.m')\ m_2 \leadsto [m_2/y]m'$ ✓

   TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

   6.3. h is not a λ-abstraction and m_1 is a λ-abstraction

   6.4. $ctype_T$ Props: $ctype_T(x, h) = A$

   6.5. $ctype_T$ Props: $A \equiv T'' \to T'$

   6.6. $ctype_T$ Props: A is a subexpression of T

   6.8. $(T, m_1) > (T'', [m_2/y]m')$

   6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

6. Suppose $m_1 \equiv \lambda y : T''.m'$.
   6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$ ✓

   6.2. $[n/x]n' \rightsquigarrow^* m_1\, m_2 \equiv (\lambda y : T''.m')\, m_2 \rightsquigarrow [m_2/y]m'$ ✓

   TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

   6.3. h is not a λ-abstraction and m_1 is a λ-abstraction ✓

   6.4. $ctype_T$ Props: $ctype_T(x, h) = A$

   6.5. $ctype_T$ Props: $A \equiv T'' \to T'$

   6.6. $ctype_T$ Props: A is a subexpression of T

   6.8. $(T, m_1) > (T'', [m_2/y]m')$

   6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

6. Suppose $m_1 \equiv \lambda y : T''.m'$.
   6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$ ✓
   6.2. $[n/x]n' \rightsquigarrow^* m_1\, m_2 \equiv (\lambda y : T''.m')\, m_2 \rightsquigarrow [m_2/y]m'$ ✓
   TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$
   6.3. h is not a λ-abstraction and m_1 is a λ-abstraction ✓
   6.4. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   6.5. $ctype_T$ Props: $A \equiv T'' \rightarrow T'$
   6.6. $ctype_T$ Props: A is a subexpression of T
   6.8. $(T, m_1) > (T'', [m_2/y]m')$
   6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

6. Suppose $m_1 \equiv \lambda y : T''.m'$.
   6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$ ✓

   6.2. $[n/x]n' \rightsquigarrow^* m_1\ m_2 \equiv (\lambda y : T''.m')\ m_2 \rightsquigarrow [m_2/y]m'$ ✓

   TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

   6.3. h is not a λ-abstraction and m_1 is a λ-abstraction ✓

   6.4. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓

   6.5. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓

   6.6. $ctype_T$ Props: A is a subexpression of T

   6.8. $(T, m_1) > (T'', [m_2/y]m')$

   6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

6. Suppose $m_1 \equiv \lambda y : T''.m'$.
   6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$ ✓
   6.2. $[n/x]n' \rightsquigarrow^* m_1\ m_2 \equiv (\lambda y : T''.m')\ m_2 \rightsquigarrow [m_2/y]m'$ ✓
   TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$
   6.3. h is not a λ-abstraction and m_1 is a λ-abstraction ✓
   6.4. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   6.5. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   6.6. $ctype_T$ Props: A is a subexpression of T ✓
   6.8. $(T, m_1) > (T'', [m_2/y]m')$
   6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

6. Suppose $m_1 \equiv \lambda y : T''.m'$.
   - 6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$ ✓
   - 6.2. $[n/x]n' \leadsto^* m_1\, m_2 \equiv (\lambda y : T''.m')\, m_2 \leadsto [m_2/y]m'$ ✓
   - TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 6.3. h is not a λ-abstraction and m_1 is a λ-abstraction ✓
   - 6.4. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   - 6.5. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   - 6.6. $ctype_T$ Props: A is a subexpression of T ✓
   - 6.8. $(T, m_1) > (T'', [m_2/y]m')$ ✓
   - 6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

6. Suppose $m_1 \equiv \lambda y : T''.m'$.
   - 6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$ ✓
   - 6.2. $[n/x]n' \rightsquigarrow^* m_1\ m_2 \equiv (\lambda y : T''.m')\ m_2 \rightsquigarrow [m_2/y]m'$ ✓
   - TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 6.3. h is not a λ-abstraction and m_1 is a λ-abstraction ✓
   - 6.4. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   - 6.5. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   - 6.6. $ctype_T$ Props: A is a subexpression of T ✓
   - 6.8. $(T, m_1) > (T'', [m_2/y]m')$ ✓
   - 6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$ ✓

A Peek at the Proof

6. Suppose $m_1 \equiv \lambda y : T''.m'$. ✓
   - 6.1. $m' \in [\![T']\!]_{\Gamma,\Gamma',y:T''}$ ✓
   - 6.2. $[n/x]n' \leadsto^* m_1\ m_2 \equiv (\lambda y : T''.m')\ m_2 \leadsto [m_2/y]m'$ ✓
   - TS. $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 6.3. h is not a λ-abstraction and m_1 is a λ-abstraction ✓
   - 6.4. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   - 6.5. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   - 6.6. $ctype_T$ Props: A is a subexpression of T ✓
   - 6.8. $(T, m_1) > (T'', [m_2/y]m')$ ✓
   - 6.9. IH(2): $[m_2/y]m' \in [\![T']\!]_{\Gamma,\Gamma'}$ ✓

A Peek at the Proof

7. Suppose $m_1 \equiv \mu y.m'$
   - 7.1. *ctype*$_T$ Props: $ctype_T(x, h) = A$
   - 7.2. *ctype*$_T$ Props: $A \equiv T'' \to T'$
   - 7.3. *ctype*$_T$ Props: A is a subexpression of T
   - 7.4. $(T, m_1) > (T'', m_1)$
   - 7.5. Consider: $(\mu y.m')\, r$, where r is fresh of type T''
   - 7.6. IH(1): $(\mu y.m')\, r \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   - 7.7. $(\mu y.m')\, r \rightsquigarrow^* m'' \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   - 7.8. IH(2): $[m_2/r]m'' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 7.9. $(\mu y.m')\, r \rightsquigarrow \mu z.[\lambda u : T'' \to T'.(z\,(u\,r))/y]m'$

A Peek at the Proof

7. Suppose $m_1 \equiv \mu y.m'$
   7.1. $ctype_T$ Props: $ctype_T(x, h) = A$
   7.2. $ctype_T$ Props: $A \equiv T'' \rightarrow T'$
   7.3. $ctype_T$ Props: A is a subexpression of T
   7.4. $(T, m_1) > (T'', m_1)$
   7.5. Consider: $(\mu y.m')\, r$, where r is fresh of type T''
   7.6. IH(1): $(\mu y.m')\, r \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   7.7. $(\mu y.m')\, r \rightsquigarrow^* m'' \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   7.8. IH(2): $[m_2/r]m'' \in [\![T']\!]_{\Gamma,\Gamma'}$
   7.9. $(\mu y.m')\, r \rightsquigarrow \mu z.[\lambda u : T'' \rightarrow T'.(z\,(u\,r))/y]m'$

A Peek at the Proof

7. Suppose $m_1 \equiv \mu y.m'$
   - 7.1. $ctype_T$ Props: $ctype_T(x,h) = A$ ✓
   - 7.2. $ctype_T$ Props: $A \equiv T'' \to T'$
   - 7.3. $ctype_T$ Props: A is a subexpression of T
   - 7.4. $(T, m_1) > (T'', m_1)$
   - 7.5. Consider: $(\mu y.m')\, r$, where r is fresh of type T''
   - 7.6. IH(1): $(\mu y.m')\, r \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   - 7.7. $(\mu y.m')\, r \rightsquigarrow^* m'' \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   - 7.8. IH(2): $[m_2/r]m'' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 7.9. $(\mu y.m')\, r \rightsquigarrow \mu z.[\lambda u : T'' \to T'.(z\,(u\,r))/y]m'$

A Peek at the Proof

7. Suppose $m_1 \equiv \mu y.m'$
   7.1. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   7.2. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   7.3. $ctype_T$ Props: A is a subexpression of T
   7.4. $(T, m_1) > (T'', m_1)$
   7.5. Consider: $(\mu y.m')\, r$, where r is fresh of type T''
   7.6. IH(1): $(\mu y.m')\, r \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   7.7. $(\mu y.m')\, r \rightsquigarrow^* m'' \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   7.8. IH(2): $[m_2/r]m'' \in [\![T']\!]_{\Gamma,\Gamma'}$
   7.9. $(\mu y.m')\, r \rightsquigarrow \mu z.[\lambda u : T'' \to T'.(z\,(u\,r))/y]m'$

A Peek at the Proof

7. Suppose $m_1 \equiv \mu y.m'$
   - 7.1. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   - 7.2. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   - 7.3. $ctype_T$ Props: A is a subexpression of T ✓
   - 7.4. $(T, m_1) > (T'', m_1)$
   - 7.5. Consider: $(\mu y.m')\, r$, where r is fresh of type T''
   - 7.6. IH(1): $(\mu y.m')\, r \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   - 7.7. $(\mu y.m')\, r \leadsto^* m'' \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   - 7.8. IH(2): $[m_2/r]m'' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 7.9. $(\mu y.m')\, r \leadsto \mu z.[\lambda u : T'' \to T'.(z\,(u\,r))/y]m'$

A Peek at the Proof

7. Suppose $m_1 \equiv \mu y.m'$
   - 7.1. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   - 7.2. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   - 7.3. $ctype_T$ Props: A is a subexpression of T ✓
   - 7.4. $(T, m_1) > (T'', m_1)$ ✓
   - 7.5. Consider: $(\mu y.m')\, r$, where r is fresh of type T''
   - 7.6. IH(1): $(\mu y.m')\, r \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   - 7.7. $(\mu y.m')\, r \rightsquigarrow^* m'' \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   - 7.8. IH(2): $[m_2/r]m'' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 7.9. $(\mu y.m')\, r \rightsquigarrow \mu z.[\lambda u : T'' \to T'.(z\,(u\,r))/y]m'$

A Peek at the Proof

7. Suppose $m_1 \equiv \mu y.m'$
   - 7.1. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   - 7.2. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   - 7.3. $ctype_T$ Props: A is a subexpression of T ✓
   - 7.4. $(T, m_1) > (T'', m_1)$ ✓
   - 7.5. Consider: $(\mu y.m')\, r$, where r is fresh of type T''
   - 7.6. IH(1): $(\mu y.m')\, r \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   - 7.7. $(\mu y.m')\, r \rightsquigarrow^* m'' \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   - 7.8. IH(2): $[m_2/r]m'' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 7.9. $(\mu y.m')\, r \rightsquigarrow \mu z.[\lambda u : T'' \to T'.(z\,(u\,r))/y]m'$

A Peek at the Proof

7. Suppose $m_1 \equiv \mu y.m'$
   - 7.1. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   - 7.2. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   - 7.3. $ctype_T$ Props: A is a subexpression of T ✓
   - 7.4. $(T, m_1) > (T'', m_1)$ ✓
   - 7.5. Consider: $(\mu y.m')\, r$, where r is fresh of type T''
   - 7.6. IH(1): $(\mu y.m')\, r \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$ ✓
   - 7.7. $(\mu y.m')\, r \rightsquigarrow^* m'' \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$
   - 7.8. IH(2): $[m_2/r]m'' \in [\![T']\!]_{\Gamma,\Gamma'}$
   - 7.9. $(\mu y.m')\, r \rightsquigarrow \mu z.[\lambda u : T'' \to T'.(z\,(u\,r))/y]m'$

A Peek at the Proof

7. Suppose $m_1 \equiv \mu y.m'$
   7.1. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   7.2. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   7.3. $ctype_T$ Props: A is a subexpression of T ✓
   7.4. $(T, m_1) > (T'', m_1)$ ✓
   7.5. Consider: $(\mu y.m')\, r$, where r is fresh of type T''
   7.6. IH(1): $(\mu y.m')\, r \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$ ✓
   7.7. $(\mu y.m')\, r \rightsquigarrow^* m'' \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$ ✓
   7.8. IH(2): $[m_2/r]m'' \in [\![T']\!]_{\Gamma,\Gamma'}$
   7.9. $(\mu y.m')\, r \rightsquigarrow \mu z.[\lambda u : T'' \to T'.(z\,(u\,r))/y]m'$

A Peek at the Proof

7. Suppose $m_1 \equiv \mu y.m'$
   - 7.1. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   - 7.2. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   - 7.3. $ctype_T$ Props: A is a subexpression of T ✓
   - 7.4. $(T, m_1) > (T'', m_1)$ ✓
   - 7.5. Consider: $(\mu y.m')\, r$, where r is fresh of type T''
   - 7.6. IH(1): $(\mu y.m')\, r \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$ ✓
   - 7.7. $(\mu y.m')\, r \rightsquigarrow^* m'' \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$ ✓
   - 7.8. IH(2): $[m_2/r]m'' \in [\![T']\!]_{\Gamma,\Gamma'}$ ✓
   - 7.9. $(\mu y.m')\, r \rightsquigarrow \mu z.[\lambda u : T'' \to T'.(z\,(u\,r))/y]m'$

A Peek at the Proof

7. Suppose $m_1 \equiv \mu y.m'$
   7.1. $ctype_T$ Props: $ctype_T(x, h) = A$ ✓
   7.2. $ctype_T$ Props: $A \equiv T'' \to T'$ ✓
   7.3. $ctype_T$ Props: A is a subexpression of T ✓
   7.4. $(T, m_1) > (T'', m_1)$ ✓
   7.5. Consider: $(\mu y.m')\, r$, where r is fresh of type T''
   7.6. IH(1): $(\mu y.m')\, r \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$ ✓
   7.7. $(\mu y.m')\, r \rightsquigarrow^* m'' \in [\![T']\!]_{\Gamma,\Gamma',r:T''}$ ✓
   7.8. IH(2): $[m_2/r]m'' \in [\![T']\!]_{\Gamma,\Gamma'}$ ✓
   7.9. $(\mu y.m')\, r \rightsquigarrow \mu z.[\lambda u : T'' \to T'.(z\,(u\,r))/y]m'$ ✓

A Peek at the Proof

8.0. $[m_2/r]((\mu y.m')\, r) \rightsquigarrow [m_2/r](\mu z.([\lambda u : T'' \to T'.(z\,(u\, r))/y]m')) \rightsquigarrow [m_2/r]m''$

8.1. $(\mu y.m')\, m_2 \equiv [m_2/r]((\mu y.m')\, r) \in [\![T']\!]_{\Gamma,\Gamma'}$

8.2. $m_1\, m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

8.0. $[m_2/r]((\mu y.m')\, r) \rightsquigarrow [m_2/r](\mu z.([\lambda u : T'' \to T'.(z\,(u\, r))/y]m')) \rightsquigarrow [m_2/r]m''$ ✓

8.1. $(\mu y.m')\, m_2 \equiv [m_2/r]((\mu y.m')\, r) \in [\![T']\!]_{\Gamma,\Gamma'}$

8.2. $m_1\, m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

8.0. $[m_2/r]((\mu y.m')\, r) \rightsquigarrow [m_2/r](\mu z.([\lambda u : T'' \to T'.(z\,(u\,r))/y]m')) \rightsquigarrow [m_2/r]m''$ ✓

8.1. $(\mu y.m')\, m_2 \equiv [m_2/r]((\mu y.m')\, r) \in [\![T']\!]_{\Gamma,\Gamma'}$ ✓

8.2. $m_1\, m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$

A Peek at the Proof

8.0. $[m_2/r]((\mu y.m')\, r) \rightsquigarrow [m_2/r](\mu z.([\lambda u : T'' \to T'.(z\,(u\,r))/y]m')) \rightsquigarrow [m_2/r]m''$ ✓

8.1. $(\mu y.m')\, m_2 \equiv [m_2/r]((\mu y.m')\, r) \in [\![T']\!]_{\Gamma,\Gamma'}$ ✓

8.2. $m_1\, m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$ ✓

A Peek at the Proof

8.0. $[m_2/r]((\mu y.m')\,r) \rightsquigarrow [m_2/r](\mu z.([\lambda u : T'' \to T'.(z\,(u\,r))/y]m')) \rightsquigarrow [m_2/r]m''$ ✓

8.1. $(\mu y.m')\,m_2 \equiv [m_2/r]((\mu y.m')\,r) \in [\![T']\!]_{\Gamma,\Gamma'}$ ✓

8.2. $m_1\,m_2 \in [\![T']\!]_{\Gamma,\Gamma'}$ ✓

Q.E.D!

Concluding Normalization

Theorem (Type Soundness)

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

Corollary (Normalization)

If $\Gamma \vdash t : T$ *then there exists a term* n *such that* $t \rightsquigarrow^! n$.

Conclusion

- Summary
  - Introduced the $\lambda\mu$-Caclulus and hereditary substitution.
  - Proved WN of the $\lambda\mu$-Calculus.

- What can we conclude from this result?
  - This proof is less complex than already known proofs of WN.
    - J. David and K. Nour:2003.
  - While short this proof does not shed any light on defining the explicit hereditary substitution function.