БЕЛЕЖКИ ПО λ-СМЯТАНЕ И ТЕОРИЯ НА ДОКАЗАТЕЛСТВАТА

ТРИФОН ТРИФОНОВ

Увод

Този курс се състои от две части: λ-смятане и теория на доказателствата. В първата част на курса ще бъдат разгледани основите на λ-смятането като модел на функционални изчисления, ще бъдат разгледани свойствата на термовата редукция и ще бъде въведено понятието за синтактичен тип. Втората част ще е посветена на една от четирите "колони" на математическата логика: теорията на доказателствата. Ще бъде разгледана логическата страна на понятието формално доказателство и ще бъдат разгледани няколко системи за изразяване на формални доказателства. Ще бъде показана тясната връзка между системите за изразяване на доказателства и типовото λ-смятане. Ще бъдат разгледани по-сложни типови системи и някои от техните приложения в компютърната лингвистика и в извличане на функционални програми от доказателства.

Ще считаме, че читателят е запознат със следните понятия и идеи:

- машина на Тюринг
- частична и тотална функция (теоретико-множествена дефиниция)
- изброимост на изчислимите функции
- неизброимост на функциите над естествени числа
- стоп-проблем на Тюринг
- други примери за неизчислими функции:
  - проверка за тоталност/недефинираност
  - проверка за теорема/тавтология
  - теорема на Райс
- разликата между разрешимост и конструктивност (има ли в π поне n поредни седмици)
- частично-рекурсивни функции
- неконструктивни доказателства (парадоксът за пияниците, $a^b \in \mathbb{Q}$)
- теорема на Кнастер-Тарски за неподвижната точка

1. Безтипово λ-смятане

λ-смятането е теория на математическите функции, разглеждаща ги като *синактични правила* вместо от теоретико-множествена гледна точка (като множества от двойки). Създадено от Alonzo Church през 1932–33 г. Замислено е като фундаментална математическа теория на функциите, но е доказано противоречива от Stephen Kleene и John Rosser през 1935 г.

Дата: 27 март 2014 г.

През 1936 г. Church се концентрира върху изчислителния аспект на λ-смятането, като в същата година Kleene показва, че функциите, определими чрез λ-смятане са точно общите рекурсивни функции на Jacques Herbrand и Kurt Gödel, а самият Alan Turing през 1936–37 г. показва, че λ-определимите функции съвпадат с тези, изчислими с машина на Тюринг. Така се оказва, че три независими опита да бъдат формализирани "ефективно" изчислимите функции водят до еквивалентни модели. Обобщението на тези наблюдения е тезисът на Church-Turing, който твърди, че интуитивното понятие за автоматично изчислима функция съвпада с формалната математическа дефиниция на изчислима функция (чрез която и да е от еквивалентните формулировки на това понятие).

λ-смятането може да се разглежда като концептуален език за програмиране и като такъв дава основите на функционалния стил на програмиране.

1.1. **Синтаксис на λ-смятането.** Нека считаме, че разполагаме с изброим набор от променливи V. Азбуката на λ-смятането се състои от всички променливи, символа λ и скоби.

Дефиниция (λ-термове)**.** Дефинираме индуктивно множеството от λ-термове Λ:

(1) Ако x е променлива, то $x \in \Lambda$.
(2) Ако $M, N \in \Lambda$, то апликацията $(MN) \in \Lambda$.
(3) Ако x е променлива и $M \in \Lambda$, то абстракцията $(\lambda_x M) \in \Lambda$.

Идеята зад апликацията е да моделира прилагане на функцията M над аргумент N. Идеята на абстракцията е да моделира генериране на функция от израза M чрез абстрахиране на променливата x като функционален аргумент.

Бележка (Нотация). Ще записваме $(\dots((M_1M_2)M_3)\dots M_n)$ съкратено като $M_1 \dots M_n$ или като $\vec{M}$, т.е. ще считаме, че апликацията е лявоасоциативна операция. Ще записваме $(\lambda_{x_1}(\lambda_{x_2}(\lambda_{x_3}\dots(\lambda_{x_n}M)\dots)))$ накратко като $\lambda_{x_1,x_2,\dots,x_n}M$ или просто $\lambda_{\vec{x}}M$. Ще пропускаме най-външните скоби. Ще бележим $M \equiv N$, в случай, че два терма съвпадат синтактично, т.е. като поредица от символи.

Пример. Няколко примера за λ-термове:

(1) $I = \lambda_x x$ — идентитет
(2) $\lambda_x y$ — "константна" функция
(3) $\lambda_{x,y} yx$ — функция, който прилага втория си аргумент над първия
(4) $x(\lambda_y x(yz)(\lambda_t xyt))$

Дефиниция (Свободни променливи)**.** Нека $M \in \Lambda$, дефинираме множеството $\mathsf{FV}(M) \subseteq V$ от свободни променливи на M по индукция по построението на M:

(1) $\mathsf{FV}(x) := \{x\}$
(2) $\mathsf{FV}(M_1M_2) := \mathsf{FV}(M_1) \cup \mathsf{FV}(M_2)$
(3) $\mathsf{FV}(\lambda_x N) := \mathsf{FV}(N) \setminus \{x\}$

Дефиниция (Затворени термове)**.** Терма $M \in \Lambda$ наричаме *затворен*, ако $\mathsf{FV}(M) = \emptyset$.

Дефиниция (Свързани променливи)**.** Нека $M \in \Lambda$, дефинираме множеството $\mathsf{BV}(M) \subseteq V$ от свързани променливи на M по индукция по построението на M:

(1) $\mathsf{BV}(x) := \emptyset$

(2) $\mathsf{BV}(M_1 M_2) := \mathsf{BV}(M_1) \cup \mathsf{BV}(M_2)$
(3) $\mathsf{BV}(\lambda_x N) := \mathsf{BV}(N) \cup \{x\}$

Пример. Нека $M := \lambda_{x,y} xyz$, $N := x(\lambda_x yx)z$. Тогава $\mathsf{FV}(M) = \{z\}, \mathsf{BV}(M) = \{x, y\}, \mathsf{FV}(N) = \{x, y, z\}, \mathsf{BV}(N) = \{x\}$.

Основна синтактично преобразувания в λ-смятането е заместването на свободна променлива с λ-терм.

Дефиниция (Субституция)**.** Нека $M, N \in \Lambda$, $x \in V$. Дефинираме индуктивно по построението на M субституцията на x с N в M, която ще отбелязваме с $M\,[x := N]$.

(1) $x\,[x := N] := N$
(2) $y\,[x := N] := y$ за $y \not\equiv x$
(3) $(M_1 M_2)\,[x := N] := (M_1\,[x := N])(M_2\,[x := N])$
(4) $\lambda_x P\,[x := N] := \lambda_x P$
(5) $(\lambda_y P)\,[x := N] := \lambda_y (P\,[x := N])$ за $y \not\equiv x$ и $y \notin \mathsf{FV}(N)$

Важно е да обърнем внимание, че поради допълнителното условие $y \notin \mathsf{FV}(N)$ в последния случай от горната дефиниция, операцията субституция не винаги е дефинирана. Така например, не сме дефинирали кой терм отговаря на прилагането на субституцията $[x := y]$ към терма $\lambda_y x$. Ако не бяхме включили условието свързаната променлива да не е сред свободните променливи на терма, с който заместваме, то бихме получили като резултат $\lambda_x x$. Интуитивно, това не би било коректно, понеже така бихме успели чрез синтактична операция да променим качествено "константната" функция, представяна от терма $\lambda_y x$ на функцията "идентитет", представяна от терма $\lambda_x x$. Подобна аномалия обикновено се нарича "прихващане на променливи" и не е феномен присъщ само за λ-смятането, а също е известна при други езици с квантори, какъвто е например езикът на предикатното смятане от първи ред. Дефиницията избягва проблематичните случаи за сметка на тоталността на операцията субституция, т.е. възможността произволна субституция да е винаги приложима над произволен терм.

Технически, това решение на проблема не е оптимално, тъй като би наложило да правим проверка за дефинираност при всяко прилагане на субституция. Затова често се прилага друга стратегия, известна като "използване на свежи свързани променливи". Тя се базира на интуитивното разбиране, че конкретното име на свързана променлива не е съществено за семантиката на терма. Така например можем да си мислим, че всички термове от вида $\lambda_x x, \lambda_y y, \ldots$ представят една и съща функция идентитет. Формално може да се дефинира релацията на еквивалентност $\overset{\alpha}{=}$, която е в сила за термове, които се различават синтактично само по имената на свързаните си променливи.