型付きλ計算の強正規化定理の形式化

坂口和彦

2015/2/13 卒業研究発表会

概要

研究:

- 型付き λ 計算の強正規化定理の証明を Coq で形式化

概要

研究:

- 型付き λ 計算の強正規化定理の証明を Coq で形式化

成果:

- λ 計算の形式化に不可欠な「基盤」となる補題に対する自動証明法を考案
- 単純型付き λ 計算と System F の強正規化定理を形式化
- Girard による証明 [GTL89] に誤りを発見

概要

研究:

- 型付き λ 計算の強正規化定理の証明を Coq で形式化

成果:

- λ 計算の形式化に不可欠な「基盤」となる補題に対する自動証明法を考案
- 単純型付き λ 計算と System F の強正規化定理を形式化
- Girard による証明 [GTL89] に誤りを発見

Coq

- 計算機上で証明を記述するためのソフトウェア
- プログラム検証などにも使われている

Coq

- 計算機上で証明を記述するためのソフトウェア
- プログラム検証などにも使われている

応用例:

- 証明付き C コンパイラ
- 四色定理の形式的証明

λ計算

関数を基本とする計算モデル

項: $t ::= x \mid t\,u \mid \lambda x.\, t$

λ計算
関数を基本とする計算モデル

項: $t ::= x \mid t\,u \mid \lambda x.\,t$

$\lambda x.\,t$ → 引数を x で取って t を返す関数

λ計算
関数を基本とする計算モデル

項: $t ::= x \mid t\,u \mid \lambda x.\,t$

$t\,u$: 関数 t に値 u を与えて得られる結果

$\lambda x.\,t$: 引数を x で取って t を返す関数

λ計算
関数を基本とする計算モデル

項: $t ::= x \mid t\,u \mid \lambda x.\,t$

関数 t に値 u を与えて得られる結果

引数を x で取って t を返す関数

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

λ計算
関数を基本とする計算モデル

項: $t ::= x \mid t\,u \mid \lambda x.\,t$

$t\,u$: 関数 t に値 u を与えて得られる結果

$\lambda x.\,t$: 引数を x で取って t を返す関数

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

$t[x := u]$: t 中の x を u で置き換えた項
(この置き換えを**代入**と呼ぶ)

λ計算
関数を基本とする計算モデル

項: $t ::= x \mid t\,u \mid \lambda x.\, t$

$t\,u$: 関数 t に値 u を与えて得られる結果

$\lambda x.\, t$: 引数を x で取って t を返す関数

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

$t[x := u]$: t 中の x を u で置き換えた項
(この置き換えを**代入**と呼ぶ)

例: $(\lambda x.\, x\,y)\,(\lambda z.\, z) \to_\beta (\lambda z.\, z)\,y \to_\beta y$

λ計算
関数を基本とする計算モデル

項: $t ::= x \mid t\,u \mid \lambda x.\, t$

$t\,u$: 関数 t に値 u を与えて得られる結果

$\lambda x.\, t$: 引数を x で取って t を返す関数

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

$t[x := u]$: t 中の x を u で置き換えた項
(この置き換えを**代入**と呼ぶ)

例:

$$(\lambda x.\, x\,y)\,(\lambda z.\, z) \to_\beta (\lambda z.\, z)\,y \to_\beta y$$

$$(\lambda x.\, x\,x)\,(\lambda x.\, x\,x) \to_\beta (\lambda x.\, x\,x)\,(\lambda x.\, x\,x) \to_\beta \ldots$$

強正規化定理

- λ 計算では簡約列が無限長になる例が無数に存在する
  - $(\lambda x.\, x\, x)\, (\lambda x.\, x\, x)$
  - $(\lambda x.\, x\, x\, x)\, (\lambda x.\, x\, x\, x)$
  - $(\lambda x.\, f\, (x\, x))\, (\lambda x.\, f\, (x\, x))$

強正規化定理

- λ 計算では簡約列が無限長になる例が無数に存在する
  - $(\lambda x.\, x\, x)\, (\lambda x.\, x\, x)$
  - $(\lambda x.\, x\, x\, x)\, (\lambda x.\, x\, x\, x)$
  - $(\lambda x.\, f\, (x\, x))\, (\lambda x.\, f\, (x\, x))$
- 型によって書ける項の形を制限された λ 計算 (型付き λ 計算) では停止しない計算を排除できる

強正規化定理

- λ 計算では簡約列が無限長になる例が無数に存在する
  - $(\lambda x.\, x\, x)\, (\lambda x.\, x\, x)$
  - $(\lambda x.\, x\, x\, x)\, (\lambda x.\, x\, x\, x)$
  - $(\lambda x.\, f\, (x\, x))\, (\lambda x.\, f\, (x\, x))$
- 型によって書ける項の形を制限された λ 計算 (型付き λ 計算) では停止しない計算を排除できる

Theorem (強正規化定理)

t が型の付く項ならば、t から始まる簡約列は有限長

強正規化定理

- λ 計算では簡約列が無限長になる例が無数に存在する
  - $(\lambda x.\, x\, x)\, (\lambda x.\, x\, x)$
  - $(\lambda x.\, x\, x\, x)\, (\lambda x.\, x\, x\, x)$
  - $(\lambda x.\, f\, (x\, x))\, (\lambda x.\, f\, (x\, x))$
- 型によって書ける項の形を制限された λ 計算 (型付き λ 計算) では停止しない計算を排除できる

Theorem (強正規化定理)

t が型の付く項ならば、t から始まる簡約列は有限長

この定理から導ける事実:

- ML では `let rec` を書かない限り無限ループを記述できない
- 対応する直観主義論理の無矛盾性

名前による表現の問題点

変数の「名前」によって束縛と被束縛の対応付けをするような項の定義は、形式的証明には向かない:

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

名前による表現の問題点

変数の「名前」によって束縛と被束縛の対応付けをするような項の定義は、形式的証明には向かない:

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

名前による表現の問題点

変数の「名前」によって束縛と被束縛の対応付けをするような項の定義は、形式的証明には向かない:

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

名前による表現の問題点

変数の「名前」によって束縛と被束縛の対応付けをするような項の定義は、形式的証明には向かない:

$$
\begin{aligned}
(\lambda y.\, y\, x)[x := y] &\neq \lambda y.\, y[x := y]\, x[x := y] \\
&= \lambda y.\, y\, y \\
&=_{\alpha} (\lambda z.\, z\, x)[x := y] \\
&= \lambda z.\, z\, x[x := y] \\
&= \lambda z.\, z\, y
\end{aligned}
$$

名前による表現の問題点

変数の「名前」によって束縛と被束縛の対応付けをするような項の定義は、形式的証明には向かない:

$$
\begin{aligned}
(\lambda y.\, y\, x)[x := y] &\neq \lambda y.\, y[x := y]\, x[x := y] \\
&= \lambda y.\, y\, y \\
&=_{\alpha} (\lambda z.\, z\, x)[x := y] \\
&= \lambda z.\, z\, x[x := y] \\
&= \lambda z.\, z\, y
\end{aligned}
$$

- 束縛変数の名前を付け替えた項は同じ項として扱う必要がある

名前による表現の問題点

変数の「名前」によって束縛と被束縛の対応付けをするような項の定義は、形式的証明には向かない:

$$
\begin{aligned}
(\lambda y.\, y\, x)[x := y] &\neq \lambda y.\, y[x := y]\, x[x := y] \\
&= \lambda y.\, y\, y \\
&=_{\alpha} (\lambda z.\, z\, x)[x := y] \\
&= \lambda z.\, z\, x[x := y] \\
&= \lambda z.\, z\, y
\end{aligned}
$$

- 束縛変数の名前を付け替えた項は同じ項として扱う必要がある
- 商集合は Coq で扱いづらい対象の 1 つ

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

本研究では λ 項の定義として de Bruijn 表現を用いる:

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

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

本研究では λ 項の定義として de Bruijn 表現を用いる:

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

束縛の位置には変数を書かない

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

例:

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

$$\lambda \quad \lambda \quad (\lambda \qquad)$$

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

例:

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

$$\lambda \quad \lambda \quad (\lambda \qquad)$$

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

例:

$$\lambda x.\,\lambda y.\,(\lambda z.\,\overset{0}{\overbrace{x}}\,y)\,x$$

$$\lambda \quad \lambda \quad (\lambda \qquad)$$

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

例:

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

$$\lambda \quad \lambda \quad (\lambda \qquad)$$

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

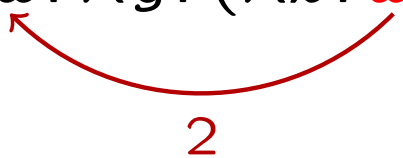
本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

例:

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



$$\lambda \quad \lambda \quad (\lambda \quad 2 \quad)$$

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

例:

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

$$\lambda \quad \lambda \quad (\lambda \quad 2 \quad)$$

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

例:

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

y → 0

$$\lambda \quad \lambda \quad (\lambda \quad 2 \quad)$$

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

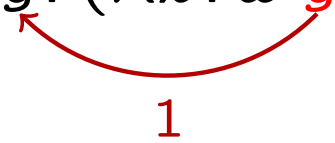
本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

例:

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



$$\lambda \quad \lambda \quad (\lambda \quad 2\ 1)$$

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

例:

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

$$\lambda \quad \lambda \quad (\lambda \quad 2\ 1)$$

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

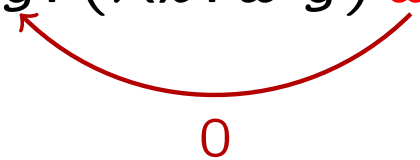
本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

例:

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



$$\lambda \quad \lambda \quad (\lambda \quad 2\ 1)$$

De Bruijn 表現 [dB72]

形式的証明に向いた λ 計算の定義

本研究では λ 項の定義として de Bruijn 表現を用いる:

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

いくつ外側の λ に対応するかを書く　　束縛の位置には変数を書かない

例:

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

$$\lambda \quad \lambda \quad (\lambda \quad 2\ 1)\ 1$$

De Bruijn 表現での代入の定義

(1) 証明向きの定義:

$$y\{x := t\} = \begin{cases} y - 1 & (x < y) \\ t & (x = y) \\ y & (x > y) \end{cases}$$

$$(u\, v)\{x := t\} = u\{x := t\}\, v\{x := t\}$$

$$(\lambda\, u)\{x := t\} = \lambda\, u\{x + 1 := t \uparrow_0^1\}$$

(2) 実装向きの定義:

$$y[x := t] = \begin{cases} y - 1 & (x < y) \\ t \uparrow_0^x & (x = y) \\ y & (x > y) \end{cases}$$

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

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

De Bruijn 表現での代入の定義

(1) 証明向きの定義:

$$y\{x := t\} = \begin{cases} y - 1 & (x < y) \\ t & (x = y) \\ y & (x > y) \end{cases}$$

$$(u\, v)\{x := t\} = u\{x := t\}\, v\{x := t\}$$

$$(\lambda\, u)\{x := t\} = \lambda\, u\{x + 1 := t \uparrow_0^1\}$$

(2) 実装向きの定義:

$$y[x := t] = \begin{cases} y - 1 & (x < y) \\ t \uparrow_0^x & (x = y) \\ y & (x > y) \end{cases}$$

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

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

De Bruijn 表現での代入の定義

(1) 証明向きの定義:

$$y\{x := t\} = \begin{cases} y - 1 & (x < y) \\ t & (x = y) \\ y & (x > y) \end{cases}$$

$$(u\, v)\{x := t\} = u\{x := t\}\, v\{x := t\}$$

$$(\lambda\, u)\{x := t\} = \lambda\, u\{x + 1 := t \uparrow_0^1\}$$

(2) 実装向きの定義:

$$y[x := t] = \begin{cases} y - 1 & (x < y) \\ t \uparrow_0^x & (x = y) \\ y & (x > y) \end{cases}$$

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

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

$t \uparrow_c^d$: 項 t 中の c 以上の自由変数に d を足した項

並列代入

項中の複数の変数を同時に置き換える代入を**並列代入**と呼ぶ:

$$u[x_0, \ldots, x_n := t_0, \ldots, t_n]^1$$

[1] $u[x_0 := t_0] \ldots [x_n := t_n]$ とは異ることに注意

並列代入

項中の複数の変数を同時に置き換える代入を**並列代入**と呼ぶ:

$$u[x_0, \ldots, x_n := t_0, \ldots, t_n]^1$$

- 強正規化定理の証明には並列代入が必要

[1] $u[x_0 := t_0] \ldots [x_n := t_n]$ とは異ることに注意

並列代入

項中の複数の変数を同時に置き換える代入を**並列代入**と呼ぶ:

$$u[x_0, \ldots, x_n := t_0, \ldots, t_n]^1$$

- 強正規化定理の証明には並列代入が必要
- 本研究では代入 (2) を並列に拡張して得られる代入を用いる:

$$u[x := t_0, \ldots, t_n]$$
$$(= u[x, \ldots, x + n := t_0, \ldots, t_n])$$

[1] $u[x_0 := t_0] \ldots [x_n := t_n]$ とは異ることに注意

並列代入

項中の複数の変数を同時に置き換える代入を**並列代入**と呼ぶ:

$$u[x_0, \ldots, x_n := t_0, \ldots, t_n]^1$$

- 強正規化定理の証明には並列代入が必要
- 本研究では代入 (2) を並列に拡張して得られる代入を用いる:

$$u[x := t_0, \ldots, t_n]$$
$$(= u[x, \ldots, x + n := t_0, \ldots, t_n])$$

- 並列代入に限れば (2) の定義が証明向きであることを明らかにした

[1] $u[x_0 := t_0] \ldots [x_n := t_n]$ とは異ることに注意

「基盤」となる補題

$$t \uparrow_n^0 = t \tag{1}$$

$$c \leq c' \leq c + d \Rightarrow t \uparrow_c^d \uparrow_{c'}^{d'} = t \uparrow_c^{d'+d} \tag{2}$$

$$c' \leq c \Rightarrow t \uparrow_c^d \uparrow_{c'}^{d'} = t \uparrow_{c'}^{d'} \uparrow_{d'+c}^{d} \tag{3}$$

$$c \leq n \Rightarrow t[n := \overline{u}] \uparrow_c^d = t \uparrow_c^d [d + n := \overline{u}] \tag{4}$$

$$n \leq c \Rightarrow t[n := \overline{u}] \uparrow_c^d = t \uparrow_{|\overline{u}|+c}^{d} [n := \overline{u} \uparrow_{c-n}^{d}] \tag{5}$$

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

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

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

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

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

where

$$\overline{t} \uparrow_c^d = [t \uparrow_c^d | \ t \leftarrow \overline{t}]$$

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

「基盤」となる補題

$$t \uparrow_n^0 = t \tag{1}$$

$$c \leq c' \leq c + d \Rightarrow t \uparrow_c^d \uparrow_{c'}^{d'} = t \uparrow_c^{d'+d} \tag{2}$$

$$c' \leq c \Rightarrow t \uparrow_c^d \uparrow_{c'}^{d'} = t \uparrow_{c'}^{d'} \uparrow_{d'+c}^{d} \tag{3}$$

$$c \leq n \Rightarrow t[n := \overline{u}] \uparrow_c^d = t \uparrow_c^d [d + n := \overline{u}] \tag{4}$$

$$n \leq c \Rightarrow t[n := \overline{u}] \uparrow_c^d = t \uparrow_{|\overline{u}|+c}^{d} [n := \overline{u} \uparrow_{c-n}^{d}] \tag{5}$$

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

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

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

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

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

where

$$\overline{t} \uparrow_c^d = [t \uparrow_c^d | \ t \leftarrow \overline{t}]$$

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

補題(5)

$$n \leq c \Rightarrow t[n := \overline{u}] \uparrow_c^d = t \uparrow_{|\overline{u}|+c}^d [n := \overline{u} \uparrow_{c-n}^d]$$

補題(5)

$$n \leq c \Rightarrow t[n :=\overline{u}] \uparrow_c^d = t \uparrow_{|\overline{u}|+c}^d [n :=\overline{u} \uparrow_{c-n}^d]$$

補題(5)

$$n \leq c \Rightarrow t[n :=\overline{u}] \uparrow_c^d = t \uparrow_{|\overline{u}|+c}^d [n :=\overline{u} \uparrow_{c-n}^d]$$

```
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.
  - rewrite !nth_default ?size_map /=; elimif_omega.
  - rewrite -shift_shift_distr // nth_map' /=;
      congr shift; apply nth_equal;
      rewrite size_map; elimif_omega.
Qed.
```

証明 - 1
仮定の除去

補題中の仮定を結論側に埋め込む:

$$n \leq c \Rightarrow t[n := \overline{u}] \uparrow_c^d = t \uparrow_{|\overline{u}|+c}^d [n := \overline{u} \uparrow_{c-n}^d]$$

$$\hookrightarrow \quad t[n := \overline{u}] \uparrow_{n+c}^d = t \uparrow_{|\overline{u}|+(n+c)}^d [n := \overline{u} \uparrow_c^d]$$

証明 - 1
仮定の除去

補題中の仮定を結論側に埋め込む:

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

$$\hookrightarrow \quad t[n := \overline{u}] \uparrow^d_{n+c} = t \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c]$$

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

$$\Leftrightarrow \forall m'.\, P[n, n + m']$$

証明 - 2
帰納法の適用

$$t[n := \overline{u}] \uparrow^d_{n+c} = t \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c]$$

証明 - 2
帰納法の適用

$$t[n := \overline{u}] \uparrow^d_{n+c} = t \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c]$$

$$\hookrightarrow \begin{cases} \textbf{1.}\ x[n := \overline{u}] \uparrow^d_{n+c} = x \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c] \\ \textbf{2.}\ t[n := \overline{u}] \uparrow^d_{n+c}\ u[n := \overline{u}] \uparrow^d_{n+c} = \\ \quad t \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c]\ u \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c] \\ \textbf{3.}\ \lambda\, t[n+1 := \overline{u}] \uparrow^d_{n+c+1} = \\ \quad \lambda\, t \uparrow^d_{|\overline{u}|+(n+c)+1} [n+1 := \overline{u} \uparrow^d_c] \end{cases}$$

証明 - 3

関数適用と λ 抽象の場合

2. $t[n := \overline{u}] \uparrow^d_{n+c} \; u[n := \overline{u}] \uparrow^d_{n+c} =$
$t \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c] \; u \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c]$

3. $\lambda\, t[n+1 := \overline{u}] \uparrow^d_{n+c+1} =$
$\lambda\, t \uparrow^d_{|\overline{u}|+(n+c)+1} [n+1 := \overline{u} \uparrow^d_c]$

証明 - 3
関数適用と λ 抽象の場合

2. $t[n := \overline{u}] \uparrow^d_{n+c} \; u[n := \overline{u}] \uparrow^d_{n+c} =$
   $t \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c] \; u \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c]$
3. $\lambda\, t[n + 1 := \overline{u}] \uparrow^d_{n+c+1} =$
   $\lambda\, t \uparrow^d_{|\overline{u}|+(n+c)+1} [n + 1 := \overline{u} \uparrow^d_c]$

- $+1$ の付け替えと帰納法の仮定での書き換えだけで示せる

証明 - 3
関数適用と λ 抽象の場合

2. $t[n := \overline{u}] \uparrow^d_{n+c} \; u[n := \overline{u}] \uparrow^d_{n+c} =$
   $t \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c] \; u \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c]$
3. $\lambda\, t[n+1 := \overline{u}] \uparrow^d_{n+c+1} =$
   $\lambda\, t \uparrow^d_{|\overline{u}|+(n+c)+1} [n+1 := \overline{u} \uparrow^d_c]$

- $+1$ の付け替えと帰納法の仮定での書き換えだけで示せる
- 以下の仮定を追加して Coq の等式に関するソルバ `congruence` を用いて証明

$$\texttt{addSn} : \forall m\, n.\, (m+1) + n = (m+n) + 1$$
$$\texttt{addnS} : \forall m\, n.\, m + (n+1) = (m+n) + 1$$

証明の長さ

Table: 基盤補題の証明の長さ [2]

補題	命題の行数	証明の行数	合計
(1)	1	1	2
(2)	2	1	3
(3)	2	1	3
(4)	2	4	6
(5)	4	6	10
(6)	3	4	7
(7)	4	6	10
(8)	2	4	6
(9)	1	1	2
合計	21	28	49

[2] 1 行の長さは 80 文字を上限とした

まとめ

- 単純型付き λ 計算と System F の強正規化定理を Coq で証明

```
$ wc coq/**/*.v
...
  1786  13363  71540 coq/LC/Debruijn/F.v
   637   4514  24145 coq/LC/Debruijn/STLC.v
   241   1545   8128 coq/LC/Debruijn/Untyped.v
...
  3754  26165 138299 total
```

まとめ

- 単純型付き λ 計算と System F の強正規化定理を Coq で証明

```
$ wc coq/**/*.v
...
  1786   13363   71540 coq/LC/Debruijn/F.v
   637    4514   24145 coq/LC/Debruijn/STLC.v
   241    1545    8128 coq/LC/Debruijn/Untyped.v
...
  3754   26165 138299 total
```

- λ 計算の形式化に必要となる多数の補題を自動的に証明する方法を考案

Appendix

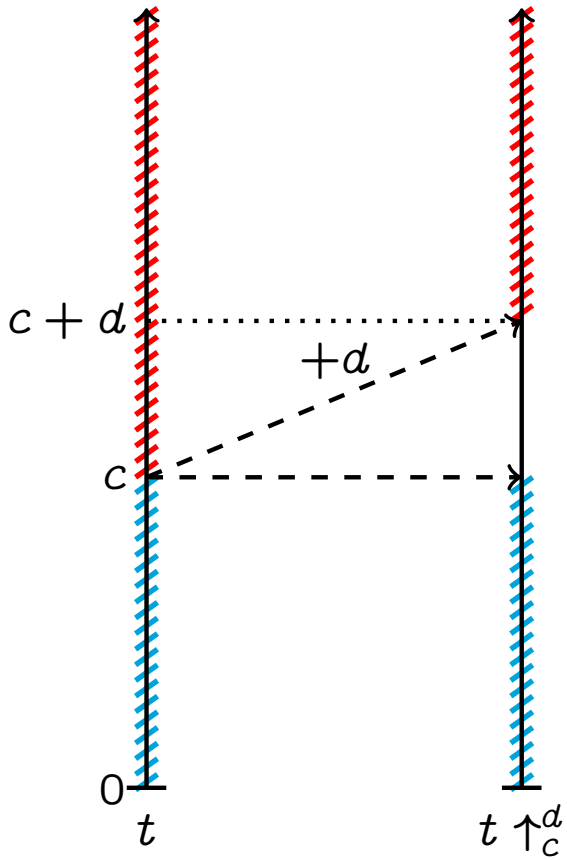
シフトの定義

$$x \uparrow_c^d = \begin{cases} x + d & (c \leq x) \\ x & (x < c) \end{cases}$$

$$(t\, u) \uparrow_c^d = t \uparrow_c^d \; u \uparrow_c^d$$

$$(\lambda\, t) \uparrow_c^d = \lambda\, t \uparrow_{c+1}^d$$

$$t \uparrow^d = t \uparrow_0^d$$



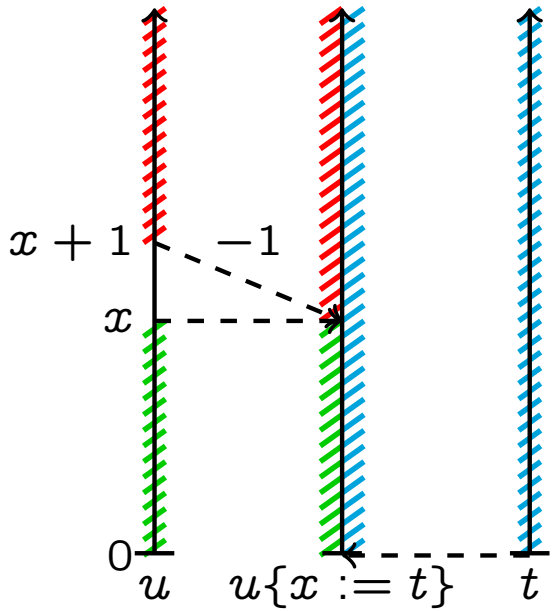
代入の定義

(1)

$$y\{x := t\} = \begin{cases} y - 1 & (x < y) \\ t & (x = y) \\ y & (x > y) \end{cases}$$

$$(u\, v)\{x := t\} = u\{x := t\}\, v\{x := t\}$$

$$(\lambda\, u)\{x := t\} = \lambda\, u\{x + 1 := t \uparrow^1\}$$

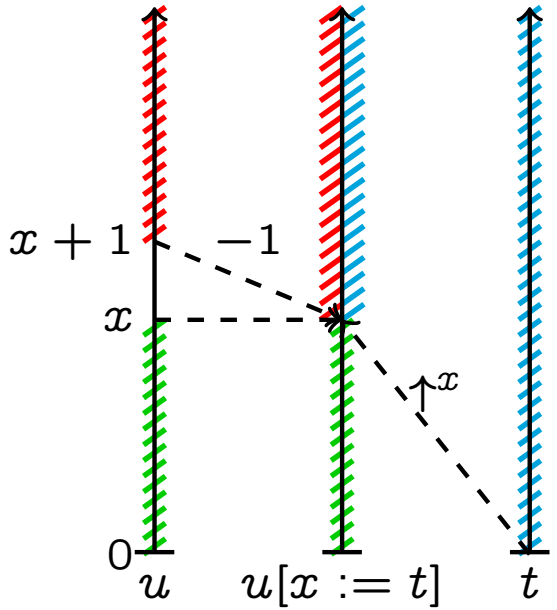


(2)

$$y[x := t] = \begin{cases} y - 1 & (x < y) \\ t \uparrow^x & (x = y) \\ y & (x > y) \end{cases}$$

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

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

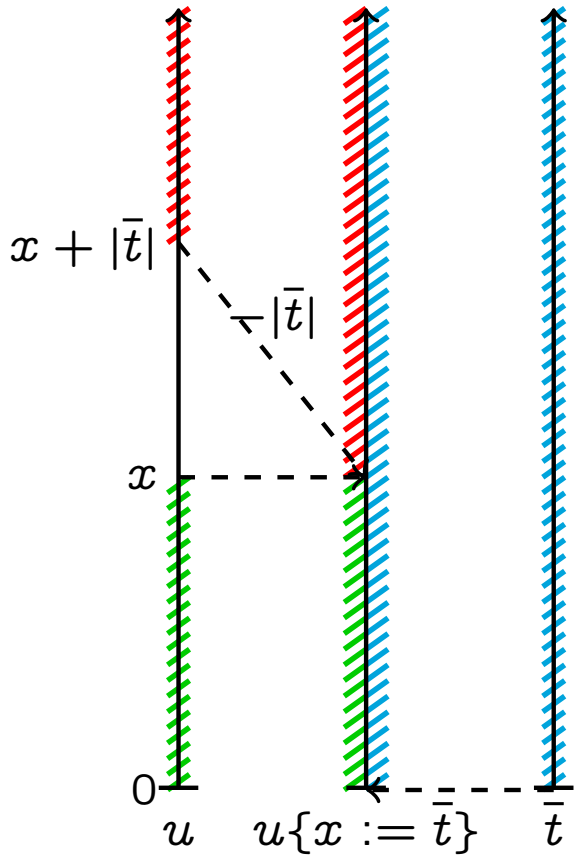


並列代入（1）

$$y\{x := \bar{t}\} = \begin{cases} y - |\bar{t}| & (x + |\bar{t}| \leq y) \\ \bar{t}_{y-x} & (x \leq y < x + |\bar{t}|) \\ y & (y < x) \end{cases}$$

$$(u\, v)\{x := \bar{t}\} = (u\{x := \bar{t}\})\,(v\{x := \bar{t}\})$$

$$(\lambda\, u)\{x := \bar{t}\} = \lambda\,(u\{x + 1 := \bar{t} \uparrow^1\})$$

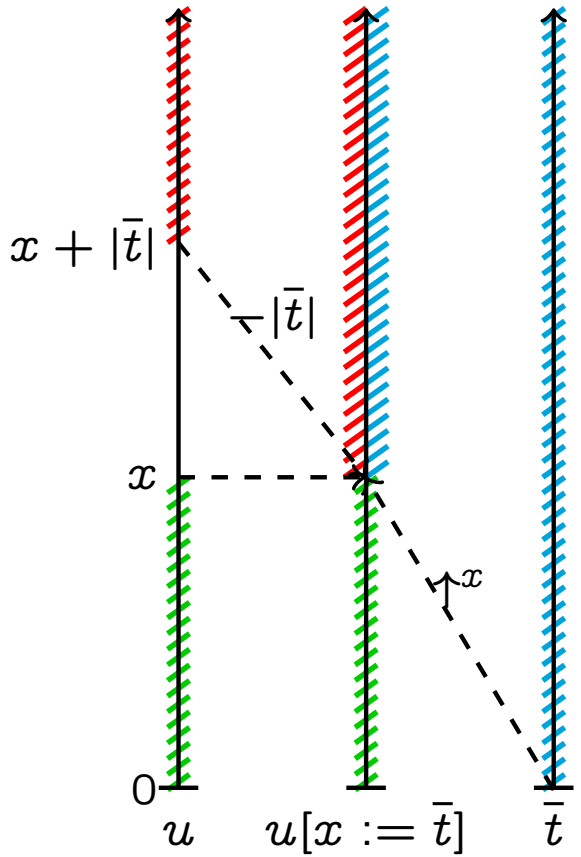


並列代入 (2)

$$y[x := \bar{t}] = \begin{cases} y - |\bar{t}| & (x + |\bar{t}| \leq y) \\ \bar{t}_{y-x} \uparrow^{x} & (x \leq y < x + |\bar{t}|) \\ y & (y < x) \end{cases}$$

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

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



シフトと代入の代数的性質 [Nip01]

(1) の定義:

シフトと代入の代数的性質 [Nip01]

(1) の定義:

- Church-Rosser の定理を示すには以下の補題があれば十分

$$c \leq c' \Rightarrow t \uparrow_c^1 \uparrow_{c'+1}^1 = t \uparrow_{c'}^1 \uparrow_c^1$$

$$x \leq c \Rightarrow t\{x := u\} \uparrow_c^1 = t \uparrow_{c+1}^1 \{x := u \uparrow_c^1\}$$

$$c \leq x \Rightarrow t\{x := u\} \uparrow_c^1 = t \uparrow_c^1 \{x + 1 := u \uparrow_c^1\}$$

$$t \uparrow_x^1 \{x := u\} = t$$

$$x \leq y \Rightarrow t\{y + 1 := v \uparrow_x^1\}\{x := u\{y := v\}\} = t\{x := u\}\{y := v\}$$

シフトと代入の代数的性質 [Nip01]

(1) の定義:

- Church-Rosser の定理を示すには以下の補題があれば十分
- 証明全体を通してシフトの右上の数は 1 以外に出現しない

$$c \leq c' \Rightarrow t \uparrow_c^1 \uparrow_{c'+1}^1 = t \uparrow_{c'}^1 \uparrow_c^1$$

$$x \leq c \Rightarrow t\{x := u\} \uparrow_c^1 = t \uparrow_{c+1}^1 \{x := u \uparrow_c^1\}$$

$$c \leq x \Rightarrow t\{x := u\} \uparrow_c^1 = t \uparrow_c^1 \{x + 1 := u \uparrow_c^1\}$$

$$t \uparrow_x^1 \{x := u\} = t$$

$$x \leq y \Rightarrow t\{y + 1 := v \uparrow_x^1\}\{x := u\{y := v\}\} = t\{x := u\}\{y := v\}$$

シフトと代入の代数的性質 [Nip01]

(1) の定義:

- Church-Rosser の定理を示すには以下の補題があれば十分
- 証明全体を通してシフトの右上の数は 1 以外に出現しない
- 証明も簡潔 (多くの補題の形式的証明は 1 行で済んでいる)

$$c \leq c' \Rightarrow t \uparrow_c^1 \uparrow_{c'+1}^1 = t \uparrow_{c'}^1 \uparrow_c^1$$

$$x \leq c \Rightarrow t\{x := u\} \uparrow_c^1 = t \uparrow_{c+1}^1 \{x := u \uparrow_c^1\}$$

$$c \leq x \Rightarrow t\{x := u\} \uparrow_c^1 = t \uparrow_c^1 \{x + 1 := u \uparrow_c^1\}$$

$$t \uparrow_x^1 \{x := u\} = t$$

$$x \leq y \Rightarrow t\{y + 1 := v \uparrow_x^1\}\{x := u\{y := v\}\} = t\{x := u\}\{y := v\}$$

シフトと代入の代数的性質 [Nip01]

(1) の定義:

- Church-Rosser の定理を示すには以下の補題があれば十分
- 証明全体を通してシフトの右上の数は 1 以外に出現しない
- 証明も簡潔 (多くの補題の形式的証明は 1 行で済んでいる)

$$c \leq c' \Rightarrow t \uparrow_c^1 \uparrow_{c'+1}^1 = t \uparrow_{c'}^1 \uparrow_c^1$$

$$x \leq c \Rightarrow t\{x := u\} \uparrow_c^1 = t \uparrow_{c+1}^1 \{x := u \uparrow_c^1\}$$

$$c \leq x \Rightarrow t\{x := u\} \uparrow_c^1 = t \uparrow_c^1 \{x+1 := u \uparrow_c^1\}$$

$$t \uparrow_x^1 \{x := u\} = t$$

$$x \leq y \Rightarrow t\{y+1 := v \uparrow_x^1\}\{x := u\{y := v\}\} = t\{x := u\}\{y := v\}$$

(2) の定義:

- 直接証明に使うのには向かない
- $u[x := t] = u\{x := t \uparrow^x\}$ で書き換えて証明すると良い

シフトと代入の代数的性質 [Nip01]

(1) の定義:

- Church-Rosser の定理を示すには以下の補題があれば十分
- 証明全体を通してシフトの右上の数は 1 以外に出現しない
- 証明も簡潔 (多くの補題の形式的証明は 1 行で済んでいる)

$$c \leq c' \Rightarrow t \uparrow_c^1 \uparrow_{c'+1}^1 = t \uparrow_{c'}^1 \uparrow_c^1$$

$$x \leq c \Rightarrow t\{x := u\} \uparrow_c^1 = t \uparrow_{c+1}^1 \{x := u \uparrow_c^1\}$$

$$c \leq x \Rightarrow t \uparrow_c^1 \{x := u\} = t \{x + 1 := u \uparrow_c^1\}$$

$$t \uparrow_x^1 \{x := u\} = t$$

$$x \leq y \Rightarrow t\{y + 1 := v \uparrow_x^1\}\{x := u\{y := v\}\} = t\{x := u\}\{y := v\}$$

強正規化定理の証明には不十分

(2) の定義:

- 直接証明に使うのには向かない
- $u[x := t] = u\{x := t \uparrow^x\}$ で書き換えて証明すると良い

証明 - 4
変数の場合

$$\mathbf{1.}\ x[n := \overline{u}] \uparrow^d_{n+c} = x \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c]$$

- 場合分けを全て展開してプレスバーガー算術に帰着させて解く
- 自動証明で解けない部分は手で証明する

証明 - 4
変数の場合

$$\mathbf{1.}\ x[n := \overline{u}] \uparrow^d_{n+c} = x \uparrow^d_{|\overline{u}|+(n+c)} [n := \overline{u} \uparrow^d_c]$$

- 場合分けを全て展開してプレスバーガー算術に帰着させて解く
- 自動証明で解けない部分は手で証明する

場合分けの結果として得られる命題:

- $|\bar{t}| + (n + c) \leq x \wedge n \leq x + d \wedge n \leq x \Rightarrow \mathrm{nth}(x - n - |\bar{t}|, \bar{t}, x - n) \uparrow^n \uparrow^d_{n+c} = \mathrm{nth}(x + d - n - |\bar{t} \uparrow^d_c|, \bar{t} \uparrow^d_c, x + d - n) \uparrow^n$
- $x < |\bar{t}| + (n + c) \wedge n \leq x \wedge n \leq x \Rightarrow \mathrm{nth}(x-n-|\bar{t}|, \bar{t}, x-n) \uparrow^n \uparrow^d_{n+c} = \mathrm{nth}(x-n-|\bar{t} \uparrow^d_c|, \bar{t} \uparrow^d_c, x-n) \uparrow^n$
- $|\bar{t}| + (n + c) \leq x \wedge x + d < n \wedge n \leq x \Rightarrow \mathrm{nth}(x - n - |\bar{t}|, \bar{t}, x - n) \uparrow^n \uparrow^d_{n+c} = x + d$
- $x < |\bar{t}| + (n + c) \wedge x < n \wedge n \leq x \Rightarrow \mathrm{nth}(x - n - |\bar{t}|, \bar{t}, x - n) \uparrow^n \uparrow^d_{n+c} = x$
- ...

Girard の証明の間違い
[GTL89, 14.2.3 Universal application]

Lemma

If $t \in \mathrm{RED}_{\Pi Y.W}[\mathcal{R}/X]$, *then* $tV \in \mathrm{RED}_{W[V/Y]}[\mathcal{R}/X]$ *for every type* V.

Girard の証明の間違い
[GTL89, 14.2.3 Universal application]

Lemma

If $t \in \mathrm{RED}_{\Pi Y.W}[\mathcal{R}/X]$, *then* $t\,V \in \mathrm{RED}_{W[V/Y]}[\mathcal{R}/X]$ *for every type* V.

($\mathcal{R}$ が U の reducibility candidate とすると) $V[U/X]$ が正しい

参考文献 I

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-Yves Girard, Paul Taylor, and Yves Lafont.

Proofs and Types.

Cambridge University Press, 1989.

Tobias Nipkow.

More Church-Rosser proofs.

Journal of Automated Reasoning, 26(1):51–66, 2001.