Formalizing Strong Normalization Proofs

Kazuhiko Sakaguchi

College of Information Science, University of Tsukuba

2014/12/3 TPP2014

Strong Normalization Theorem

In typed λ-calculi, strong normalization (SN) theorem is as follows.

$$\forall \Gamma, t, \tau.\, \Gamma \vdash t : \tau \Rightarrow \mathrm{SN}(t)$$

Strong Normalization Theorem

In typed λ-calculi, strong normalization (SN) theorem is as follows.

$$\forall \Gamma, t, \tau.\, \underbrace{\Gamma \vdash t : \tau} \Rightarrow \mathrm{SN}(t)$$

If t is a typed term,

Strong Normalization Theorem

In typed λ-calculi, strong normalization (SN) theorem is as follows.

$$\forall \Gamma, t, \tau.\, \underline{\Gamma \vdash t : \tau} \Rightarrow \underline{\mathrm{SN}(t)}$$

If t is a typed term,

then all reduction sequences from t are finite.

Strong Normalization Theorem

In typed λ-calculi, strong normalization (SN) theorem is as follows.

$$\forall \Gamma, t, \tau.\, \underline{\Gamma \vdash t : \tau} \Rightarrow \underline{\mathrm{SN}(t)}$$

If t is a typed term,

then all reduction sequences from t are finite.

Non-terminating example of a untyped λ-term:

$$\begin{aligned} & (\lambda x.\, x\, x)\, (\lambda x.\, x\, x) \\ \rightarrow_\beta\ & (\lambda x.\, x\, x)\, (\lambda x.\, x\, x) \end{aligned}$$

Our Formalization of the λ-Calculus [Sak15]

- https://github.com/pi8027/lambda-calculus
- Goal
  - Formalize many diffrent proofs of the strong normalization theorem in Coq.
  - Build a general framework for formalizations of the strong normalization theorem.

Our Formalization of the λ-Calculus [Sak15]

- https://github.com/pi8027/lambda-calculus
- Goal
  - Formalize many differnt proofs of the strong normalization theorem in Coq.
  - Build a general framework for formalizations of the strong normalization theorem.
- Current developments
  - using de Bruijn representation
  - untyped λ-calculus
    - Church-Rosser theorem
  - simply typed λ-calculus ($\lambda^{\to}$) and System F ($\lambda 2$)
    - subject reduction theorem
    - strong normalization theorem
      (contains 3 different definitions of the reducibility for each system)

Our Formalization of the λ-Calculus [Sak15]

- https://github.com/pi8027/lambda-calculus
- Goal
  - Formalize many differnt proofs of the strong normalization theorem in Coq.
  - Build a general framework for formalizations of the strong normalization theorem.
- Current developments
  - using de Bruijn representation
  - untyped λ-calculus
    - Church-Rosser theorem
  - simply typed λ-calculus ($\lambda^{\to}$) and System F ($\lambda 2$)
    - subject reduction theorem
    - strong normalization theorem (contains 3 different definitions of the reducibility for each system)

Part I

Nameless Terms and Proof Automation

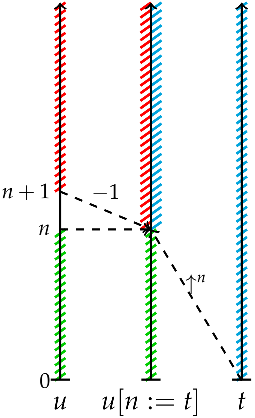
Substitution for Nameless Terms

$$m[n := t] = \begin{cases} m-1 & \text{if } n < m \\ t\uparrow^n & \text{if } n = m \\ m & \text{if } n > m \end{cases}$$

$$(u\,v)[n := t] = u[n := t]\,v[n := t]$$

$$(\lambda u)[n := t] = \lambda u[n+1 := t]$$

$\mathbf{t \uparrow^n}$ is a term which is obtained by adding n to all the free variables of t. This operation is called a **shift** or **lift**.

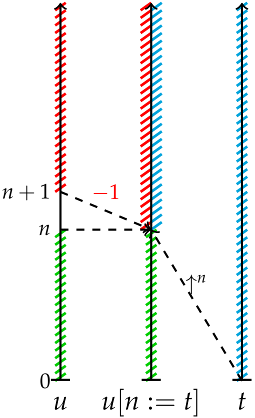


Substitution for Nameless Terms

$$m[n := t] = \begin{cases} m - 1 & \text{if } n < m \\ t \uparrow^n & \text{if } n = m \\ m & \text{if } n > m \end{cases}$$

$$(u\, v)[n := t] = u[n := t]\, v[n := t]$$

$$(\lambda u)[n := t] = \lambda u[n + 1 := t]$$



$\mathbf{t} \uparrow^{\mathbf{n}}$ is a term which is obtained by adding n to all the free variables of t. This operation is called a **shift** or **lift**.

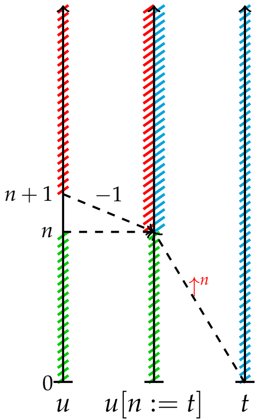
Substitution for Nameless Terms

$$m[n := t] = \begin{cases} m-1 & \text{if } n < m \\ t\uparrow^{n} & \text{if } n = m \\ m & \text{if } n > m \end{cases}$$

$$(u\,v)[n := t] = u[n := t]\,v[n := t]$$

$$(\lambda u)[n := t] = \lambda u[n+1 := t]$$

$t\uparrow^{n}$ is a term which is obtained by adding n to all the free variables of t. This operation is called a **shift** or **lift**.



Shift

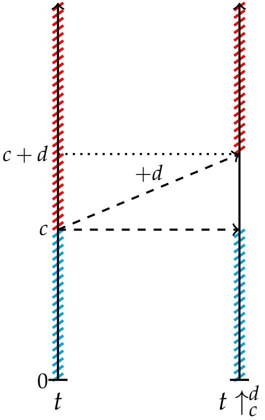
$\mathbf{t} \uparrow^{\mathbf{d}}_{\mathbf{c}}$ adds d to all the free variable of t that are greater than or equal to c.

$$n \uparrow^d_c = \begin{cases} n + d & \text{if } c \leq n \\ n & \text{if } n < c \end{cases}$$

$$(t\, u) \uparrow^d_c = t \uparrow^d_c \; u \uparrow^d_c$$

$$(\lambda\, t) \uparrow^d_c = \lambda\, t \uparrow^d_{c+1}$$

$$t \uparrow^d = t \uparrow^d_0$$



Shift

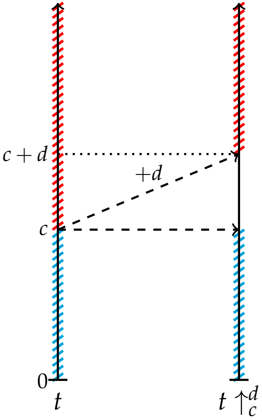
$\mathbf{t} \uparrow^{\mathbf{d}}_{\mathbf{c}}$ adds d to all the free variable of t that are greater than or equal to c.

$$n \uparrow^d_c = \begin{cases} n + d & \text{if } c \leq n \\ n & \text{if } n < c \end{cases}$$

$$(t\, u) \uparrow^d_c = t \uparrow^d_c \; u \uparrow^d_c$$

$$(\lambda\, t) \uparrow^d_c = \lambda\, t \uparrow^d_{c+1}$$

$$t \uparrow^d = t \uparrow^d_0$$



Shift

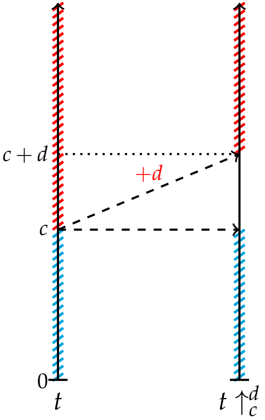
$\mathbf{t} \uparrow^{\mathbf{d}}_{\mathbf{c}}$ adds d to all the free variable of t that are greater than or equal to c.

$$n \uparrow^d_c = \begin{cases} n + d & \text{if } c \leq n \\ n & \text{if } n < c \end{cases}$$

$$(t\,u) \uparrow^d_c = t \uparrow^d_c \; u \uparrow^d_c$$

$$(\lambda\, t) \uparrow^d_c = \lambda\, t \uparrow^d_{c+1}$$

$$t \uparrow^d = t \uparrow^d_0$$



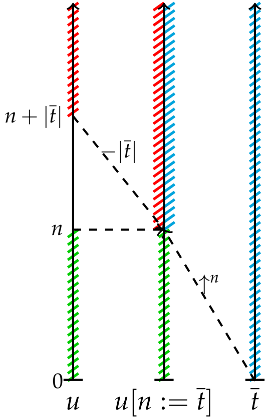
(Restricted) Parallel Substitution

$\mathbf{u}[\mathbf{n} := \mathbf{t_1}, \ldots, \mathbf{t_m}]$ substitutes $t_1, \ldots, t_m$ for free variables $n, \ldots, n+m-1$ in u.

$$m[n := \bar{t}] = \begin{cases} m - |\bar{t}| & \text{if } n + |\bar{t}| \leq m \\ \bar{t}_{m-n} \uparrow^n & \text{if } n \leq m < n + |\bar{t}| \\ m & \text{if } m < n \end{cases}$$

$$(u\,v)[n := \bar{t}] = (u[n := \bar{t}])\,(v[n := \bar{t}])$$

$$(\lambda\,u)[n := \bar{t}] = \lambda\,(u[n+1 := \bar{t}])$$



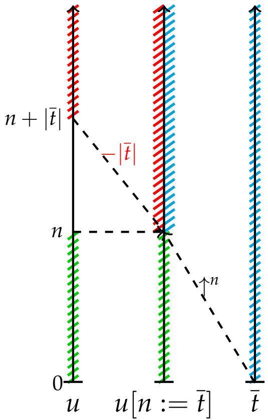
(Restricted) Parallel Substitution

$\mathbf{u}[\mathbf{n} := \mathbf{t_1}, \ldots, \mathbf{t_m}]$ substitutes $t_1, \ldots, t_m$ for free variables $n, \ldots, n+m-1$ in u.

$$m[n := \bar{t}] = \begin{cases} m - |\bar{t}| & \text{if } n + |\bar{t}| \leq m \\ \bar{t}_{m-n} \uparrow^n & \text{if } n \leq m < n + |\bar{t}| \\ m & \text{if } m < n \end{cases}$$

$$(u\, v)[n := \bar{t}] = (u[n := \bar{t}])\,(v[n := \bar{t}])$$

$$(\lambda\, u)[n := \bar{t}] = \lambda\,(u[n+1 := \bar{t}])$$



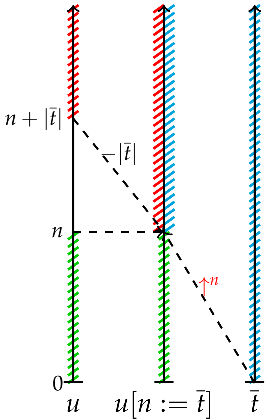
(Restricted) Parallel Substitution

$\mathbf{u}[\mathbf{n} := \mathbf{t_1}, \ldots, \mathbf{t_m}]$ substitutes $t_1, \ldots, t_m$ for free variables $n, \ldots, n+m-1$ in u.

$$m[n := \bar{t}] = \begin{cases} m - |\bar{t}| & \text{if } n + |\bar{t}| \leq m \\ \bar{t}_{m-n} \uparrow^{n} & \text{if } n \leq m < n + |\bar{t}| \\ m & \text{if } m < n \end{cases}$$

$$(u\, v)[n := \bar{t}] = (u[n := \bar{t}])\,(v[n := \bar{t}])$$

$$(\lambda\, u)[n := \bar{t}] = \lambda\,(u[n+1 := \bar{t}])$$



(Restricted) Parallel Substitution

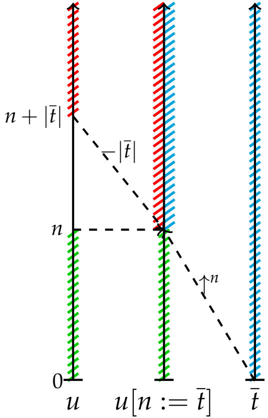
$\mathbf{u}[\mathbf{n} := \mathbf{t_1}, \ldots, \mathbf{t_m}]$ substitutes $t_1, \ldots, t_m$ for free variables $n, \ldots, n+m-1$ in u.

$$m[n := \bar{t}] = \begin{cases} m - |\bar{t}| & \text{if } n + |\bar{t}| \leq m \\ \bar{t}_{m-n} \uparrow^n & \text{if } n \leq m < n + |\bar{t}| \\ m & \text{if } m < n \end{cases}$$

$$(u\, v)[n := \bar{t}] = (u[n := \bar{t}])\, (v[n := \bar{t}])$$

$$(\lambda\, u)[n := \bar{t}] = \lambda\, (u[n+1 := \bar{t}])$$

This is useful for proving the strong normalization theorem.



Equational Properties

of Shift and Parallel Substitution

$$t \uparrow^0_n = t \tag{1}$$

$$c \le c' \le c + d \Rightarrow t \uparrow^d_c \uparrow^{d'}_{c'} = t \uparrow^{d'+d}_c \tag{2}$$

$$c' \le c \Rightarrow t \uparrow^d_c \uparrow^{d'}_{c'} = t \uparrow^{d'}_{c'} \uparrow^d_{d'+c} \tag{3}$$

$$c \le n \Rightarrow t[n := \overline{u}] \uparrow^d_c = t \uparrow^d_c [d + n := \overline{u}] \tag{4}$$

$$n \le c \Rightarrow t[n := \overline{u}] \uparrow^d_c = t \uparrow^d_{|\overline{u}|+c} [n := \overline{u} \uparrow^d_{c-n}] \tag{5}$$

$$c \le n \wedge |\overline{u}| + n \le d + c \Rightarrow$$

$$t \uparrow^d_c [n := \overline{u}] = t \uparrow^{d-|\overline{u}|}_c \tag{6}$$

$$m \le n \Rightarrow t[m := \overline{u}][n := \overline{v}] = t[|\overline{u}| + n := \overline{v}][m := \overline{u}[n - m := \overline{v}]] \tag{7}$$

$$t[|\overline{v}| + n := \overline{u}][n := \overline{v}] = t[n := \overline{v} +\!\!+ \overline{u}] \tag{8}$$

$$t[n := []] = t \tag{9}$$

where

$$\bar{t} \uparrow^d_c = [t \uparrow^d_c | \; t \leftarrow \bar{t}]$$

$$\bar{t}[n := \overline{u}] = [t[n := \overline{u}] \mid t \leftarrow \bar{t}]$$

Equational Property (5)

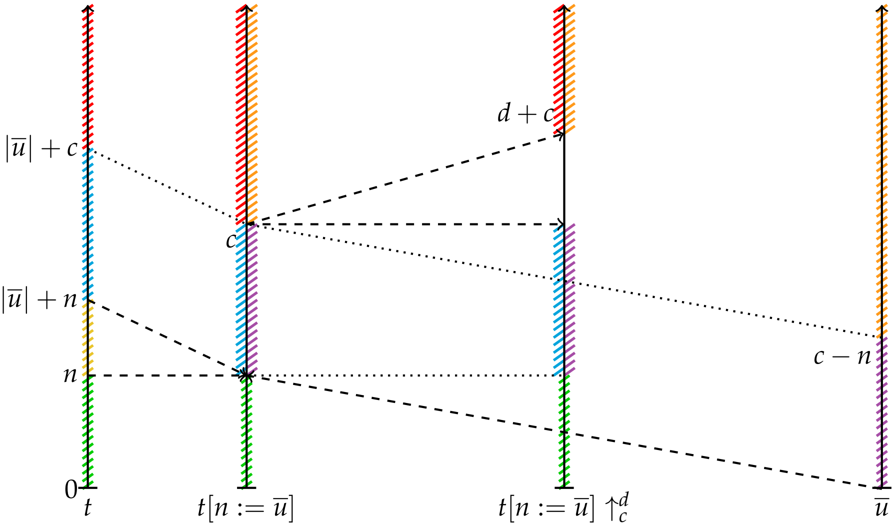
$$n \leq c \Rightarrow t[n := \overline{u}] \uparrow^d_c = t \uparrow^d_{|\overline{u}|+c} [n := \overline{u} \uparrow^d_{c-n}]$$

Equational Property (5)

$$n \leq c \Rightarrow t[n := \overline{u}] \uparrow^d_c = t \uparrow^d_{|\overline{u}|+c} [n := \overline{u} \uparrow^d_{c-n}]$$

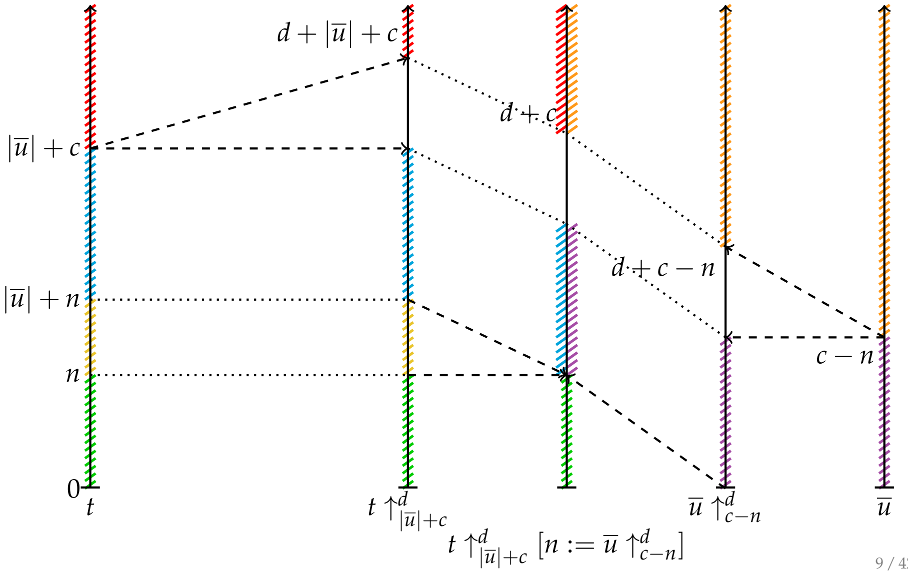
Equational Property (5)

$$n \leq c \Rightarrow t[n := \overline{u}] \uparrow^d_c = t \uparrow^d_{|\overline{u}|+c} [n := \overline{u} \uparrow^d_{c-n}]$$



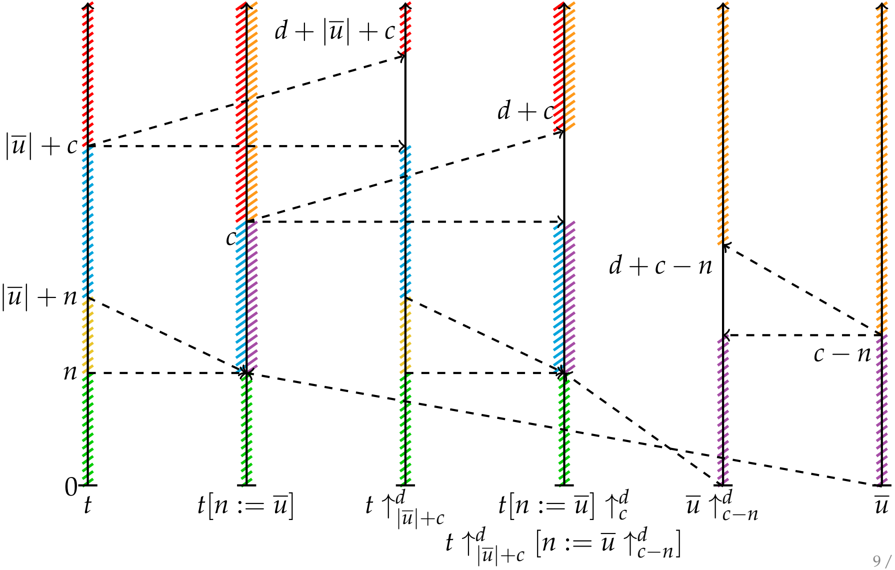
Equational Property (5)

$$n \leq c \Rightarrow t[n :=\overline{u}] \uparrow^d_c = t \uparrow^d_{|\overline{u}|+c} [n :=\overline{u} \uparrow^d_{c-n}]$$



Equational Property (5)

$$n \le c \Rightarrow t[n := \overline{u}] \uparrow^d_c = t \uparrow^d_{|\overline{u}|+c} [n := \overline{u} \uparrow^d_{c-n}]$$



Proof Outline of the Equational Properties

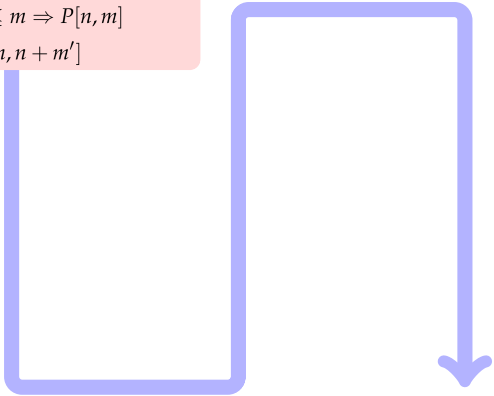
We can apply some automation techniques for these proofs.

Proof Outline of the Equational Properties

We can apply some automation techniques for these proofs.

eliminating hypotheses

$$\forall m. n \leq m \Rightarrow P[n,m]$$
$$\hookrightarrow \forall m'. P[n, n+m']$$



Proof Outline of the Equational Properties

We can apply some automation techniques for these proofs.

eliminating hypotheses

$$\forall m.\, n \leq m \Rightarrow P[n,m]$$
$$\hookrightarrow \forall m'.\, P[n, n+m']$$

structural induction on term

Proof Outline of the Equational Properties

We can apply some automation techniques for these proofs.

eliminating hypotheses

$$\forall m.\, n \le m \Rightarrow P[n, m]$$
$$\hookrightarrow \forall m'.\, P[n, n + m']$$

structural induction on term

applying congruence tactic

with some hypotheses:

$Sm + n = S(m + n), m + Sn = S(m + n)$

Proof Outline of the Equational Properties

We can apply some automation techniques for these proofs.

eliminating hypotheses

$$\forall m.\, n \leq m \Rightarrow P[n,m]$$
$$\hookrightarrow \forall m'.\, P[n, n+m']$$

structural induction on term

applying `congruence` tactic

with some hypotheses:

$Sm + n = S(m+n), m + Sn = S(m+n)$

case analysis

`do !case: ifP`

Proof Outline of the Equational Properties

We can apply some automation techniques for these proofs.

eliminating hypotheses

$$\forall m.\, n \le m \Rightarrow P[n,m]$$
$$\hookrightarrow \forall m'.\, P[n, n+m']$$

structural induction on term

applying `congruence` tactic

with some hypotheses:

$Sm + n = S(m+n), m + Sn = S(m+n)$

case analysis

`do !case: ifP`

automation based on `omega`

Proof Outline of the Equational Properties

We can apply some automation techniques for these proofs.

eliminating hypotheses

$$\forall m. n \leq m \Rightarrow P[n, m]$$
$$\hookrightarrow \forall m'. P[n, n + m']$$

structural induction on term

applying `congruence` tactic

with some hypotheses:

$Sm + n = S(m + n), m + Sn = S(m + n)$

case analysis

`do !case: ifP`

automation based on `omega`

manual proof

Example

```
Lemma subst_shift_distr n d c ts t :
  n <= c ->
  shift d c (substitute n ts t) =
  substitute n (map (shift d (c - n)) ts)
    (shift d (size ts + c) t).
Proof.
  elimleq; elim: t n; congruence' => v n; elimif_omega.
  - rewrite !nth_default ?size_map /=; elimif_omega.
  - rewrite -shift_shift_distr // nth_map' /=;
      congr shift; apply nth_equal;
      rewrite size_map; elimif_omega.
Qed.
```

Performance Problem of omega

```
Notation minn x y := (x - (x - y)).

Lemma minnA x y z :
  minn x (minn y z) = minn (minn x y) z.
Proof. omega.
```

Performance Problem of `omega`

`minn` is the smallest number of two natural numbers.

```
Notation minn x y := (x - (x - y)).

Lemma minnA x y z :
  minn x (minn y z) = minn (minn x y) z.
Proof. omega.
```

Performance Problem of `omega`

`minn` is the smallest number of two natural numbers.

$x - (x - y)$ is a frequently appearing pattern in proofs relevant to the de Bruijn representation.

```
Notation minn x y := (x - (x - y)).

Lemma minnA x y z :
  minn x (minn y z) = minn (minn x y) z.
Proof. omega.
```

Performance Problem of `omega`

`minn` is the smallest number of two natural numbers.

$x - (x - y)$ is a frequently appearing pattern in proofs relevant to the de Bruijn representation.

```
Notation minn x y := (x - (x - y)).

Lemma minnA x y z :
  minn x (minn y z) = minn (minn x y) z.
Proof. omega.
```

associativity of `minn`

Performance Problem of `omega`

`minn` is the smallest number of two natural numbers.

$x - (x - y)$ is a frequently appearing pattern in proofs relevant to the de Bruijn representation.

```
Notation minn x y := (x - (x - y)).

Lemma minnA x y z :
  minn x (minn y z) = minn (minn x y) z.
Proof. omega.
```

associativity of `minn`

runs forever

Performance Problem of `omega`

`minn` is the smallest number of two natural numbers.

$x - (x - y)$ is a frequently appearing pattern in proofs relevant to the de Bruijn representation.

```
Notation minn x y := (x - (x - y)).

Lemma minnA x y z :
  minn x (minn y z) = minn (minn x y) z.
Proof. omega.
```

associativity of `minn`

runs forever

Some techniques are required to use the `omega` tactic for proving the equational properties.

Part II

Strong Normalization Theorem

$\lambda^{\rightarrow}$: Simply Typed λ-Calculus

types and terms:

$$U ::= X \mid (U \rightarrow U)$$

$$t ::= x \mid (t\,t) \mid (\lambda x : U.\, t)$$

typing rules:

$$\frac{\Gamma(x) = U}{\Gamma \vdash x : U}$$

$$\frac{\Gamma \vdash t : U \rightarrow V \qquad \Gamma \vdash u : U}{\Gamma \vdash t\,u : V}$$

$$\frac{\{x : U\} + \Gamma \vdash t : V}{\Gamma \vdash \lambda x : U.\, t : U \rightarrow V}$$

reduction rules:

$$(\lambda x : U.\, t)\, u \rightarrow_\beta t[x := u]$$

$$\frac{t_1 \rightarrow_\beta t_2}{t_1\, u \rightarrow_\beta t_2\, u} \qquad \frac{u_1 \rightarrow_\beta u_2}{t\, u_1 \rightarrow_\beta t\, u_2}$$

$$\frac{t \rightarrow_\beta t'}{\lambda x : U.\, t \rightarrow_\beta \lambda x : U.\, t'}$$

Proof Outline of the SN Theorem

Our proofs are based on the Girard's proof [GTL89].

Proof Outline of the SN Theorem

Our proofs are based on the Girard's proof [GTL89].

$$\Gamma \vdash t : U$$

$$\mathrm{SN}_{\to_\beta}(t)$$

Proof Outline of the SN Theorem

Our proofs are based on the Girard's proof [GTL89].

$\Gamma \vdash t : U$

Proof by induction
on t or U.

$\mathrm{SN}_{\rightarrow_\beta}(t)$

Proof Outline of the SN Theorem

Our proofs are based on the Girard's proof [GTL89].

$\Gamma \vdash t : U$

✗ Proof by induction on t or U.

$\mathrm{SN}_{\rightarrow_\beta}(t)$

Proof Outline of the SN Theorem

Our proofs are based on the Girard's proof [GTL89].

$\Gamma \vdash t : U$

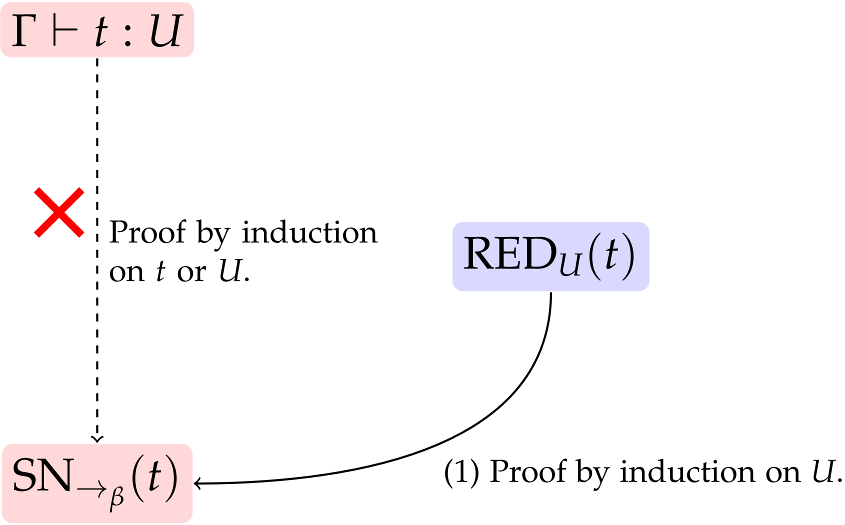
✗ Proof by induction on t or U.

$\mathrm{RED}_U(t)$

$\mathrm{SN}_{\to_\beta}(t)$

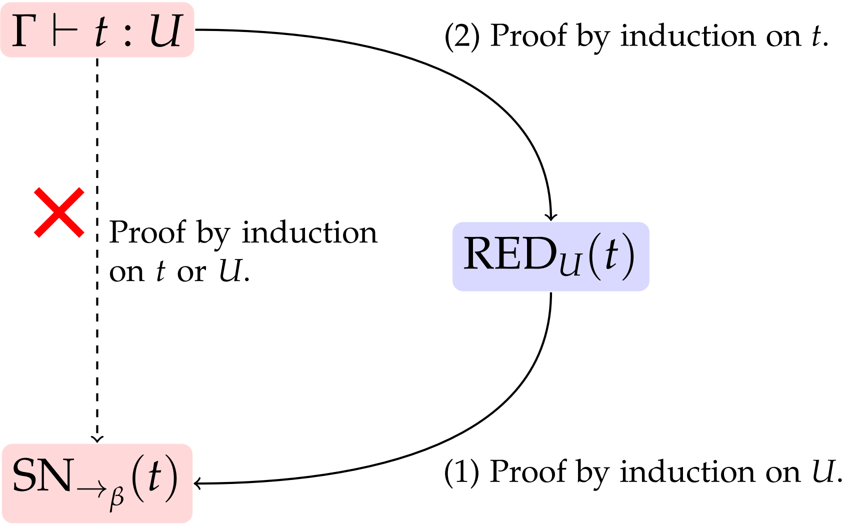
Proof Outline of the SN Theorem

Our proofs are based on the Girard's proof [GTL89].



Proof Outline of the SN Theorem

Our proofs are based on the Girard's proof [GTL89].



What is RED_U?
in the Simply Typed λ-Calculus

RED_U (**reducibility**) is defined by induction on the type U as follows:

$$\mathrm{RED}_X(t) \stackrel{\text{def}}{\iff} \mathrm{SN}_{\to_\beta}(t)$$
$$\mathrm{RED}_{U\to V}(t) \stackrel{\text{def}}{\iff} \forall u.\, \mathrm{RED}_U(u) \Rightarrow \mathrm{RED}_V(t\,u).$$

What is RED_U?
in the Simply Typed λ-Calculus

RED_U (**reducibility**) is defined by induction on the type U as follows:

$$\mathrm{RED}_X(t) \overset{\text{def}}{\iff} \mathrm{SN}_{\to_\beta}(t)$$
$$\mathrm{RED}_{U \to V}(t) \overset{\text{def}}{\iff} \forall u.\, \mathrm{RED}_U(u) \Rightarrow \mathrm{RED}_V(t\,u).$$

In typical definitions, RED_U is a set of typed terms. But this definition of RED_U contains untyped terms. For example,

$$(\lambda x : U.\, x\,x) \in \mathrm{RED}_X.$$

Part 1: Reducible Terms are SN

CR_1 $\mathrm{RED}_U(t) \Rightarrow \mathrm{SN}_{\to_\beta}(t)$

CR_2 $t \to_\beta t' \wedge \mathrm{RED}_U(t) \Rightarrow \mathrm{RED}_U(t')$

CR_3 $\mathrm{neutral}(t) \wedge (\forall t'.\, t \to_\beta t' \Rightarrow \mathrm{RED}_U(t')) \Rightarrow \mathrm{RED}_U(t)$

Part 1: Reducible Terms are SN

CR_1 $\mathrm{RED}_U(t) \Rightarrow \mathrm{SN}_{\rightarrow_\beta}(t)$

CR_2 $t \rightarrow_\beta t' \wedge \mathrm{RED}_U(t) \Rightarrow \mathrm{RED}_U(t')$

CR_3 $\mathrm{neutral}(t) \wedge (\forall t'.\, t \rightarrow_\beta t' \Rightarrow \mathrm{RED}_U(t')) \Rightarrow \mathrm{RED}_U(t)$

↑ t is not of the form $\lambda x.\, u$.

Part 1: Reducible Terms are SN

CR_1 $\mathrm{RED}_U(t) \Rightarrow \mathrm{SN}_{\to_\beta}(t)$

CR_2 $t \to_\beta t' \wedge \mathrm{RED}_U(t) \Rightarrow \mathrm{RED}_U(t')$

CR_3 $\mathrm{neutral}(t) \wedge (\forall t'.\, t \to_\beta t' \Rightarrow \mathrm{RED}_U(t')) \Rightarrow \mathrm{RED}_U(t)$

↑ t is not of the form $\lambda x.\, u$.

CR_2 is proved by induction on U. $\mathrm{CR}_{1,3}$ are proved together by induction on U.

Part 2: Typed Terms are Reducible

First, we prove the following proposition (reducibility theorem) by induction on t.

$$\begin{aligned}\{x_1 : U_1, \ldots, x_n : U_n, y_1 : V_1, \ldots, y_m : V_m\} \vdash t : U \\ \wedge (\forall i \in \{1, \ldots, n\}.\, \mathrm{RED}_{U_i}(t_i)) \\ \Rightarrow \mathrm{RED}_U(t[x_1, \ldots, x_n := t_1, \ldots, t_n])\end{aligned}$$

Part 2: Typed Terms are Reducible

First, we prove the following proposition (reducibility theorem) by induction on t.

$$\begin{aligned}\{x_1 : U_1, \ldots, x_n : U_n, y_1 : V_1, \ldots, y_m : V_m\} \vdash t : U \\ \wedge (\forall i \in \{1, \ldots, n\}. \, \mathrm{RED}_{U_i}(t_i)) \\ \Rightarrow \mathrm{RED}_U(t[x_1, \ldots, x_n := t_1, \ldots, t_n])\end{aligned}$$

In case of $n = 0$, this proposition is equivalent to

$$\{y_1 : V_1, \ldots, y_m : V_m\} \vdash t : U \Rightarrow \mathrm{RED}_U(t).$$

Part 2: Typed Terms are Reducible

First, we prove the following proposition (reducibility theorem) by induction on t.

$$\begin{aligned}\{x_1 : U_1, \ldots, x_n : U_n, y_1 : V_1, \ldots, y_m : V_m\} \vdash t : U \\ \wedge (\forall i \in \{1, \ldots, n\}.\, \mathrm{RED}_{U_i}(t_i)) \\ \Rightarrow \mathrm{RED}_U(t[x_1, \ldots, x_n := t_1, \ldots, t_n])\end{aligned}$$

In case of $n = 0$, this proposition is equivalent to

$$\{y_1 : V_1, \ldots, y_m : V_m\} \vdash t : U \Rightarrow \mathrm{RED}_U(t).$$

Finally, we get a proof of the strong normalization theorem.

Now, we redefine the reducibility as a set of typed terms.

$$\mathrm{RED'}^{\Gamma}_{X}(t) \stackrel{\text{def}}{\iff} \mathrm{SN}_{\to_\beta}(t)$$
$$\mathrm{RED'}^{\Gamma}_{U\to V}(t) \stackrel{\text{def}}{\iff} \forall u.\, \mathrm{RED}^{\Gamma}_{U}(u) \Rightarrow \mathrm{RED}^{\Gamma}_{V}(t\,u)$$

$$\mathrm{RED}^{\Gamma}_{U}(t) \stackrel{\text{def}}{\iff} \Gamma \vdash t : U \wedge \mathrm{RED'}^{\Gamma}_{U}(t)$$

Typed Reducibility

Unsuccessful Example

Now, we redefine the reducibility as a set of typed terms.

$$\mathrm{RED'}^{\Gamma}_{X}(t) \overset{\mathrm{def}}{\iff} \mathrm{SN}_{\to_\beta}(t)$$

$$\mathrm{RED'}^{\Gamma}_{U\to V}(t) \overset{\mathrm{def}}{\iff} \forall u.\, \mathrm{RED}^{\Gamma}_{U}(u) \Rightarrow \mathrm{RED}^{\Gamma}_{V}(t\,u)$$

$$\mathrm{RED}^{\Gamma}_{U}(t) \overset{\mathrm{def}}{\iff} \Gamma \vdash t : U \wedge \mathrm{RED'}^{\Gamma}_{U}(t)$$

In this definition, proof of CR_1 is unsuccessful.

A Proof of CR_1
in Untyped Settings

CR_1 $\mathrm{RED}_U(t) \Rightarrow \mathrm{SN}_{\rightarrow_\beta}(t)$
CR_3 $\mathrm{neutral}(t) \wedge (\forall t'.\, t \rightarrow_\beta t' \Rightarrow \mathrm{RED}_U(t')) \Rightarrow \mathrm{RED}_U(t)$

$\mathrm{CR}_{1,3}$ *are proved together by induction on* U.

A Proof of CR_1
in Untyped Settings

CR_1 $\mathrm{RED}_U(t) \Rightarrow \mathrm{SN}_{\rightarrow_\beta}(t)$
CR_3 $\mathrm{neutral}(t) \wedge (\forall t'.\, t \rightarrow_\beta t' \Rightarrow \mathrm{RED}_U(t')) \Rightarrow \mathrm{RED}_U(t)$

$\mathrm{CR}_{1,3}$ *are proved together by induction on U.*

Proof. If U is a type variable, CR_1 is a tautology. The only remaining case is $U = V \rightarrow W$.

A Proof of CR_1
in Untyped Settings

CR_1 $\mathrm{RED}_U(t) \Rightarrow \mathrm{SN}_{\rightarrow_\beta}(t)$
CR_3 $\mathrm{neutral}(t) \wedge (\forall t'.\, t \rightarrow_\beta t' \Rightarrow \mathrm{RED}_U(t')) \Rightarrow \mathrm{RED}_U(t)$

$CR_{1,3}$ *are proved together by induction on U.*

Proof. If U is a type variable, CR_1 is a tautology. The only remaining case is $U = V \rightarrow W$.

$$
\begin{aligned}
& \mathrm{RED}_{V\rightarrow W}(t) & \\
\Leftrightarrow\ & \forall u.\, \mathrm{RED}_V(u) \Rightarrow \mathrm{RED}_W(t\,u) & \text{(definition of RED)} \\
\Rightarrow\ & \mathrm{RED}_V(x) \Rightarrow \mathrm{RED}_W(t\,x) & \text{(x is a fresh variable)} \\
\Rightarrow\ & \mathrm{RED}_W(t\,x) & \text{(I.H. of CR}_3\text{)} \\
\Rightarrow\ & \mathrm{SN}_{\rightarrow_\beta}(t\,x) & \text{(I.H. of CR}_1\text{)} \\
\Rightarrow\ & \mathrm{SN}_{\rightarrow_\beta}(t) & \text{(basic property of SN)}
\end{aligned}
$$

A Proof of CR_1
in Untyped Settings

CR_1 $RED_U(t) \Rightarrow SN_{\rightarrow_\beta}(t)$
CR_3 $\text{neutral}(t) \land (\forall t'. t \rightarrow_\beta t' \Rightarrow RED_U(t')) \Rightarrow RED_U(t)$

$CR_{1,3}$ *are proved together by induction on* U*.*

Proof. If U is a type variable, CR_1 is a tautology. The only remaining case is $U = V \rightarrow W$.

$RED_{V \rightarrow W}(t)$	
$\Leftrightarrow \forall u. RED_V(u) \Rightarrow RED_W(t\,u)$	(definition of RED)
$\Rightarrow RED_V(x) \Rightarrow RED_W(t\,x)$	(x is a fresh variable)
$\Rightarrow RED_W(t\,x)$	(I.H. of CR_3)
$\Rightarrow SN_{\rightarrow_\beta}(t\,x)$	(I.H. of CR_1)
$\Rightarrow SN_{\rightarrow_\beta}(t)$	(basic property of SN)

In typed settings, a term of type V is not always existing.

Solutions

There are 2 ways to solve this issue:

- Construct finite set of types by traversing proof tree of $\Gamma \vdash t : U$, and add it to Γ with fresh variables.
- Redefine RED_U as a Kripke logical predicate.

Solution 1: Proof Tree Traversal

$$\mathcal{T}\left(\frac{\Gamma(x) = U}{\Gamma \vdash x : U}\right) = \mathcal{T}'(U)$$

$$\mathcal{T}\left(\frac{P_1 \qquad P_2}{\Gamma \vdash t\, u : V}\right) = \mathcal{T}'(V) \cup \mathcal{T}(P_1) \cup \mathcal{T}(P_2)$$

$$\mathcal{T}\left(\frac{P_1}{\Gamma \vdash \lambda x : U.t : U \to V}\right) = \mathcal{T}'(U \to V) \cup \mathcal{T}(P_1)$$

$$\mathcal{T}'(X) = \emptyset$$

$$\mathcal{T}'(U \to V) = \{U\} \cup \mathcal{T}'(U) \cup \mathcal{T}'(V)$$

Solution 1: Proof Tree Traversal

It is possible to prove the following $\mathrm{CR}_{1,2,3}$ in a similar method.

CR_1 $(\forall V \in \mathcal{T}'(U).\, V \in \Gamma) \wedge \mathrm{RED}^{\Gamma}_{U}(t) \Rightarrow \mathrm{SN}_{\rightarrow_\beta}(t)$

CR_2 $t \rightarrow_\beta t' \wedge \mathrm{RED}^{\Gamma}_{U}(t) \Rightarrow \mathrm{RED}^{\Gamma}_{U}(t')$

CR_3 $(\forall V \in \mathcal{T}'(U).\, V \in \Gamma) \wedge \Gamma \vdash t : U$
$\wedge\, \mathrm{neutral}(t) \wedge (\forall t'.\, t \rightarrow_\beta t' \Rightarrow \mathrm{RED}^{\Gamma}_{U}(t')) \Rightarrow \mathrm{RED}^{\Gamma}_{U}(t)$

Solution 1: Proof Tree Traversal

It is possible to prove the following $\text{CR}_{1,2,3}$ in a similar method.

CR_1 $(\forall V \in \mathcal{T}'(U).\, V \in \Gamma) \wedge \text{RED}^{\Gamma}_{U}(t) \Rightarrow \text{SN}_{\rightarrow_\beta}(t)$

CR_2 $t \rightarrow_\beta t' \wedge \text{RED}^{\Gamma}_{U}(t) \Rightarrow \text{RED}^{\Gamma}_{U}(t')$

CR_3 $(\forall V \in \mathcal{T}'(U).\, V \in \Gamma) \wedge \Gamma \vdash t : U$
$\wedge\, \text{neutral}(t) \wedge (\forall t'.\, t \rightarrow_\beta t' \Rightarrow \text{RED}^{\Gamma}_{U}(t')) \Rightarrow \text{RED}^{\Gamma}_{U}(t)$

Solution 2: Kripke Logical Predicate

$$
\begin{aligned}
\mathrm{RED}'_X(\Gamma, t) &\stackrel{\text{def}}{\iff} \mathrm{SN}_{\to_\beta}(t) \\
\mathrm{RED}'_{U \to V}(\Gamma, t) &\stackrel{\text{def}}{\iff} \forall \Delta, u.\, \Gamma \lessdot \Delta \land \mathrm{RED}_U(\Delta, u) \Rightarrow \mathrm{RED}_V(\Delta, t\, u) \\
\mathrm{RED}_U(\Gamma, t) &\stackrel{\text{def}}{\iff} \Gamma \vdash t : U \land \mathrm{RED}'_U(\Gamma, t)
\end{aligned}
$$

Solution 2: Kripke Logical Predicate

$\forall x \in \mathrm{dom}(\Gamma).\, \Gamma(x) = \Delta(x)$

$$
\begin{aligned}
\mathrm{RED}'_X(\Gamma, t) &\overset{\text{def}}{\iff} \mathrm{SN}_{\to_\beta}(t) \\
\mathrm{RED}'_{U \to V}(\Gamma, t) &\overset{\text{def}}{\iff} \forall \Delta, u.\, \Gamma \leq \Delta \wedge \mathrm{RED}_U(\Delta, u) \Rightarrow \mathrm{RED}_V(\Delta, t\, u) \\
\mathrm{RED}_U(\Gamma, t) &\overset{\text{def}}{\iff} \Gamma \vdash t : U \wedge \mathrm{RED}'_U(\Gamma, t)
\end{aligned}
$$

Solution 2: Kripke Logical Predicate

$\forall x \in \mathrm{dom}(\Gamma).\, \Gamma(x) = \Delta(x)$

$$\begin{aligned}
\mathrm{RED}'_X(\Gamma, t) &\overset{\text{def}}{\iff} \mathrm{SN}_{\to_\beta}(t) \\
\mathrm{RED}'_{U \to V}(\Gamma, t) &\overset{\text{def}}{\iff} \forall \Delta, u.\, \Gamma \leq \Delta \land \mathrm{RED}_U(\Delta, u) \Rightarrow \mathrm{RED}_V(\Delta, t\, u) \\
\mathrm{RED}_U(\Gamma, t) &\overset{\text{def}}{\iff} \Gamma \vdash t : U \land \mathrm{RED}'_U(\Gamma, t)
\end{aligned}$$

CR$_1$ $\mathrm{RED}_U(\Gamma, t) \Rightarrow \mathrm{SN}_{\to_\beta}(t)$

CR$_2$ $t \to_\beta t' \land \mathrm{RED}_U(\Gamma, t) \Rightarrow \mathrm{RED}_U(\Gamma, t')$

CR$_3$ $\Gamma \vdash t : U \land \mathrm{neutral}(t) \land (\forall t'.\, t \to_\beta t' \Rightarrow \mathrm{RED}_U(\Gamma, t'))$
$\Rightarrow \mathrm{RED}_U(\Gamma, t)$

$\lambda 2$: System F [Gir72, GTL89]

types and terms:

$$U ::= \dots \mid (\Pi X.\, U)$$

$$t ::= \dots \mid (t\, U) \mid (\Lambda X.\, t)$$

additional typing rules:

$$\frac{\Gamma \vdash t : \Pi X.\, U}{\Gamma \vdash t\, V : U[X := V]}$$

$$\frac{X \notin \Gamma \qquad \Gamma \vdash t : U}{\Gamma \vdash \Lambda X.\, t : \Pi X.\, U}$$

additional reduction rules:

$$(\Lambda X.\, t)\, U \to_\beta t[X := U]$$

$$\frac{t_1 \to_\beta t_2}{t_1\, U \to_\beta t_2\, U}$$

$$\frac{t_1 \to_\beta t_2}{\Lambda X.\, t_1 \to_\beta \Lambda X.\, t_2}$$

Strong Normalization Proofs for System F

It is impossible to define a reducibility for System F directly.

Strong Normalization Proofs for System F

It is impossible to define a reducibility for System F directly.
Proof outline of the part 1:

1. Define the **reducibility candidates**. This is a (type indexed) family of terms, and defined by conditions like $\mathrm{CR}_{1,2,3}$.
2. Define the **reducibility with parameters**. This corresponds to the reducibility of $\lambda^{\to}$.
3. Prove that the reducibility with parameters is a reducibility candidate.

Untyped Reducibility Candidates

Set of terms $\mathcal{R}$ is **reducibility candidate** if and only if

CR_1 $\mathcal{R}(t) \Rightarrow \mathrm{SN}(t)$

CR_2 $t \rightarrow_\beta t' \wedge \mathcal{R}(t) \Rightarrow \mathcal{R}(t)$

CR_3 $\mathrm{neutral}(t) \wedge (\forall t'.t \rightarrow_\beta t' \Rightarrow \mathcal{R}(t')) \Rightarrow \mathcal{R}(t).$

Untyped Reducibility Candidates

Set of terms $\mathcal{R}$ is **reducibility candidate** if and only if

CR$_1$ $\mathcal{R}(t) \Rightarrow \mathrm{SN}(t)$

CR$_2$ $t \rightarrow_\beta t' \wedge \mathcal{R}(t) \Rightarrow \mathcal{R}(t)$

CR$_3$ $\mathrm{neutral}(t) \wedge (\forall t'.t \rightarrow_\beta t' \Rightarrow \mathcal{R}(t')) \Rightarrow \mathcal{R}(t)$.

↑ t is not of the form $\lambda x.\, u$ or $\Lambda X.\, u$.

Untyped Reducibility Candidates

Set of terms $\mathcal{R}$ is **reducibility candidate** if and only if

CR$_1$ $\mathcal{R}(t) \Rightarrow \mathrm{SN}(t)$

CR$_2$ $t \rightarrow_\beta t' \wedge \mathcal{R}(t) \Rightarrow \mathcal{R}(t)$

CR$_3$ $\mathrm{neutral}(t) \wedge (\forall t'. t \rightarrow_\beta t' \Rightarrow \mathcal{R}(t')) \Rightarrow \mathcal{R}(t)$.

↳ t is not of the form $\lambda x.\, u$ or $\Lambda X.\, u$.

For example, SN is a reducibility candidate.

Untyped Reducibility with Parameters

$$\mathrm{RED}_Y[\overline{X} := \overline{\mathcal{R}}](t) \overset{\text{def}}{\Longleftrightarrow} \begin{cases} \overline{\mathcal{R}}_i(t) & \text{if } \; Y = \overline{X}_i \\ \mathrm{SN}(t) & \text{if } \; Y \notin \overline{X} \end{cases}$$

$$\mathrm{RED}_{U \to V}[\overline{X} := \overline{\mathcal{R}}](t) \overset{\text{def}}{\Longleftrightarrow} \forall u.\, \mathrm{RED}_U[\overline{X} := \overline{\mathcal{R}}](u) \Rightarrow \mathrm{RED}_V[\overline{X} := \overline{\mathcal{R}}](t\, u)$$

$$\mathrm{RED}_{\Pi Y.\, U}[\overline{X} := \overline{\mathcal{R}}](t) \overset{\text{def}}{\Longleftrightarrow} \forall V, \mathcal{S}.\, \mathrm{RC}(\mathcal{S}) \Rightarrow \mathrm{RED}_U[Y, \overline{X} := \mathcal{S}, \overline{\mathcal{R}}](t\, V)$$

Untyped Reducibility with Parameters

$$\mathrm{RED}_Y[\overline{X} := \overline{\mathcal{R}}](t) \overset{\text{def}}{\iff} \begin{cases} \overline{\mathcal{R}}_i(t) & \text{if } Y = \overline{X}_i \\ \mathrm{SN}(t) & \text{if } Y \notin \overline{X} \end{cases}$$

$$\mathrm{RED}_{U \to V}[\overline{X} := \overline{\mathcal{R}}](t) \overset{\text{def}}{\iff} \forall u.\, \mathrm{RED}_U[\overline{X} := \overline{\mathcal{R}}](u) \Rightarrow \mathrm{RED}_V[\overline{X} := \overline{\mathcal{R}}](t\,u)$$

$$\mathrm{RED}_{\Pi Y.\,U}[\overline{X} := \overline{\mathcal{R}}](t) \overset{\text{def}}{\iff} \forall V, \mathcal{S}.\, \mathrm{RC}(\mathcal{S}) \Rightarrow \mathrm{RED}_U[Y, \overline{X} := \mathcal{S}, \overline{\mathcal{R}}](t\,V)$$

Lemma If $\overline{\mathcal{R}}$ is a sequence of reducibility candidates, $\mathrm{RED}_U[\overline{X} := \overline{\mathcal{R}}]$ is a reducibility candidate.

Typed Reducibility Candidates 1 [Hur10]

Set of terms $\mathcal{R}$ is reducibility candidate of Γ, U if and only if

CR$_1$ $\#\Gamma \vdash t : U \wedge \mathcal{R}(t) \Rightarrow \mathrm{SN}(t)$

CR$_2$ $\#\Gamma \vdash t : U \wedge t \rightarrow_\beta t' \wedge \mathcal{R}(t) \Rightarrow \mathcal{R}(t')$

CR$_3$ $\#\Gamma \vdash t : U \wedge \mathrm{neutral}(t) \wedge (\forall t'.t \rightarrow_\beta t' \Rightarrow \mathcal{R}(t')) \Rightarrow \mathcal{R}(t).$

where $\#\Gamma = \{b : \Pi X.\, X\} + \Gamma$

Typed Reducibility Candidates 1 [Hur10]

Set of terms $\mathcal{R}$ is reducibility candidate of Γ, U if and only if

CR$_1$ $\#\Gamma \vdash t : U \wedge \mathcal{R}(t) \Rightarrow \mathrm{SN}(t)$

CR$_2$ $\#\Gamma \vdash t : U \wedge t \rightarrow_\beta t' \wedge \mathcal{R}(t) \Rightarrow \mathcal{R}(t')$

CR$_3$ $\#\Gamma \vdash t : U \wedge \mathrm{neutral}(t) \wedge (\forall t'.t \rightarrow_\beta t' \Rightarrow \mathcal{R}(t')) \Rightarrow \mathcal{R}(t)$.

where $\#\Gamma = \{b : \Pi X.\, X\} + \Gamma$

↑

$\#\Gamma \vdash b\,U : U$

$b\,U$ is neutral and normal

Typed Reducibility with Parameters 1

$$\mathrm{RED}^{\Gamma}_{Y}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t) \overset{\text{def}}{\iff} \begin{cases} \overline{\mathcal{R}}_i(t) & \text{if } \ Y = \overline{X}_i \\ \mathrm{SN}(t) & \text{if } \ Y \notin \overline{X} \end{cases}$$

$$\mathrm{RED}^{\Gamma}_{V \to W}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t) \overset{\text{def}}{\iff} \forall u.\, \#\Gamma \vdash u : V[\overline{X} := \overline{U}] \Rightarrow \mathrm{RED}^{\Gamma}_{V}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](u) \Rightarrow \mathrm{RED}^{\Gamma}_{W}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t\,u)$$

$$\mathrm{RED}^{\Gamma}_{\Pi Y.\,V}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t) \overset{\text{def}}{\iff} \forall W, \mathcal{S}.\, \mathrm{RC}^{\Gamma}_{W}(\mathcal{S}) \Rightarrow \mathrm{RED}^{\Gamma}_{V}[Y, \overline{X} := \mathcal{S}, \overline{\mathcal{R}} : W, \overline{U}](t\,W)$$

Typed Reducibility with Parameters 1

$$\mathrm{RED}^{\Gamma}_{Y}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t) \stackrel{\text{def}}{\iff} \begin{cases} \overline{\mathcal{R}}_i(t) & \text{if } Y = \overline{X}_i \\ \mathrm{SN}(t) & \text{if } Y \notin \overline{X} \end{cases}$$

$$\mathrm{RED}^{\Gamma}_{V \to W}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t) \stackrel{\text{def}}{\iff} \forall u.\, \#\Gamma \vdash u : V[\overline{X} := \overline{U}] \Rightarrow \mathrm{RED}^{\Gamma}_{V}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](u) \Rightarrow \mathrm{RED}^{\Gamma}_{W}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t\,u)$$

$$\mathrm{RED}^{\Gamma}_{\Pi Y.\,V}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t) \stackrel{\text{def}}{\iff} \forall W, \mathcal{S}.\, \mathrm{RC}^{\Gamma}_{W}(\mathcal{S}) \Rightarrow \mathrm{RED}^{\Gamma}_{V}[Y, \overline{X} := \mathcal{S}, \overline{\mathcal{R}} : W, \overline{U}](t\,W)$$

Lemma If $\overline{\mathcal{R}}_i$ is a reducibility candidate of $\Gamma, \overline{U}_i$ for all $i \leq |\overline{X}|$, $\mathrm{RED}^{\Gamma}_{V}[\overline{X} := \overline{\mathcal{R}} : \overline{U}]$ is a reducibility candidate of $\Gamma, V[\overline{X} := \overline{U}]$.

Typed Reducibility Candidates 2 [Gal89]

Set of pairs of type environment and term $\mathcal{R}$ is reducibility candidate of type U if and only if

CR$_{\text{typed}}$ $\mathcal{R}(\Gamma, t) \Rightarrow \Gamma \vdash t : U$

CR$_0$ $\Gamma \leq \Delta \wedge \mathcal{R}(\Gamma, t) \Rightarrow \mathcal{R}(\Delta, t)$

CR$_1$ $\mathcal{R}(t) \Rightarrow \mathrm{SN}(t)$

CR$_2$ $t \rightarrow_\beta t' \wedge \mathcal{R}(t) \Rightarrow \mathcal{R}(t')$

CR$_3$ $\Gamma \vdash t : U \wedge \mathrm{neutral}(t) \wedge (\forall t'.t \rightarrow_\beta t' \Rightarrow \mathcal{R}(t')) \Rightarrow \mathcal{R}(t)$.

Typed Reducibility Candidates 2 [Gal89]

Set of pairs of type environment and term $\mathcal{R}$ is reducibility candidate of type U if and only if

CR$_{\text{typed}}$ $\mathcal{R}(\Gamma, t) \Rightarrow \Gamma \vdash t : U$

CR$_0$ $\Gamma \leq \Delta \wedge \mathcal{R}(\Gamma, t) \Rightarrow \mathcal{R}(\Delta, t)$

CR$_1$ $\mathcal{R}(t) \Rightarrow \mathrm{SN}(t)$

CR$_2$ $t \rightarrow_\beta t' \wedge \mathcal{R}(t) \Rightarrow \mathcal{R}(t')$

CR$_3$ $\Gamma \vdash t : U \wedge \mathrm{neutral}(t) \wedge (\forall t'.t \rightarrow_\beta t' \Rightarrow \mathcal{R}(t')) \Rightarrow \mathcal{R}(t)$.

Typed Reducibility with Parameters 2

$$\mathrm{RED}_Y[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Gamma, t) \overset{\text{def}}{\iff} \begin{cases} \overline{\mathcal{R}}_i(\Gamma, t) & \text{if } Y = \overline{X}_i \\ \mathrm{SN}'(\Gamma, t) & \text{if } Y \notin \overline{X} \end{cases}$$

$$\begin{aligned} \mathrm{RED}_{V \to W}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Gamma, t) \overset{\text{def}}{\iff} \; & \Gamma \vdash t : V \to W \\ & \wedge (\forall \Delta, u.\, \Gamma \leq \Delta \\ & \qquad \Rightarrow \mathrm{RED}_V[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Delta, u) \\ & \qquad \Rightarrow \mathrm{RED}_W[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Delta, t\, u)) \end{aligned}$$

$$\begin{aligned} \mathrm{RED}_{\Pi Y.\, V}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Gamma, t) \overset{\text{def}}{\iff} \; & \forall W, \mathcal{S}.\, \mathrm{RC}_W(\mathcal{S}) \\ & \qquad \Rightarrow \mathrm{RED}_V[Y, \overline{X} := \mathcal{S}, \overline{\mathcal{R}} : W, \overline{U}](t\, W) \end{aligned}$$

Typed Reducibility with Parameters 2

$$\mathrm{RED}_Y[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Gamma, t) \overset{\text{def}}{\iff} \begin{cases} \overline{\mathcal{R}}_i(\Gamma, t) & \text{if } \ Y = \overline{X}_i \\ \mathrm{SN}'(\Gamma, t) & \text{if } \ Y \notin \overline{X} \end{cases}$$

$$\begin{aligned} \mathrm{RED}_{V \to W}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Gamma, t) \overset{\text{def}}{\iff} \ & \Gamma \vdash t : V \to W \\ & \wedge (\forall \Delta, u.\, \Gamma \leq \Delta \\ & \qquad \Rightarrow \mathrm{RED}_V[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Delta, u) \\ & \qquad \Rightarrow \mathrm{RED}_W[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Delta, t\, u)) \end{aligned}$$

$$\begin{aligned} \mathrm{RED}_{\Pi Y.\, V}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Gamma, t) \overset{\text{def}}{\iff} \ & \forall W, \mathcal{S}.\, \mathrm{RC}_W(\mathcal{S}) \\ & \quad \Rightarrow \mathrm{RED}_V[Y, \overline{X} := \mathcal{S}, \overline{\mathcal{R}} : W, \overline{U}](t\, W) \end{aligned}$$

Lemma If $\overline{\mathcal{R}_i}$ is a reducibility candidate of $\overline{U}_i$ for all $i \leq |\overline{X}|$, $\mathrm{RED}_V[\overline{X} := \overline{\mathcal{R}} : \overline{U}]$ is a reducibility candidate of $V[\overline{X} := \overline{U}]$.

Comparison of the SN Proofs

- SN proofs with typed reducibility requires type preservation lemmas. On the other hand, SN proofs with untyped reducibility are completed without type preservation lemmas. (Untyped proofs are relatively simple.)
- Typed reducibilities are capturing the features of reducible terms.

Conclusion

- We formalized strong normalization proofs with 6 different definitions of the reducibility.
- `$ wc -lc **/*.v`

  ```
  ...
    1808  72327 coq/LC/Debruijn/F.v
     647  24413 coq/LC/Debruijn/STLC.v
  ...
    3746 138149 total
  ```
- https://github.com/pi8027/lambda-calculus

Appendix

λ-Calculus and Representations of Binding

named representation
(name-carrying term)

$$
\begin{aligned}
t ::= {} & x \ (\in \mathbf{Var}) \\
\mid {} & (t\, t) \\
\mid {} & (\lambda x.\, t)
\end{aligned}
$$

de Bruijn representation [dB72]
(nameless terms)

$$
\begin{aligned}
t ::= {} & x \ (\in \mathbb{N}) \\
\mid {} & (t\, t) \\
\mid {} & (\lambda t)
\end{aligned}
$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$
\begin{aligned}
t ::= &\ x \ (\in \mathbf{Var}) \\
\mid &\ (t\, t) \\
\mid &\ (\lambda x.\, t)
\end{aligned}
$$

de Bruijn representation [dB72] (nameless terms)

$$
\begin{aligned}
t ::= &\ x \ (\in \mathbb{N}) \longleftarrow \\
\mid &\ (t\, t) \\
\mid &\ (\lambda t)
\end{aligned}
$$

This number indicates the index of corresponding binder.

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\, t) \mid (\lambda x.\, t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\, t) \mid (\lambda t)$$

← This number indicates the index of corresponding binder.

↑ Nameless terms don't require a variable name in binding positions.

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\,t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$\lambda x.\,\lambda y.\,(\lambda z.\,y\,x\,z)\,x\,a$

$\lambda\lambda(\lambda 1\,2\,0)\,1\,2$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

$$\lambda\lambda(\lambda 1\, 2\, 0)\, 1\, 2$$

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

$$\lambda\lambda(\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \;\; (\in \mathbf{Var}) \;\mid\; (t\,t) \;\mid\; (\lambda x.\, t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \;\; (\in \mathbb{N}) \;\mid\; (t\,t) \;\mid\; (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$

$\lambda\lambda(\lambda 1\, 2\, 0)\, 1\, 2$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\, t) \mid (\lambda x.\, t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\, t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$

$\lambda\lambda(\lambda 1\, 2\, 0)\, 1\, 2$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \; (\in \mathbf{Var}) \mid (t\, t) \mid (\lambda x.\, t)$$

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \; (\in \mathbb{N}) \mid (t\, t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

$$\lambda\lambda(\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$t ::= x \ (\in \mathbf{Var})$
$\quad \mid \ (t\,t)$
$\quad \mid \ (\lambda x.\, t)$

de Bruijn representation [dB72] (nameless terms)

$t ::= x \ (\in \mathbb{N})$ ⟵ This number indicates the index of corresponding binder.
$\quad \mid \ (t\,t)$
$\quad \mid \ (\lambda t)$

↑ Nameless terms don't require a variable name in binding positions.

examples:

$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$

$\lambda\lambda(\lambda 1\, 2\, 0)\, 1\, 2$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \; (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

examples:

$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \; (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

$$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

$$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

$$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

← This number indicates the index of corresponding binder.

↑ Nameless terms don't require a variable name in binding positions.

$$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\, t) \mid (\lambda x.\, t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\, t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

$$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$
\begin{aligned}
t ::= {} & x \quad (\in \mathbf{Var}) \\
\mid {} & (t\,t) \\
\mid {} & (\lambda x.\, t)
\end{aligned}
$$

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\; x\, a$$

de Bruijn representation [dB72] (nameless terms)

$$
\begin{aligned}
t ::= {} & x \quad (\in \mathbb{N}) \\
\mid {} & (t\,t) \\
\mid {} & (\lambda t)
\end{aligned}
$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

$$\lambda \lambda (\lambda 1\, 2\, 0)\; 1\, 2$$

1

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\, t) \mid (\lambda x.\, t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\, t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

$$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$$

2

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

← This number indicates the index of corresponding binder.

↑ Nameless terms don't require a variable name in binding positions.

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

$$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$$

0

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

$$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

$$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation
(name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\, t) \mid (\lambda x.\, t)$$

de Bruijn representation [dB72]
(nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\, t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

free occurrence of a

$$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \textbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

free occurrence of a

$$\lambda\lambda(\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\, t) \mid (\lambda x.\, t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\, t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$$

free occurrence of a

0

$$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$$

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \; (\in \mathbf{Var}) \mid (t\, t) \mid (\lambda x.\, t)$$

examples:

$$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\; x\; a$$

free occurrence of a

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \; (\in \mathbb{N}) \mid (t\, t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

1

$$\lambda \lambda (\lambda 1\, 2\, 0)\; 1\; 2$$

λ-Calculus and Representations of Binding

named representation
(name-carrying term)

$$t ::= x \;(\in \mathbf{Var}) \;\mid\; (t\,t) \;\mid\; (\lambda x.\, t)$$

de Bruijn representation [dB72]
(nameless terms)

$$t ::= x \;(\in \mathbb{N}) \;\mid\; (t\,t) \;\mid\; (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$

free occurrence of a

$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$

free occurrence **of 0**

λ-Calculus and Representations of Binding

named representation (name-carrying term)

$$t ::= x \ (\in \mathbf{Var}) \mid (t\,t) \mid (\lambda x.\, t)$$

de Bruijn representation [dB72] (nameless terms)

$$t ::= x \ (\in \mathbb{N}) \mid (t\,t) \mid (\lambda t)$$

This number indicates the index of corresponding binder.

Nameless terms don't require a variable name in binding positions.

examples:

$\lambda x.\, \lambda y.\, (\lambda z.\, y\, x\, z)\, x\, a$

free occurrence of a

$\lambda \lambda (\lambda 1\, 2\, 0)\, 1\, 2$

free occurrence **of 0**

We use the de Bruijn representation for our formalization.

(Informal) Definition of Reduction

$$(\lambda x.\, t)\, u \rightarrow_\beta t[x := u]$$

$$\frac{t_1 \rightarrow_\beta t_2}{t_1\, u \rightarrow_\beta t_2\, u} \qquad \frac{u_1 \rightarrow_\beta u_2}{t\, u_1 \rightarrow_\beta t\, u_2} \qquad \frac{t \rightarrow_\beta t'}{\lambda x.\, t \rightarrow_\beta \lambda x.\, t'}$$

(Informal) Definition of Reduction

substituting u for every free occurrence of x in t

$$(\lambda x.\, t)\, u \to_\beta t[x := u]$$

$$\frac{t_1 \to_\beta t_2}{t_1\, u \to_\beta t_2\, u} \qquad \frac{u_1 \to_\beta u_2}{t\, u_1 \to_\beta t\, u_2} \qquad \frac{t \to_\beta t'}{\lambda x.\, t \to_\beta \lambda x.\, t'}$$

Capture

$$(\lambda y.\, \lambda x.\, y)\, x$$

Capture

free occurrence of x

$$(\lambda y.\, \lambda x.\, y)\, x$$

Capture

free occurrence of x

$$(\lambda y.\, \lambda x.\, y)\; x$$

$$\to_\beta (\lambda x.\, y)[y := x]$$

Capture

free occurrence of x

$$
\begin{aligned}
&(\lambda y.\, \lambda x.\, y)\, x \\
\rightarrow_\beta\ &(\lambda x.\, y)[y := x] \\
=\ &\lambda x.\, x
\end{aligned}
$$

Capture

free occurrence of x

$$(\lambda y.\, \lambda x.\, y)\; x$$

$$\rightarrow_\beta (\lambda x.\, y)[y := x]$$

$$= \lambda x.\, x$$

x is bound variable. (captured)

Capture

free occurrence of x

$$(\lambda y.\, \lambda x.\, y)\, x$$

$$\rightarrow_\beta (\lambda x.\, y)[y := x]$$

$$= \lambda x.\, x$$

x is bound variable. (captured)

In the named representation, it is necessary to use a restricted reduction rule and the α-equivalence relation or capture-avoiding substitutions.

Comparison of the Representations

- named representation
  - Non essential part of the proofs relevant to bindings are large.
  - Most part of proofs are required conditions relevant to free variables such as $x \notin \mathrm{FV}(t)$, $\mathrm{FV}(t) \cap \mathrm{FV}(t') = \emptyset$, etc.
- de Bruijn representation
  - We can concentrate on the essential part of the proofs.
  - Conditions relevant to free variables are replaced by inequality between indices.

Comparison of the Representations

- named representation
  - Non essential part of the proofs relevant to bindings are large.
  - Most part of proofs are required conditions relevant to free variables such as $x \notin \mathrm{FV}(t)$, $\mathrm{FV}(t) \cap \mathrm{FV}(t') = \emptyset$, etc.
- de Bruijn representation
  - We can concentrate on the essential part of the proofs.
  - Conditions relevant to free variables are replaced by inequality between indices.

- The set of nameless terms corresponds to the quotient set of the named terms by α-equivalence relation.

Strong Normalization Property

The set of strongly normalizable terms $\mathrm{SN}_{\rightsquigarrow} \subseteq A$ can be defined by following axioms.

SN-Intro $\forall x \in A. (\forall y \in A. x \rightsquigarrow y \Rightarrow \mathrm{SN}_{\rightsquigarrow}(y)) \Rightarrow \mathrm{SN}_{\rightsquigarrow}(x)$

SN-Elim $\forall P \subseteq A. (\forall x \in A. (\forall y \in A. x \rightsquigarrow y \Rightarrow \mathrm{SN}_{\rightsquigarrow}(y) \wedge P(y)) \Rightarrow P(x)) \Rightarrow \mathrm{SN}_{\rightsquigarrow} \subseteq P$

Strong Normalization Property

The set of strongly normalizable terms $\mathrm{SN}_{\leadsto} \subseteq A$ can be defined by following axioms.

SN-Intro	$\forall x \in A.\,(\forall y \in A.\, x \leadsto y \Rightarrow \mathrm{SN}_{\leadsto}(y)) \Rightarrow \mathrm{SN}_{\leadsto}(x)$
SN-Elim	$\forall P \subseteq A.\,(\forall x \in A.\,(\forall y \in A.\, x \leadsto y \Rightarrow \mathrm{SN}_{\leadsto}(y) \wedge P(y)) \Rightarrow P(x)) \Rightarrow \mathrm{SN}_{\leadsto} \subseteq P$

In the Coq standard library, the strong normalization property is defined as a inductive predicate `Acc`. Constructor and induction principle of `Acc` correspond to SN-Intro and SN-Elim.

Nicolaas Govert de Bruijn.
Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem.
Indagationes Mathematicae, 75(5):381–392, 1972.

Jean H. Gallier.
On Girard's "candidats de reductibilité".
In *Logic and Computer Science*. Academic Press, 1989.

Jean-Yves Girard.
Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur.
PhD thesis, Université de Paris 7, 1972.

Jean-Yves Girard, Paul Taylor, and Yves Lafont.
Proofs and Types.
Cambridge University Press, 1989.

Chung-Kil Hur.
Heq : a Coq library for heterogeneous equality, 2010.
URL: http://sf.snu.ac.kr/gil.hur/Heq/.

Kazuhiko Sakaguchi.
A formalization of typed and untyped λ-calculi in SSReflect-Coq and Agda2, 2011-2015.
URL: https://github.com/pi8027/lambda-calculus.