Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

ТЕМА ЗА ПРОЕКТ

към курс “Функционално програмиране”
за специалности Информатика, Компютърни науки (2 поток)
и избираема дисциплина
зимен семестър 2024/25 г.

Извод на типове

Synopsis: Имплементирайте система за извеждане на типове в ламбда смятането.

Ламбда смятането (lambda calculus) е формална система в математическата логика, която описва изчисленията само на базата на абстракция (построяване) на функции и апликация (прилагане) на функции върху променливи и/или други функции чрез замяна на свързани променливи. Термовете (изразите) в ламбда смятането се построяват по следните правила:

- ако `x` е променлива, то `x` е терм. Приемаме, че разполагаме с изброимо безкраен списък с променливи.
- ако `M` и `N` са термове, то `(MN)` е терм, получен от прилагането на терма `M` над `N`. Прилагането е ляво асоциативна операция и когато пишем `MNP`, ще подразбираме `((MN)P)`.
- ако `M` е терм, а `x` е променлива, то `λx.M` е терм, получен като построяване на функция с аргумент `x` и тяло `M`. За удобство вместо `λx.λy.`… можем да пишем `λxy.`

Примери за ламбда термове: `λx.x`, `λx.y`, `λxy.x`, `λfx.f(fx)`, `λxyz.xz(yz)`

Едно от свойствата, които даден терм може да има, е тип. Типовете се построяват по следните правила:

- ако α е типова променлива, то α е тип
- ако σ и τ са типове, то σ→τ е тип на функция, приемаща аргумент от тип σ и връщаща резултат от тип τ. Операцията → е дясноасоциативна и когато пишем ρ→σ→τ, ще подразбираме (ρ→(σ→τ)).

Примери за типове: $\alpha \to \alpha$, $\alpha \to \beta$, $\alpha \to \beta \to \alpha$, $(\alpha \to \alpha) \to \alpha \to \alpha$

Важно свойство на типовете, които разглеждаме е, че те са *крайни*, правещи невъзможно съществуването на безкраен тип от рода на $\alpha \to \beta \to \gamma \to \ldots$. За целите на този проект можем да считаме, че няма и безкрайни ламбда термове.

Ще обозначаваме твърдението "термът `M` има тип τ" с `M:τ`. За да определим типа на даден терм, първо е необходимо да направим някакви допускания за типовете на променливите, които участват в него. При различни допускания за променливите е възможно да се получи различен тип на терма. Затова ще разглеждаме твърдения от вида "при допускания Γ термът `M` има тип τ", където Γ е множество от допускания от вида `M:τ`. Така можем да определим типа на произволен термов чрез следните правила

- ако `M:σ→τ` и `N:σ` при едни и същи допускания Γ, то `(MN):τ` при същите допускания Γ
- ако `M:τ` при допускания Γ, сред които присъства и `x:σ`, то `(λx.M):σ→τ` при допускания Γ с премахнато вече използваното допускане `x:σ`.

Примери: `xy:τ` при допускания `x:σ→τ` и `y:σ`, `λx.xy:(σ→τ)→τ` при допускане `y:σ`, `λyx.xy:σ→(σ→τ)→τ` без използването на допускания.

Целта на този проект е по даден ламбда терм, например `λxy.x(xy)` или `λxyz.xz(yz)`, да се изведе негов тип (ако има такъв) чрез типов извод. Извеждането на тип се получава по следната стратегия:

- за да намерим типа τ на терм от вида `(MN)`, трябва първо да намерим такъв тип σ, такъв че `M:σ→τ` и `N:σ`
- за да намерим типа на терм от вида `(λx.M)`, можем да допуснем, че `x:σ` за някой неизползван до момент тип σ и след това да потърсим типа на тялото `M` с добавеното ново допускане, че `x:σ`. Ако в процеса на търсене определим, че `M:τ`, то ще знаем, че `(λx.M):σ→τ`.
- в дъното на рекурсивното търсене ще ни се наложи да потърсим типа на променлива x; този тип можем да определим като погледнем в натрупаните допускания за предположение от вида `x:σ`.

Пример: За да намерим типа на терма `λyx.xy`:

- допускаме, че `y:σ`
- за да намерим типа на терма `λx.xy`:
    - допускаме, че `x:ρ`
    - за да намерим типа на терма xy:

            - търсим типа на терма y
                - но по допускане имаме $y:\sigma$
            - търсим типа на терма x от вида $\sigma\to\tau$
                - но по допускане имаме $x:\rho$
                - полагаме $\rho := \sigma\to\tau$
                - така получаваме $x:\sigma\to\tau$
            - така получаваме $xy:\tau$
        - така получаваме $\lambda x.xy:\rho\to\tau$
        - но $\rho = \sigma\to\tau$
        - затова получаваме $\lambda x.xy:(\sigma\to\tau)\to\tau$
- така получаваме, че $\lambda yx.xy:\sigma\to(\sigma\to\tau)\to\tau$

Примерно описание на идеята за типов извод и основните дефиниции в ламбда смятане можете да намерите тук.