Изчисления в λ-смятането

Трифон Трифонов

λ-смятане и теория на доказателствата, 2018/19 г.

19–26 март 2019 г.

Симулиране на естествени числа

Дефиниция (n-кратно прилагане на функция)

Нека $n \in \mathbb{N}$ и $F, X \in \Lambda$. Дефинираме n-кратно прилагане на F над X ($F^n X \in \Lambda$):

1. $F^0 X := X$,
2. $F^{n+1} X := F(F^n X)$.

$F^5 X \equiv F(F(F(F(FX))))$

Симулиране на естествени числа

Дефиниция (n-кратно прилагане на функция)

Нека $n \in \mathbb{N}$ и $F, X \in \Lambda$. Дефинираме n-кратно прилагане на F над X ($F^n X \in \Lambda$):

1. $F^0 X := X$,
2. $F^{n+1} X := F(F^n X)$.

Дефиниция (Нумерали на Church)

За $n \in \mathbb{N}$ дефинираме комбинатора $c_n := \lambda_{f,x} f^n x$, който представя n.

Симулиране на естествени числа

Дефиниция (n-кратно прилагане на функция)

Нека $n \in \mathbb{N}$ и $F, X \in \Lambda$. Дефинираме n-кратно прилагане на F над X ($F^n X \in \Lambda$):

1. $F^0 X := X$,
2. $F^{n+1} X := F(F^n X)$.

Дефиниция (Нумерали на Church)

За $n \in \mathbb{N}$ дефинираме комбинатора $c_n := \lambda_{f,x} f^n x$, който представя n.

Примери:

- $c_0 := \lambda_{f,x} x \equiv \lambda_f I \equiv K^*$

Симулиране на естествени числа

Дефиниция (n-кратно прилагане на функция)

Нека $n \in \mathbb{N}$ и $F, X \in \Lambda$. Дефинираме n-кратно прилагане на F над X ($F^n X \in \Lambda$):

1. $F^0 X := X$,
2. $F^{n+1} X := F(F^n X)$.

Дефиниция (Нумерали на Church)

За $n \in \mathbb{N}$ дефинираме комбинатора $c_n := \lambda_{f,x} f^n x$, който представя n.

Примери:

- $c_0 := \lambda_{f,x} x \equiv \lambda_f I \equiv K^*$
- $c_1 := \lambda_{f,x} fx \overset{\eta}{=} \lambda_f f \equiv I$

Симулиране на естествени числа

Дефиниция (n-кратно прилагане на функция)

Нека $n \in \mathbb{N}$ и $F, X \in \Lambda$. Дефинираме n-кратно прилагане на F над X ($F^n X \in \Lambda$):

1. $F^0 X := X$,
2. $F^{n+1} X := F(F^n X)$.

Дефиниция (Нумерали на Church)

За $n \in \mathbb{N}$ дефинираме комбинатора $c_n := \lambda_{f,x} f^n x$, който представя n.

Примери:

- $c_0 := \lambda_{f,x} x \equiv \lambda_f I \equiv K^*$
- $c_1 := \lambda_{f,x} fx \overset{\eta}{=} \lambda_f f \equiv I$
- $c_5 := \lambda_{f,x} f(f(f(f(fx))))$

Числови функции

Да се реализират:

- $c_S\, c_n \overset{\beta}{=} c_{n+1}$

Числови функции

Да се реализират:

- $c_S\, c_n \overset{\beta}{=} c_{n+1}$
- $c_+\, c_m\, c_n \overset{\beta}{=} c_{m+n}$

Числови функции

Да се реализират:

- $c_S c_n \overset{\beta}{=} c_{n+1}$
- $c_+ c_m c_n \overset{\beta}{=} c_{m+n}$
- $c_* c_m c_n \overset{\beta}{=} c_{mn}$

Числови функции

Да се реализират:

- $c_S\, c_n \overset{\beta}{=} c_{n+1}$
- $c_+\, c_m\, c_n \overset{\beta}{=} c_{m+n}$
- $c_*\, c_m\, c_n \overset{\beta}{=} c_{mn}$
- $c_{\exp}\, c_m\, c_n \overset{\beta}{=} c_{m^n}$

Числови функции

Да се реализират:

- $c_S\, c_n \overset{\beta}{=} c_{n+1}$
- $c_+\, c_m\, c_n \overset{\beta}{=} c_{m+n}$
- $c_*\, c_m\, c_n \overset{\beta}{=} c_{mn}$
- $c_{\exp}\, c_m\, c_n \overset{\beta}{=} c_{m^n}$
- $c_{\mathrm{hyp}}\, c_m\, c_n \overset{\beta}{=} c_p$, където $p = \underbrace{m^{m^{\cdot^{\cdot^{\cdot^{m}}}}}}_{n}$.

Числови функции

Да се реализират:

- $c_S c_n \overset{\beta}{=} c_{n+1}$
- $c_+ c_m c_n \overset{\beta}{=} c_{m+n}$
- $c_* c_m c_n \overset{\beta}{=} c_{mn}$
- $c_{\exp} c_m c_n \overset{\beta}{=} c_{m^n}$
- $c_{\mathrm{hyp}} c_m c_n \overset{\beta}{=} c_p$, където $p = \underbrace{m^{m^{\cdot^{\cdot^{\cdot^m}}}}}_{n}$.

Задача

Нека дефинираме $c_I := \lambda_n n c_S c_0$.

1. *Да се докаже, че за произволно* $n \in \mathbb{N}$ *е изпълнено* $c_I c_n \overset{\beta}{=} c_n$.
2. *Вярно ли е, че* $C_I \overset{\beta\eta}{=} I$*? Доказателство или контрапример.*

Булеви стойности

Дефинираме

- $c_{\mathsf{tt}} := \lambda_{x,y} x \equiv K$ — булева истина
- $c_{\mathsf{ff}} := \lambda_{x,y} y \equiv K^*$ — булева лъжа

Булеви стойности

Дефинираме

- $c_{\mathsf{tt}} := \lambda_{x,y} x \equiv K$ — булева истина
- $c_{\mathsf{ff}} := \lambda_{x,y} y \equiv K^*$ — булева лъжа

Как можем да дефинираме оператор, съответстващ на “if-then-else”?

Булеви стойности

Дефинираме

- $c_{\mathsf{tt}} := \lambda_{x,y} x \equiv K$ — булева истина
- $c_{\mathsf{ff}} := \lambda_{x,y} y \equiv K^*$ — булева лъжа

Как можем да дефинираме оператор, съответстващ на “if-then-else”?
Да се дефинират логическите операции $c_\neg, c_\wedge, c_\vee$.

Булеви стойности

Дефинираме

- $c_{\mathsf{tt}} := \lambda_{x,y} x \equiv K$ — булева истина
- $c_{\mathsf{ff}} := \lambda_{x,y} y \equiv K^*$ — булева лъжа

Как можем да дефинираме оператор, съответстващ на “if-then-else”?
Да се дефинират логическите операции $c_\neg, c_\wedge, c_\vee$.
Да се дефинират следните предикати:

- $c_{=0}\, c_n \overset{\beta}{=} c_{n=0}$

Булеви стойности

Дефинираме

- $c_{\mathsf{tt}} := \lambda_{x,y} x \equiv K$ — булева истина
- $c_{\mathsf{ff}} := \lambda_{x,y} y \equiv K^*$ — булева лъжа

Как можем да дефинираме оператор, съответстващ на “if-then-else”?
Да се дефинират логическите операции $c_\neg, c_\wedge, c_\vee$.
Да се дефинират следните предикати:

- $c_{=0} c_n \stackrel{\beta}{=} c_{n=0}$
- $c_{/2} c_n \stackrel{\beta}{=} c_{(n \mod 2)=0}$

Итерация и рекурсия

Да се опитаме да реализираме:

- $c_P c_n \stackrel{\beta}{=} c_{\max(n-1,0)}$

Итерация и рекурсия

Да се опитаме да реализираме:

- $c_P\, c_n \overset{\beta}{=} c_{\max(n-1,0)}$
- $c_!\, c_n \overset{\beta}{=} c_{n!}$

Итерация и рекурсия

Да се опитаме да реализираме:

- $c_P\, c_n \overset{\beta}{=} c_{\max(n-1,0)}$
- $c_!\, c_n \overset{\beta}{=} c_{n!}$

Трудности:

Итерация и рекурсия

Да се опитаме да реализираме:

- $c_P\, c_n \overset{\beta}{=} c_{\max(n-1,0)}$
- $c_!\, c_n \overset{\beta}{=} c_{n!}$

Трудности:

- Как да “пропуснем” едно прилагане?

Итерация и рекурсия

Да се опитаме да реализираме:

- $c_P\, c_n \overset{\beta}{=} c_{\max(n-1,0)}$
- $c_!\, c_n \overset{\beta}{=} c_{n!}$

Трудности:

- Как да "пропуснем" едно прилагане?
- Как да умножаваме по различно число на всяка стъпка?

Итерация и рекурсия

Да се опитаме да реализираме:

- $c_P\, c_n \overset{\beta}{=} c_{\max(n-1,0)}$
- $c_!\, c_n \overset{\beta}{=} c_{n!}$

Трудности:

- Как да "пропуснем" едно прилагане?
- Как да умножаваме по различно число на всяка стъпка?

Итерация

$$\begin{aligned} h(0) &= x \\ h(n+1) &= f(h(n)) \end{aligned}$$

Рекурсия

$$\begin{aligned} h(0) &= x \\ h(n+1) &= f(n, h(n)) \end{aligned}$$

Наредени двойки

Искаме да дефинираме $c_{\langle\rangle}, c_{\llcorner}, c_{\lrcorner}$ така, че:

- $c_{\llcorner}(c_{\langle\rangle} MN) \overset{\beta}{=} M$
- $c_{\lrcorner}(c_{\langle\rangle} MN) \overset{\beta}{=} N$

Наредени двойки

Искаме да дефинираме $c_{\langle\rangle}, c_{\llcorner}, c_{\lrcorner}$ така, че:

- $c_{\llcorner}(c_{\langle\rangle} MN) \stackrel{\beta}{=} M$
- $c_{\lrcorner}(c_{\langle\rangle} MN) \stackrel{\beta}{=} N$

Дефинираме:

- $c_{\langle\rangle} := \lambda_{x,y,z} zxy$
- $c_{\llcorner} := \lambda_p p c_{\mathsf{tt}}$
- $c_{\lrcorner} := \lambda_p p c_{\mathsf{ff}}$

Реализация на рекурсивни функции

Идея: на всяка стъпка пресмятаме наредената двойка $\langle n, h(n) \rangle$ вместо само $h(n)$.

Реализация на рекурсивни функции

Идея: на всяка стъпка пресмятаме наредената двойка $\langle n, h(n)\rangle$ вместо само $h(n)$.

Реализация на c_P: от $\langle c_n, c_{n-1}\rangle$ трябва да получим $\langle c_{n+1}, c_n\rangle$.

Реализация на рекурсивни функции

Идея: на всяка стъпка пресмятаме наредената двойка $\langle n, h(n)\rangle$ вместо само $h(n)$.

Реализация на c_P: от $\langle c_n, c_{n-1}\rangle$ трябва да получим $\langle c_{n+1}, c_n\rangle$.

Реализация на $c_!$: от $\langle c_n, c_{n!}\rangle$ трябва да получим $\langle c_{n+1}, c_{(n+1)!}\rangle$.

Реализация на рекурсивни функции

Идея: на всяка стъпка пресмятаме наредената двойка $\langle n, h(n)\rangle$ вместо само $h(n)$.

Реализация на c_P: от $\langle c_n, c_{n-1}\rangle$ трябва да получим $\langle c_{n+1}, c_n\rangle$.

Реализация на $c_!$: от $\langle c_n, c_{n!}\rangle$ трябва да получим $\langle c_{n+1}, c_{(n+1)!}\rangle$.

Да се дефинират:

- $c_{\dot{-}}\, c_m c_n \overset{\beta}{=} c_{\max(m-n,0)}$

Реализация на рекурсивни функции

Идея: на всяка стъпка пресмятаме наредената двойка $\langle n, h(n)\rangle$ вместо само $h(n)$.

Реализация на c_P: от $\langle c_n, c_{n-1}\rangle$ трябва да получим $\langle c_{n+1}, c_n\rangle$.

Реализация на $c_!$: от $\langle c_n, c_{n!}\rangle$ трябва да получим $\langle c_{n+1}, c_{(n+1)!}\rangle$.

Да се дефинират:

- $c_{\dot{-}}\, c_m c_n \overset{\beta}{=} c_{\max(m-n,0)}$
- $c_{=} c_m c_n \overset{\beta}{=} c_{m=n}$

Реализация на рекурсивни функции

Идея: на всяка стъпка пресмятаме наредената двойка $\langle n, h(n)\rangle$ вместо само $h(n)$.

Реализация на c_P: от $\langle c_n, c_{n-1}\rangle$ трябва да получим $\langle c_{n+1}, c_n\rangle$.

Реализация на $c_!$: от $\langle c_n, c_{n!}\rangle$ трябва да получим $\langle c_{n+1}, c_{(n+1)!}\rangle$.

Да се дефинират:

- $c_{\dot{-}}\, c_m c_n \overset{\beta}{=} c_{\max(m-n,0)}$
- $c_= c_m c_n \overset{\beta}{=} c_{m=n}$
- $c_< c_m c_n \overset{\beta}{=} c_{m<n}$

Реализация на рекурсивни функции

Идея: на всяка стъпка пресмятаме наредената двойка $\langle n, h(n)\rangle$ вместо само $h(n)$.

Реализация на c_P: от $\langle c_n, c_{n-1}\rangle$ трябва да получим $\langle c_{n+1}, c_n\rangle$.

Реализация на $c_!$: от $\langle c_n, c_{n!}\rangle$ трябва да получим $\langle c_{n+1}, c_{(n+1)!}\rangle$.

Да се дефинират:

- $c_{\dot{-}}\, c_m c_n \overset{\beta}{=} c_{\max(m-n,0)}$
- $c_{=} c_m c_n \overset{\beta}{=} c_{m=n}$
- $c_{<} c_m c_n \overset{\beta}{=} c_{m<n}$
- комбинатори c_{quot} и c_{rem} такива, че:

Реализация на рекурсивни функции

Идея: на всяка стъпка пресмятаме наредената двойка $\langle n, h(n)\rangle$ вместо само $h(n)$.

Реализация на c_P: от $\langle c_n, c_{n-1}\rangle$ трябва да получим $\langle c_{n+1}, c_n\rangle$.

Реализация на $c_!$: от $\langle c_n, c_{n!}\rangle$ трябва да получим $\langle c_{n+1}, c_{(n+1)!}\rangle$.

Да се дефинират:

- $c_{\dot{-}}\, c_m c_n \overset{\beta}{=} c_{\max(m-n,0)}$
- $c_{=} c_m c_n \overset{\beta}{=} c_{m=n}$
- $c_{<} c_m c_n \overset{\beta}{=} c_{m<n}$
- комбинатори c_{quot} и c_{rem} такива, че:
  - $c_{+}(c_{*}(c_{\mathrm{quot}}\, c_m c_n)c_n)(c_{\mathrm{rem}}\, c_m c_n) \overset{\beta}{=} c_m$

Реализация на рекурсивни функции

Идея: на всяка стъпка пресмятаме наредената двойка $\langle n, h(n)\rangle$ вместо само $h(n)$.
Реализация на c_P: от $\langle c_n, c_{n-1}\rangle$ трябва да получим $\langle c_{n+1}, c_n\rangle$.
Реализация на $c_!$: от $\langle c_n, c_{n!}\rangle$ трябва да получим $\langle c_{n+1}, c_{(n+1)!}\rangle$.
Да се дефинират:

- $c_{\dot{-}}\, c_m c_n \overset{\beta}{=} c_{\max(m-n,0)}$
- $c_{=} c_m c_n \overset{\beta}{=} c_{m=n}$
- $c_{<} c_m c_n \overset{\beta}{=} c_{m<n}$
- комбинатори c_{quot} и c_{rem} такива, че:
  - $c_{+}\big(c_{*}\big(c_{\mathrm{quot}}\, c_m c_n\big)c_n\big)\big(c_{\mathrm{rem}}\, c_m c_n\big) \overset{\beta}{=} c_m$
  - $c_{<}\big(c_{\mathrm{rem}}\, c_m c_n\big)c_n \overset{\beta}{=} c_{\mathrm{tt}}$

Реализация на рекурсивни функции

Идея: на всяка стъпка пресмятаме наредената двойка $\langle n, h(n)\rangle$ вместо само $h(n)$.
Реализация на c_P: от $\langle c_n, c_{n-1}\rangle$ трябва да получим $\langle c_{n+1}, c_n\rangle$.
Реализация на $c_!$: от $\langle c_n, c_{n!}\rangle$ трябва да получим $\langle c_{n+1}, c_{(n+1)!}\rangle$.
Да се дефинират:

- $c_{\dot{-}}\, c_m c_n \overset{\beta}{=} c_{\max(m-n,0)}$
- $c_{=}\, c_m c_n \overset{\beta}{=} c_{m=n}$
- $c_{<}\, c_m c_n \overset{\beta}{=} c_{m<n}$
- комбинатори c_{quot} и c_{rem} такива, че:
  - $c_{+}\big(c_{*}\big(c_{\mathrm{quot}}\, c_m c_n\big)c_n\big)\big(c_{\mathrm{rem}}\, c_m c_n\big) \overset{\beta}{=} c_m$
  - $c_{<}\big(c_{\mathrm{rem}}\, c_m c_n\big)c_n \overset{\beta}{=} c_{\mathrm{tt}}$
- $c_{/}\, c_m c_n \overset{\beta}{=} c_{\exists_k(km=n)}$

Реализация на рекурсивни функции

Идея: на всяка стъпка пресмятаме наредената двойка $\langle n, h(n)\rangle$ вместо само $h(n)$.
Реализация на c_P: от $\langle c_n, c_{n-1}\rangle$ трябва да получим $\langle c_{n+1}, c_n\rangle$.
Реализация на $c_!$: от $\langle c_n, c_{n!}\rangle$ трябва да получим $\langle c_{n+1}, c_{(n+1)!}\rangle$.
Да се дефинират:

- $c_{\dot{-}}\, c_m c_n \overset{\beta}{=} c_{\max(m-n,0)}$
- $c_{=}\, c_m c_n \overset{\beta}{=} c_{m=n}$
- $c_{<}\, c_m c_n \overset{\beta}{=} c_{m<n}$
- комбинатори c_{quot} и c_{rem} такива, че:
  - $c_{+}\big(c_{*}\big(c_{\mathrm{quot}}\, c_m c_n\big)c_n\big)\big(c_{\mathrm{rem}}\, c_m c_n\big) \overset{\beta}{=} c_m$
  - $c_{<}\big(c_{\mathrm{rem}}\, c_m c_n\big)c_n \overset{\beta}{=} c_{\mathrm{tt}}$
- $c_{/}\, c_m c_n \overset{\beta}{=} c_{\exists_k(km=n)}$
- $c_{\mathrm{prime}}\, c_n \overset{\beta}{=} c_{\neg\exists_{k,l>1}(kl=n)}$

Симулация на списъци

Задача

Да се предложи дефиниция на списъци в безтиповото λ-смятане. С предложената дефиниция да се реализират:

- *стандартните функции* `length`, `append` *и* `member`
- *функциите от по-висок ред* `map`, `foldr` *и* `filter`.

Симулация на списъци

Задача

Да се предложи дефиниция на списъци в безтиповото λ-смятане. С предложената дефиниция да се реализират:

- *стандартните функции* `length`, `append` *и* `member`
- *функциите от по-висок ред* `map`, `foldr` *и* `filter`.

Упътване: може да се използва модификация на нумералите.

Частично рекурсивни функции

Дефиниция (Частично-рекурсивни функции, ЧРФ)

- функциите $O(x) := 0$, $S(x) := x + 1$ и $I_k^n(x_1, \ldots, x_n) := x_k$ са ЧРФ

Частично рекурсивни функции

Дефиниция (Частично-рекурсивни функции, ЧРФ)

- функциите $O(x) := 0$, $S(x) := x + 1$ и $I_k^n(x_1, \ldots, x_n) := x_k$ са ЧРФ
- ако $g_i(\vec{x})$ и $f(\vec{y})$ са ЧРФ, то $h(\vec{x}) := f(g_1(\vec{x}), \ldots, g_n(\vec{x}))$ е ЧРФ (композиция)

Частично рекурсивни функции

Дефиниция (Частично-рекурсивни функции, ЧРФ)

- функциите $O(x) := 0$, $S(x) := x + 1$ и $I_k^n(x_1, \ldots, x_n) := x_k$ са ЧРФ
- ако $g_i(\vec{x})$ и $f(\vec{y})$ са ЧРФ, то $h(\vec{x}) := f(g_1(\vec{x}), \ldots, g_n(\vec{x}))$ е ЧРФ (композиция)
- ако $f(\vec{x})$ и $g(\vec{x}, y, z)$ са ЧРФ, то функцията h е ЧРФ:

$$\begin{aligned} h(\vec{x}, 0) &:= f(\vec{x}) \\ h(\vec{x}, y+1) &:= g(\vec{x}, y, h(\vec{x}, y)) \end{aligned}$$

(примитивна рекурсия)

$h(\vec{x}, n) = g(\vec{x}, n, f(\vec{x}, n-1, \ldots, g(\vec{x}, 0, f(\vec{x})) \ldots))$

Частично рекурсивни функции

Дефиниция (Частично-рекурсивни функции, ЧРФ)

- функциите $O(x) := 0$, $S(x) := x + 1$ и $I_k^n(x_1, \ldots, x_n) := x_k$ са ЧРФ
- ако $g_i(\vec{x})$ и $f(\vec{y})$ са ЧРФ, то $h(\vec{x}) := f(g_1(\vec{x}), \ldots, g_n(\vec{x}))$ е ЧРФ (композиция)
- ако $f(\vec{x})$ и $g(\vec{x}, y, z)$ са ЧРФ, то функцията h е ЧРФ:

$$\begin{array}{ll} h(\vec{x}, 0) & := f(\vec{x}) \\ h(\vec{x}, y+1) & := g(\vec{x}, y, h(\vec{x}, y)) \end{array} \qquad \text{(примитивна рекурсия)}$$

- ако $f(\vec{x}, y)$ е ЧРФ, то функцията g е ЧРФ:

$$g(\vec{x}) := \begin{cases} y, & \text{ако } f(\vec{x}, y) = 0, f(\vec{x}, z) > 0 \text{ за } z < y, \\ \text{недефинирана}, & \text{иначе.} \end{cases} \qquad \text{(минимизация)}$$

Симулация на ЧРФ

$FV(M) = \emptyset$

Дефиниция (λ-определимост)

Казваме, че функцията $f(\vec{x})$ е λ-определима, ако съществува *комбинатор* $M \in \Lambda$, така че:

- ако $f(\vec{x})$ е дефинирана, то $Mc_{x_1} \ldots c_{x_n} \overset{\beta}{=} c_{f(\vec{x})}$,
- ако $f(\vec{x})$ не е дефинирана, то $Mc_{x_1} \ldots c_{x_n}$ няма нормална форма.

Симулация на ЧРФ

Дефиниция (λ-определимост)

Казваме, че функцията $f(\vec{x})$ е λ-определима, ако съществува *комбинатор* $M \in \Lambda$, така че:

- ако $f(\vec{x})$ е дефинирана, то $Mc_{x_1} \ldots c_{x_n} \overset{\beta}{=} c_{f(\vec{x})}$,
- ако $f(\vec{x})$ не е дефинирана, то $Mc_{x_1} \ldots c_{x_n}$ няма нормална форма.

Ясно е как се λ-определят функциите O, S, I_k^n и композицията.

Симулация на ЧРФ

Дефиниция (λ-определимост)

Казваме, че функцията $f(\vec{x})$ е λ-определима, ако съществува *комбинатор* $M \in \Lambda$, така че:

- ако $f(\vec{x})$ е дефинирана, то $Mc_{x_1} \ldots c_{x_n} \overset{\beta}{=} c_{f(\vec{x})}$,
- ако $f(\vec{x})$ не е дефинирана, то $Mc_{x_1} \ldots c_{x_n}$ няма нормална форма.

Ясно е как се λ-определят функциите O, S, I_k^n и композицията.

Задача

Ако $f(\vec{x})$ и $g(\vec{x}, y, z)$ са λ-определими, да се покаже, че функцията h също е λ-определима:

$$\begin{aligned} h(\vec{x}, 0) &:= f(\vec{x}) \\ h(\vec{x}, y+1) &:= g(\vec{x}, y, h(\vec{x}, y)) \end{aligned} \qquad \text{(примитивна рекурсия)}$$

Неограничено търсене

Има ли функции, които не са примитивно рекурсивни?

Неограничено търсене

Има ли функции, които не са примитивно рекурсивни?
Функция на Ackermann:

$$\begin{aligned} A(0, n) &:= n + 1, \\ A(m + 1, 0) &:= A(m, 1), \\ A(m + 1, n + 1) &:= A(m, A(m + 1, n)). \end{aligned}$$

Неограничено търсене

Има ли функции, които не са примитивно рекурсивни?
Функция на Ackermann:

$$\begin{aligned} A(0, n) &:= n + 1, \\ A(m + 1, 0) &:= A(m, 1), \\ A(m + 1, n + 1) &:= A(m, A(m + 1, n)). \end{aligned}$$

Задача

Да се λ-определи функцията на Ackermann.

Неограничено търсене

Има ли функции, които не са примитивно рекурсивни?
Функция на Ackermann:

$$\begin{aligned} A(0, n) &:= n + 1, \\ A(m + 1, 0) &:= A(m, 1), \\ A(m + 1, n + 1) &:= A(m, A(m + 1, n)). \end{aligned}$$

Задача

Да се λ-определи функцията на Ackermann.

Кога имаме нужда от минимизация?

Неограничено търсене

Има ли функции, които не са примитивно рекурсивни?
Функция на Ackermann:

$$\begin{aligned} A(0, n) &:= n + 1, \\ A(m + 1, 0) &:= A(m, 1), \\ A(m + 1, n + 1) &:= A(m, A(m + 1, n)). \end{aligned}$$

Задача

Да се λ-определи функцията на Ackermann.

Кога имаме нужда от минимизация?
Когато **не знаем**, кога ще приключим търсенето.

Неограничено търсене

Има ли функции, които не са примитивно рекурсивни?
Функция на Ackermann:

$$\begin{aligned} A(0,n) &:= n+1, \\ A(m+1,0) &:= A(m,1), \\ A(m+1,n+1) &:= A(m,A(m+1,n)). \end{aligned}$$

Задача

Да се λ-определи функцията на Ackermann.

Кога имаме нужда от минимизация?
Когато **не знаем**, кога ще приключим търсенето.
Пример:

$$f(n) := \begin{cases} 1, & \text{за } n \leq 1, \\ 1 + f(\frac{n}{2}), & \text{за } n > 1 \text{ — четно}, \\ 1 + f(3n+1), & \text{за } n > 1 \text{ — нечетно}. \end{cases}$$

Теорема за неподвижната точка

Искаме да можем да смятаме произволни (общи) рекурсивни функции!

Теорема за неподвижната точка

Искаме да можем да смятаме произволни (общи) рекурсивни функции!

Теорема (за неподвижната точка)

За всяко $F \in \Lambda$ *съществува* $X \in \Lambda$ *такова, че* $FX \overset{\beta}{=} X$.

Теорема за неподвижната точка

Искаме да можем да смятаме произволни (общи) рекурсивни функции!

Теорема (за неподвижната точка)

За всяко $F \in \Lambda$ съществува $X \in \Lambda$ такова, че $FX \stackrel{\beta}{=} X$. Нещо повече, съществува комбинатор $Y \in \Lambda$, който намира X, т.е. $F(YF) \stackrel{\beta}{=} YF$.

Теорема за неподвижната точка

Искаме да можем да смятаме произволни (общи) рекурсивни функции!

Теорема (за неподвижната точка)

За всяко $F \in \Lambda$ съществува $X \in \Lambda$ такова, че $FX \overset{\beta}{=} X$.Нещо повече, съществува комбинатор $Y \in \Lambda$, който намира X, т.е. $F(YF) \overset{\beta}{=} YF$.

Доказателство.

Дефинираме $\omega_F := \lambda_x F(xx)$ и $Y := \lambda_F \omega_F \omega_F$.

Теорема за неподвижната точка

Искаме да можем да смятаме произволни (общи) рекурсивни функции!

Теорема (за неподвижната точка)

За всяко $F \in \Lambda$ съществува $X \in \Lambda$ такова, че $FX \overset{\beta}{=} X$.Нещо повече, съществува комбинатор $Y \in \Lambda$, който намира X, т.е. $F(YF) \overset{\beta}{=} YF$.

Доказателство.

Дефинираме $\omega_F := \lambda_x F(xx)$ и $Y := \lambda_F \omega_F \omega_F$.Така

$$YF \xrightarrow{\beta} \omega_F \omega_F \equiv (\lambda_x F(xx))\omega_F \xrightarrow{\beta} F(\omega_F \omega_F) \xleftarrow{\beta} F(YF).$$

□

Комбинатори за неподвижни точки

Видяхме, че $YF \overset{\beta}{=} F(YF)$. Има ли комбинатор Θ за който $\Theta F \overset{\beta}{\twoheadrightarrow} F(\Theta F)$?

Комбинатори за неподвижни точки

Видяхме, че $YF \overset{\beta}{=} F(YF)$. Има ли комбинатор Θ за който $\Theta F \overset{\beta}{\twoheadrightarrow} F(\Theta F)$?

$$\Theta := (\lambda_{x,F}(xxF))(\lambda_{x,F}F(xxF))$$

Комбинатори за неподвижни точки

Видяхме, че $YF \stackrel{\beta}{=} F(YF)$. Има ли комбинатор Θ за който $\Theta F \stackrel{\beta}{\twoheadrightarrow} F(\Theta F)$?

$$\Theta := (\lambda_{x,F}(xxF))(\lambda_{x,F}F(xxF))$$

При наивна реализация на комбинаторите Y или Θ се сблъскваме с проблем: рекурсивните функции не връщат резултат!

Комбинатори за неподвижни точки

Видяхме, че $YF \overset{\beta}{=} F(YF)$. Има ли комбинатор Θ за който $\Theta F \overset{\beta}{\twoheadrightarrow} F(\Theta F)$?

$$\Theta := (\lambda_{x,F}(xxF))(\lambda_{x,F}F(xxF))$$

При наивна реализация на комбинаторите Y или Θ се сблъскваме с проблем: рекурсивните функции не връщат резултат!
Причината е, че Y и Θ могат да се редуцират до безкрайност.

Комбинатори за неподвижни точки

Видяхме, че $YF \overset{\beta}{=} F(YF)$. Има ли комбинатор Θ за който $\Theta F \overset{\beta}{\twoheadrightarrow} F(\Theta F)$?

$$\Theta := (\lambda_{x,F}(xxF))(\lambda_{x,F}F(xxF))$$

При наивна реализация на комбинаторите Y или Θ се сблъскваме с проблем: рекурсивните функции не връщат резултат!
Причината е, че Y и Θ могат да се редуцират до безкрайност.
При оценяване **по стойност** процесът не завършва.

Комбинатори за неподвижни точки

Видяхме, че $YF \overset{\beta}{=} F(YF)$. Има ли комбинатор Θ за който $\Theta F \overset{\beta}{\twoheadrightarrow} F(\Theta F)$?

$$\Theta := (\lambda_{x,F}(xxF))(\lambda_{x,F}F(xxF))$$

При наивна реализация на комбинаторите Y или Θ се сблъскваме с проблем: рекурсивните функции не връщат резултат!
Причината е, че Y и Θ могат да се редуцират до безкрайност.
При оценяване **по стойност** процесът не завършва.
Идея: "Блокираме" редукцията чрез η-експанзия.

Комбинатори за неподвижни точки

Видяхме, че $YF \overset{\beta}{=} F(YF)$. Има ли комбинатор Θ за който $\Theta F \overset{\beta}{\twoheadrightarrow} F(\Theta F)$?

$$\Theta := (\lambda_{x,F}(xxF))(\lambda_{x,F}F(xxF))$$

При наивна реализация на комбинаторите Y или Θ се сблъскваме с проблем: рекурсивните функции не връщат резултат!
Причината е, че Y и Θ могат да се редуцират до безкрайност.
При оценяване **по стойност** процесът не завършва.
Идея: "Блокираме" редукцията чрез η-експанзия.
Можем да използва следния комбинатор $Z \overset{\eta}{=} Y$:

$$Z := (\lambda_x F(\lambda_y xxy))(\lambda_x F(\lambda_y xxy))$$

Още комбинатори за неподвижни точки

Лема 1 (Характеризация на комбинатори за неподвижни точки)

$M \in \Lambda$ е комбинатор за неподвижна точка $\iff$ M е неподвижна точка на SI.

Още комбинатори за неподвижни точки

Лема 1 (Характеризация на комбинатори за неподвижни точки)

$M \in \Lambda$ е комбинатор за неподвижна точка $\iff$ M е неподвижна точка на SI.

Други комбинатори за неподвижни точки:

- $Y^0 := Y$, $Y^{n+1} := Y^n(SI)$
- Факт: $Y^1 \overset{\beta}{\twoheadrightarrow} \Theta$
- $Y_M := \lambda_f(\lambda_{x,z} f(xxz))(\lambda_{x,z} f(xxz))M$ за произволен $M \in \Lambda$

Симулация на минимизация

Задача

Ако $f(\vec{x}, y)$ е λ-определима, да се покаже, че функцията g също е λ-определима

$$g(\vec{x}) := \begin{cases} y, & \text{ако } f(\vec{x}, y) = 0, f(\vec{x}, z) > 0 \text{ за } z < y, \\ \text{недефинирана}, & \text{иначе.} \end{cases} \quad \text{(минимизация)}$$

Симулация на минимизация

Задача

Ако $f(\vec{x}, y)$ е λ-определима, да се покаже, че функцията g също е λ-определима

$$g(\vec{x}) := \begin{cases} y, & \text{ако } f(\vec{x}, y) = 0, f(\vec{x}, z) > 0 \text{ за } z < y, \\ \text{недефинирана}, & \text{иначе.} \end{cases} \quad \text{(минимизация)}$$

Следствие (Kleene)

Всяка ЧРФ е λ-определима.