定理証明支援系を用いた CPS 変換の性質の形式化

薄井千春

プログラム論理研究室

February 13, 2015

本研究の概要

本研究ではプログラムに対するある変換の、重要な性質を
定理証明支援系により形式的に検証した

概要

単純型付きラムダ計算での

- CPS 変換の型保存定理
- CPS 変換の完全性定理の中心的補題

に対して、定理証明支援系 Coq での形式化と形式検証を行った

CPS 変換とはなにか

CPS 変換

プログラム変換の一種で

- **プログラム解析・検証**
- **コンパイラでの利用**
- **コントロールオペレータの導入**

など、幅広い応用を持つ重要なもの

CPS 変換とはなにか

CPS 変換

プログラム変換の一種で

- プログラム解析・検証
- コンパイラでの利用
- コントロールオペレータの導入

など、幅広い応用を持つ重要なもの

その性質の厳密な検証が重要である

CPS 変換とはなにか

CPS 変換

プログラム変換の一種で

- **プログラム解析・検証**
- **コンパイラでの利用**
- **コントロールオペレータの導入**

など、幅広い応用を持つ重要なもの

その性質の厳密な検証が重要である

Example (**階乗関数**)

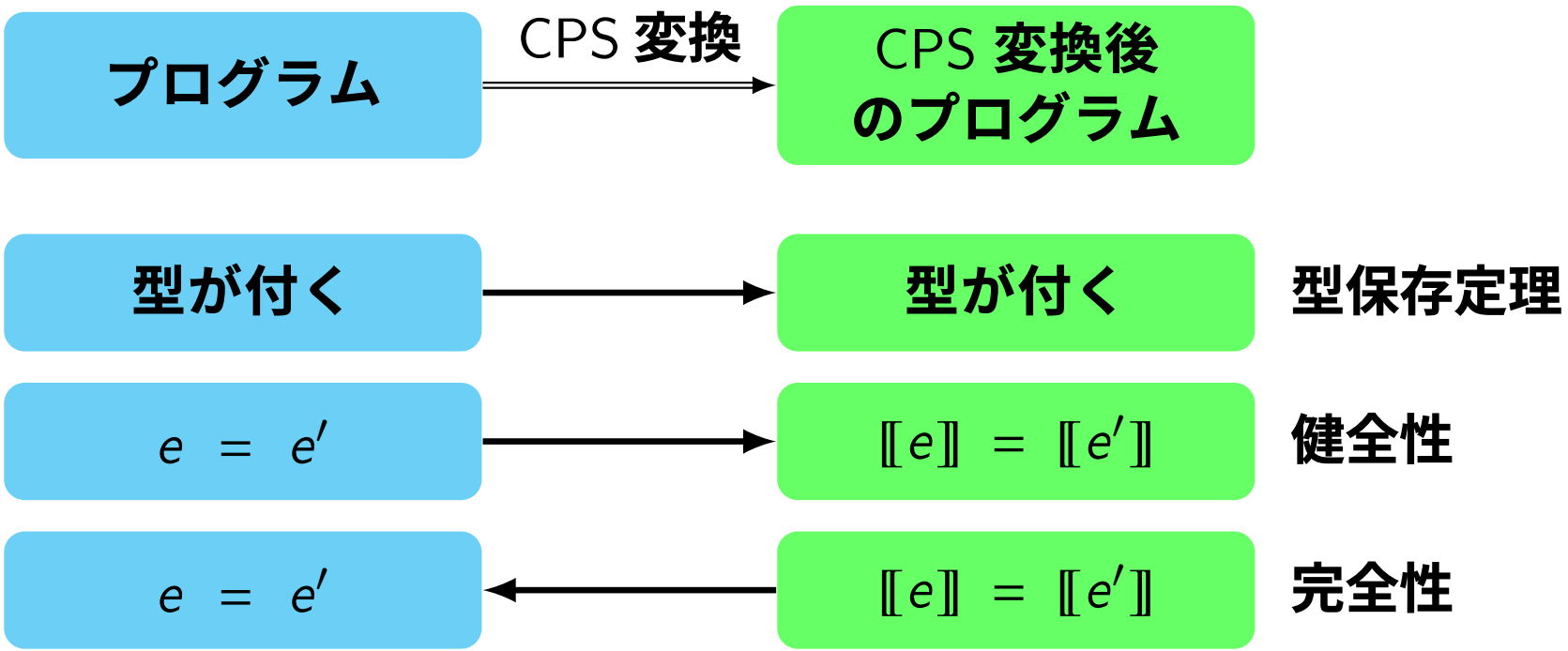
```
let rec fact n = if n = 1 then 1 else n * fact (n-1)
```

↓ CPS **変換**

```
let rec fact_cps n k =     (* 継続変数 k が追加される *)
if n = 1 then (k 1) else fact_cps (n-1) (fun v -> (k (n * v)))
```

CPS 変換の性質

本研究で形式化した定理と関連する定理



定理証明支援系を用いた形式化

なぜ定理証明支援系を用いて形式化するか？

定理証明支援系を用いた形式化

なぜ定理証明支援系を用いて形式化するか？
定理証明支援系を用いて形式化することで

1. 機械的な検証で証明の間違いを見つけ出せる
2. 形式的に検証済みのプログラムの生成ができる
3. 形式化の過程で新たな視点から問題を検討できる

という利点がある

定理証明支援系を用いた形式化

なぜ定理証明支援系を用いて形式化するか？
定理証明支援系を用いて形式化することで

1. 機械的な検証で証明の間違いを見つけ出せる
2. 形式的に検証済みのプログラムの生成ができる
3. 形式化の過程で新たな視点から問題を検討できる

という利点がある
定理証明支援系による形式化の具体例

1. 四色定理（2005 年）
2. CompCert C（2005 年）

定理証明支援系を用いた形式化

なぜ定理証明支援系を用いて形式化するか？
定理証明支援系を用いて形式化することで

1. 機械的な検証で証明の間違いを見つけ出せる
2. 形式的に検証済みのプログラムの生成ができる
3. 形式化の過程で新たな視点から問題を検討できる

という利点がある
定理証明支援系による形式化の具体例

1. 四色定理（2005 年）
2. CompCert C（2005 年）

定理証明支援系による厳密な CPS 変換の性質の検証が可能

変数束縛の形式化手法 1

CPS 変換の性質の検証にはラムダ計算（プログラム言語）を
形式的に表す必要がある

形式化とは

非形式的に表現された式や概念を、厳密に文法と意味が
定められた記号表現に置き替えること

変数束縛の形式化手法 1

CPS 変換の性質の検証にはラムダ計算（プログラム言語）を形式的に表す必要がある

形式化とは

非形式的に表現された式や概念を、厳密に文法と意味が定められた記号表現に置き替えること

ラムダ計算やプログラム言語の形式化にあたって
「変数と変数束縛を、どう表現するか」が問題

変数束縛の形式化手法 2

変数名を文字列で表す

$\lambda x.\lambda y.x + y$ **の表現**:
(Lam "x" (Lam "y" (Plus (Var "x") (Var "y"))))

問題点 **変数名の衝突**
1 つの式が、2 通りに表現される
恒等関数 Lam "x" (Var "x") **と** Lam "y" (Var "y")

変数をインデックスで表す (de Bruijn インデックス)

インデックス:**変数宣言と出現の間の距離・宣言の数**
$\lambda x.\lambda y.x + y$ **の表現**: (Lam (Lam (Plus (Var 1) (Var 0))))

利点 **1 つの式の表現が１つのみ**

CPS 変換の完全性

Theorem (CPS 変換の完全性 (非形式的))

任意のラムダ式 e と e' について

$$\vdash_{CPS} [\![e]\!] = [\![e']\!] \text{ ならば } \vdash_{\lambda_C} e = e'$$

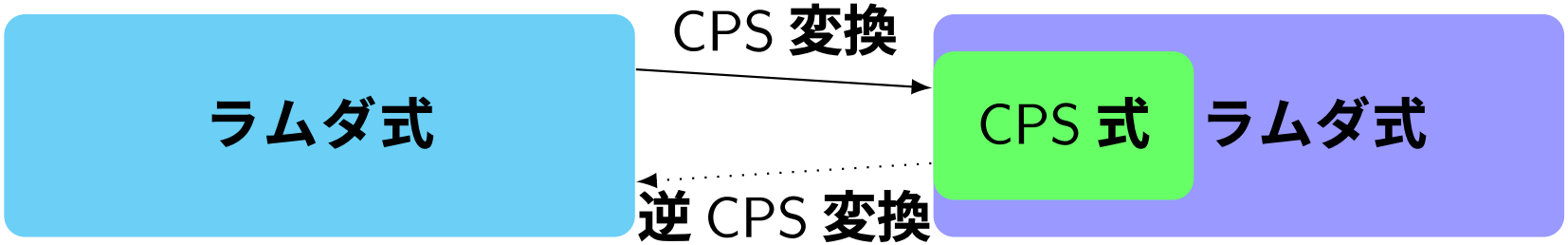
e と e' を CPS 変換したものが等しければ、
ラムダ計算 (関数型プログラム言語) で $e = e'$ である
すなわち、CPS 変換は、「プログラムの等しさ」を保存する

証明の形式化に存在する問題 1

完全性の証明の概要

1. CPS **変換に対して、逆** CPS **変換を定義する**
2. **逆** CPS **変換が、**CPS **変換の弱い逆関数であることを示す**
3. **逆** CPS **変換が、「プログラムの等しさ」を保存することを証明する**

逆 CPS **変換** (CPS^{-1}) **の定義域**=CPS **式**



証明の形式化に存在する問題 2

Definition (逆 CPS 変換の定義の一部)

継続変数 k について $CPS^{-1}(k) = \lambda x.x$

証明の形式化に存在する問題 2

Definition (逆 CPS 変換の定義の一部)

継続変数 k について $CPS^{-1}(k) = \lambda x.x$

CPS **式中の継続変数は逆** CPS **変換において消去される**

$$CPS^{-1}(\lambda x.\lambda k.\lambda y.\cdots)$$
$$\Rightarrow \lambda x.\_\_\_\lambda y.\cdots$$

証明の形式化に存在する問題 2

Definition (逆 CPS 変換の定義の一部)

継続変数 k について $CPS^{-1}(k) = \lambda x.x$

CPS **式中の継続変数は逆** CPS **変換において消去される**

$$CPS^{-1}(\lambda x.\lambda k.\lambda y.\cdots)$$
$$\Rightarrow \lambda x.\_\_\_\lambda y.\cdots$$

⇒de Bruijn **インデックス計算に影響**

証明の形式化に存在する問題 2

Definition (逆 CPS 変換の定義の一部)

継続変数 k について $CPS^{-1}(k) = \lambda x.x$

CPS **式中の継続変数は逆** CPS **変換において消去される**

$$CPS^{-1}(\lambda x.\lambda k.\lambda y.\cdots)$$
$$\Rightarrow \lambda x.\_\_\_\lambda y.\cdots$$

⇒de Bruijn **インデックス計算に影響**
このような継続変数 k の計算を形式化するには、通常の de Bruijn **インデックスの計算より複雑な定義が必要になる**

本研究での解決法

解決法

継続変数と、それ以外の（通常の）変数を別に扱う
$\Rightarrow$ それぞれの変数のインデックスを別々に計算して表現する

本研究での解決法

解決法

継続変数と、それ以外の（通常の）変数を別に扱う
$\Rightarrow$ それぞれの変数のインデックスを別々に計算して表現する

式 $\lambda x.\lambda k.\lambda y.(k\ x)$ の表現の比較

	$\lambda x.$	$\lambda k.$	$\lambda y.$	$(k\ x)$
従来	Lam	$(Lam$	$(Lam$	$(App\ (KVar\ 1)\ (Var\ 2))))$
本研究	Lam	$(KLam$	$(Lam$	$(App\ (KVar\ 0)\ (Var\ 1))))$

本研究での解決法

解決法

継続変数と、それ以外の（通常の）変数を別に扱う
$\Rightarrow$ それぞれの変数のインデックスを別々に計算して表現する

式 $\lambda x.\lambda k.\lambda y.(k\ x)$ の表現の比較

	$\lambda x.$	$\lambda k.$	$\lambda y.$	$(k\ x)$
従来	Lam	$(Lam$	$(Lam$	$(App\ (KVar\ 1)\ (Var\ 2))))$
本研究	Lam	$(KLam$	$(Lam$	$(App\ (KVar\ 0)\ (Var\ 1))))$

継続変数が逆 CPS 変換により消去されても
通常変数のインデックスには影響がでない

本研究での解決法

解決法

継続変数と、それ以外の（通常の）変数を別に扱う
⇒ それぞれの変数のインデックスを別々に計算して表現する

式 $\lambda x.\lambda k.\lambda y.(k\ x)$ **の表現の比較**

	$\lambda x.$	$\lambda k.$	$\lambda y.$	$(k\ x)$
従来	Lam	$(Lam$	$(Lam$	$(App\ (KVar\ 1)\ (Var\ 2))))$
本研究	Lam	$(KLam$	$(Lam$	$(App\ (KVar\ 0)\ (Var\ 1))))$

継続変数が逆 CPS **変換により消去されても**
通常変数のインデックスには影響がでない
形式化の複雑さを解消できた

まとめ・課題

まとめ

- CPS 変換に関する性質（型保存定理・完全性の中心的補題）を Coq を用いて形式化した
- 完全性の中心的補題の形式化で、変数を区別するように de Bruijn インデックスを改良することで、形式化の複雑さを解消した

課題

完全性の証明を完結させること