Systémy typovaných lambda kalkulů (část druhá)

LKFP přednáška 12, jaro 2024/25

Jan Laštovička

6. května 2025

1 Dvojí interpretace typů

Popišme výrazy typu A nějaké objekty. Pak A můžeme vykládat jako množinu, kde objekt náleží do množiny, právě když existuje výraz typu A, který jej popisuje. Například uvažujme v systému $\lambda\!\to$ kontext

$$N : *, z : N, s : N \to N.$$

V zadaném kontextu z popisuje nulu, sA, kde $A : N$, popisuje následníka A. Tedy $s(sz)$ popisuje číslo dva. Typ N můžeme vykládat jako množinu všech přirozených čísel s nulou.

Typy můžeme vykládat také jako výroky. Přesněji typy, které jsou proměnnými, vykládáme jako atomické výroky. Typ $(\Pi x : A.B)$ vykládáme jako výrok „pro každé x splňující A platí B“. Speciálně typ $A \to B$ vyložíme jako výrok „jestliže A, pak B“. Například v systému $\lambda\!\to$ můžeme vzhledem ke kontextu

$$A : *, B : *$$

vyložit typ $A \to (B \to A)$ jako výrok „jestliže A, pak B implikuje A“. Pokud existuje výraz A typu B, pak řekneme, že B je *pravdivý výrok* a A nazveme jeho *důkazem*. Například ve stejném systému máme:

$$A : *, B : * \vdash (\lambda a : A \lambda b : B.a) : A \to B \to A$$

Tedy výrok $A \to B \to A$ je pravdivý a výraz $\lambda a : A \lambda b : B.a$ je jeho důkazem. Výrok $A \to B$ vzhledem ke stejnému kontextu pravdivý není.

2 Ještě k systému $\lambda 2$

Připomeňme, že výraz $(\Pi \alpha : *.\alpha)$ jsme označili $\bot$ a že neexistuje výraz A tak, aby $\vdash A : \bot$. Tedy $\bot$ vykládáme jako nepravdu.

Lze odvodit:

$$\vdash (\lambda \beta : * \lambda a : \bot.a\beta) : (\Pi \beta : *.\bot \to \beta).$$

Typ vyložíme jako pravdivý výrok *ex falso sequitur quodlibet*, tedy z nepravdivého výroku plyne cokoliv. V Agdě:

```
proof1 : (β : Set₀) → ((α : Set₀) -> α) → β
proof1 β a = a β
```

Dále lze odvodit:

$$\alpha : *, \beta : * \vdash (\Pi\gamma : *.(\alpha \to \beta \to \gamma) \to \gamma) : *$$

Konjunkci dvou výroků můžeme definovat následovně.

$$\alpha \wedge \beta \equiv \Pi\gamma : *.(\alpha \to \beta \to \gamma) \to \gamma$$

Například platí:

$$A : *, B : *, a : A, b : B \vdash (\lambda\gamma : *\lambda f : (A \to B \to \gamma).fab) : A \wedge B$$

Ověříme v Agdě:

```
data A : Set₀ where
  a : A

data B : Set₀ where
  b : B

proof2 : (γ : Set₀) → (A → B → γ) → γ
proof2 γ f = f a b
```

V kontextu $A : *, B : *, a : A$ je konjunkce $A \wedge B$ nepravdivá.

Konjunkci můžeme použít k definici párů. Například pár čísel $(3, 7)$ reprezentujeme výrazem:

```
pair1 : (γ : Set₀) → (N → N → γ) → γ
pair1 γ f = f 3 7
```

První prvek páru čísel získáme funkcí:

```
first : ((γ : Set₀) → (N → N → γ) → γ) → N
first pair = pair N (λ m n → m)
```

Například výraz `first pair1` má normální formu číslo tři.

Podobně můžeme zavést disjunkci:

$$\alpha \vee \beta \equiv \Pi\gamma : *.(\alpha \to \gamma) \to (\beta \to \gamma) \to \gamma.$$

Například platí:

$$A : *, B : *, a : A \vdash (\lambda\gamma : *\lambda f : (A \to \gamma)\lambda g : (B \to \gamma).fa) : A \vee B$$

V kontextu $A : *, B : *$ je disjunkce $A \vee B$ nepravdivá. Typ $\alpha \vee \beta$ odpovídá součtu typů α a β. Například levé vložení čísla tři do součtu typu `N` a `N` reprezentujeme výrazem:

```
inl3 : (γ : Set₀) → (N → γ) → (N → γ) → γ
inl3 γ f g = f 3
```

3 Systém $\lambda\underline{\omega}$

Systém $\lambda\underline{\omega}$ má všechna základní pravidla a navíc zvláštní pravidla $(*, *)$ a $(\Box, \Box)$. Pomocí jeho pravidel odvodíme:

$$\dfrac{\vdash * : \Box \qquad \dfrac{\vdash * : \Box \qquad \vdash * : \Box}{x : * \vdash * : \Box}}{\vdash (\Pi x : *.*) : \Box}$$

Tedy $\vdash (* \to *) : \Box$, což znamenám že $* \to *$ je forma. Odvodíme její konstruktor:

$$\dfrac{\dfrac{\vdots}{\alpha : * \vdash (\alpha \to \alpha) : *} \qquad \dfrac{\vdots}{\vdash (* \to *) : \Box}}{\vdash (\lambda\alpha : *.\alpha \to \alpha) : (* \to *)}$$

V Agdě:

```
UnaryOp : Set₀ → Set₀
UnaryOp α = α → α
```

Jak následující příklad ukazuje konstruktor $\lambda\alpha : *.\alpha \to \alpha$ lze využít k vytváření typů. Přesněji pokud jej aplikujeme na typ, dostaneme typ funkcí na zadaném typu.

$$\beta : * \vdash (\lambda\alpha : *.\alpha \to \alpha)\beta : *;$$
$$\beta : *, x : \beta \vdash (\lambda y : \beta.x) : (\lambda\alpha : *.\alpha \to \alpha)\beta.$$

Použití v Agdě:

```
f1 : UnaryOp N
f1 = succ
```

Můžeme vytvářet abstrakce, které očekávají konstruktory:

$$\alpha : *, f : * \to * \vdash f(f\alpha) : *;$$
$$\alpha : * \vdash (\lambda f : * \to *.f(f\alpha)) : (* \to *) \to *.$$

V Agdě:

```
FFN : (Set₀ → Set₀) → Set₀
FFN f = f (f N)

f2 : FFN UnaryOp
f2 f x = f (f x)
```

4 Systém λP

Systém λP používá základní pravidla a zvláštní pravidla $(*, *)$ a $(*, \Box)$. Můžeme v něm odvodit:

$$\dfrac{\dfrac{\vdash * : \Box}{A : * \vdash A : *} \qquad \dfrac{\dfrac{\vdash * : \Box \qquad \vdash * : \Box}{A : * \vdash * : \Box} \qquad \dfrac{\vdash * : \Box}{A : * \vdash A : *}}{A : *, x : A \vdash * : \Box}}{A : * \vdash (\Pi x : A.*) : \Box}$$

Tedy $A : * \vdash (A \to *) : \Box$, neboli $A \to *$ je forma. Pokud A vykládáme jako množinu, pak konstruktory formy $A \to *$ můžeme vykládat jako predikáty na A.

Platí:

$$A : *, P : (A \to *), a : A \vdash Pa : *$$

Typ Pa můžeme vykládat jako výrok „objekt a vyhovuje predikátu P“. Například takto můžeme definovat v Agdě sudá čísla:

```
data IsEven : N → Set₀ where
  even-zero : IsEven zero
  even-addTwo : (n : N) → IsEven n → IsEven (succ (succ n))
```

Důkaz, že dva je sudé číslo:

```
proof1 : IsEven (succ (succ zero))
proof1 = even-addTwo zero even-zero
```

Výrok „číslo jedna je sudé číslo“ je nepravdivý, protože nelze nají výraz typu `IsEven (succ zero)`.

Lze odvodit:

$$A : *, P : (A \to A \to *) \vdash (\Pi a : A.Paa) : *$$

Výraz P můžeme vykládat jako binární relaci na množině A. Odvozený typ lze vykládat jako výrok tvrdící, že P je reflexivní. Například vezměme definici relace menší nebo rovno na číslech:

```
data ≤ : N → N → Set₀ where
  ≤zero : (n : N) → ≤ zero n
  ≤succ : (m : N) → (n : N) → ≤ m n → ≤ (succ m) (succ n)
```

Symbol $\leq$ vložíte napsáním `\le`. Důkaz, že $1 \leq 2$:

```
proof2 : ≤ 1 2
proof2 = ≤succ 0 1 (≤zero 1)
```

Důkaz reflexivity $\leq$:

```
≤refl : (n : N) → ≤ n n
≤refl zero = ≤zero zero
≤refl (succ n) = ≤succ n n (≤refl n)
```

Dále lze odvodit:

$$A : *, P : A \to *, Q : A \to * \vdash (\Pi a : A.(Pa \to Qa)) : *$$

Proměnné P a Q vykládáme jako podmnožiny množiny A. Typ vykládáme jako výrok tvrdící, že P je podmnožinou Q.

Výrok tvrdící, že podmnožinovost je reflexivní:

$$A : *, P : A \to * \vdash (\Pi a : A.(Pa \to Pa)) : *$$

Důkaz jeho pravdivosti:

$$A : *, P : A \to * \vdash (\lambda a : A \lambda x : Pa.x) : (\Pi a : A.(Pa \to Pa)).$$

Průzkum systému uzavřeme důkazem pravdivosti následujícího výroku. Vezměme neprázdnou množinu A, predikát P na A a výrok Q. Jestliže

1. pro každé $a \in A$ splňující P je Q pravdivé
2. a každý prvek A splňuje P,

pak Q je pravdivé. Přepis výroku do typu a nalezení důkazu:

$$A : *, P : A \to *, Q : *, a_0 : A \vdash (\lambda x : (\Pi a : A.Pa \to Q) \lambda y : (\Pi a : A.Pa).xa_0(ya_0)) : \\ (\Pi a : A.Pa \to Q) \to (\Pi a : A.Pa) \to Q$$

5 Systém $\lambda\omega$

Systém používá základní pravidla a zvláštní pravidla $(*, *)$, $(\Box, *)$ $(\Box, \Box)$.

Definujeme negaci $\neg \equiv (\lambda \alpha : *.\alpha \to \bot)$. Platí:

$$\vdash \neg : (* \to *)$$
$$\alpha : * \vdash \neg\alpha : *$$

Agda používá nekonečně mnoho druhů $\texttt{Set}_0$, $\texttt{Set}_1$, ... Minule jsme řekli, že $\texttt{Set}_0$ odpovídá druhu $*$ a $\texttt{Set}_1$ druhu $\Box$. To je pravda jen pro první seznámení. Přesněji je jejich vztah složitější. V Agdě platí, že $\texttt{Set}_i$:$\texttt{Set}_{i+1}$ pro každé i. Navíc platí, že pokud A :$\texttt{Set}_i$, pak A :$\texttt{Set}_{i+1}$ pro každé i. Pravidlo uvedení výrazu součinu má v Agdě tvar:

$$\frac{\Gamma \vdash A : \texttt{Set}_i \quad \Gamma, x : A \vdash B : \texttt{Set}_i}{\Gamma \vdash (x : A \to B) : \texttt{Set}_i}$$

Definice negace v Agdě:

```
⊥ : Set₁
⊥ = (α : Set₀) → α

¬ : Set₀ → Set₁
¬ α = α → ⊥
```

Dále budeme místo $\texttt{Set}_0$ používat i zkratku `Set`.

Dokážeme následující tvrzení. Pro každé výroky α a β platí, že jestliže α implikuje β, pak negace β implikuje negaci α. Vyjádřením v typovém systému dostaneme:

$$\alpha : *, \beta : * \vdash (\lambda f : (\alpha \to \beta) \lambda n : \neg\beta \lambda a : \alpha . n(fa) : ((\alpha \to \beta) \to (\neg\beta \to \neg\alpha))$$

Ověření důkazu v Agdě:

```
proof1 : (α : Set) → (β : Set) → (α → β) → (¬ β → ¬ α)
proof1 α β f g a = g (f a)
```

Můžeme definovat výraz, který očekává dvě tvrzení a vrací jejich konjunkci $\text{AND} \equiv \lambda\alpha : *\lambda\beta : *.\alpha \wedge \beta$. Platí:

$$\vdash \text{AND} : (* \to * \to *)$$

Konjunkce v Agdě:

```
and : Set → Set → Set₁
and α β = (γ : Set) → (α → β → γ) → γ
```

Dokážeme, že z pravdivosti konjunkce výroků α a β plyne pravdivost výroku α.

$$\alpha : *, \beta : * \vdash (\lambda x : \text{AND}\alpha\beta . x\alpha(\text{K}\alpha\beta)) : (\text{AND}\alpha\beta \to \alpha) : *.$$

Připomeňme použitou definici kombinátoru K:

$$\text{K} \equiv \lambda\alpha : *\lambda\beta : *\lambda x : \alpha\lambda y : \beta . x$$

Ověření v Agdě:

```
proof2 : (α : Set) → (β : Set) → (and α β) → α
proof2 α β f = f α (λ a b → a)
```

Tedy jestliže je pravdivá konjunkce α a β, pak je pravdivá α.
Definujeme negaci $\neg \equiv (\lambda\alpha : *.\alpha \to \bot)$. Platí:

$$\vdash \neg : (* \to *)$$
$$\alpha : * \vdash \neg\alpha : *$$
$$\alpha : *, \beta : * \vdash (\lambda f : (\alpha \to \beta)\lambda n : \neg\beta\lambda a : \alpha.n(fa) : ((\alpha \to \beta) \to (\neg\beta \to \neg\alpha))$$

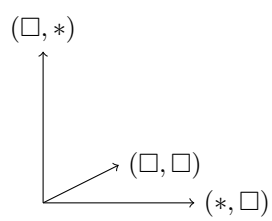
Tedy jestliže α implikuje β, pak negace β implikuje negaci α.

6 Krychle

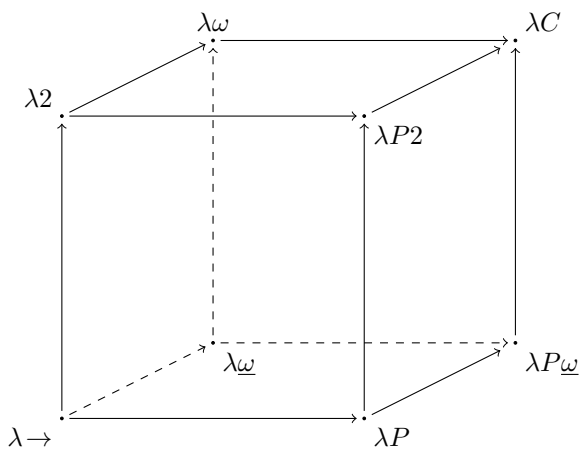
Níže uvedená tabulka uvádí osm systému, z nichž každý má všechna základní pravidla a tabulkou přiřazenou množinu zvláštních pravidel. S některými systémy jsme se již seznámili.

Systém	Množina zvláštních pravidel			
$\lambda\to$	$(*,*)$			
$\lambda 2$	$(*,*)$	$(\Box,*)$		
λP	$(*,*)$		$(*,\Box)$	
$\lambda P2$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	
$\lambda\underline{\omega}$	$(*,*)$			$(\Box,\Box)$
$\lambda\omega$	$(*,*)$	$(\Box,*)$		$(\Box,\Box)$
$\lambda P\underline{\omega}$	$(*,*)$		$(*,\Box)$	$(\Box,\Box)$
λC	$(*,*)$	$(\Box,*)$	$(*,\Box)$	$(\Box,\Box)$

Zvláštní pravidlo $(*,*)$ považujeme za základní, ostatní jsou pokročilá. Pokročilá zvláštní pravidla lze ztotožnit s osami trojrozměrného prostoru:



Typové systémy pak můžeme umístit do vrcholů krychle:



Typové systémy do krychle uspořádal nizozemský logik Henk Barendregt v přehledovém článku *Lambda Calculi with Types* z roku 1992. Článek lze stáhnout z jeho webových stránek.[1] Z právě zmíněného článku jsme vycházeli během posledních dvou přednášek. Stručně ještě uvedeme, co je možné odvodit v zbývajících systémech.

6.1 Systém $\lambda P2$

Negace predikátu:

$$A : *, P : A \to * \vdash (\lambda a : A.Pa \to \bot) : (A \to *)$$

Výrok tvrdí, že binární relace, která je asymetrická je ireflexivní:

$$A : *, P : A \to A \to * \vdash ((\Pi a : A \Pi b : A.Pab \to Pba \to \bot) \to (\Pi a : A.Paa \to \bot)) : *$$

Definujeme takzvanou Leibnizovu rovnost:

$$(x =_A y) \equiv \Pi P : (A \to *).Px \to Py$$

Platí:

$$A : *, x : A, y : A \vdash (x =_A y) : *$$

Důkaz symetrie rovnosti:

$$A : * \vdash (\lambda x : A \lambda y : A \lambda e : (x =_A y) \lambda P : (A \to *) \lambda p : Py.e(\lambda z : A.Pz \to Px)(\lambda q.q)p) : \\ (\Pi x : A \Pi y : A.(x =_A y) \to (y =_A x))$$

V Agdě:

[1] `http://www.cs.ru.nl/~henk/papers.html`

```
Eq : (A : Set₀) → A → A → Set₁
Eq A x y = (P : A → Set₀) → (P x) → (P y)

eqSym : (A : Set₀) → (x : A) → (y : A) →
  (Eq A x y) → (Eq A y x)
eqSym A x y eq P py = eq (λ z → P z → P x) (λ px → px) py
```

Zavedeme existenční kvantifikátor:

$$(\exists x : \alpha.\beta) \equiv \Pi\gamma : *.(\Pi x : \alpha.(\beta \to \gamma)) \to \gamma.$$

6.2 Systém $\lambda P\underline{\omega}$

Funkce přiřazující binárnímu predikátu P na množině A jeho diagonalizaci:

$$A : * \vdash (\lambda P : A \to A \to *\lambda a : A.Paa) : ((A \to A \to *) \to (A \to *)) : \Box$$

To samé, kde A je parametr:

$$\vdash (\lambda A : *\lambda P : A \to A \to *\lambda a : A.Paa) : (\Pi A : *\Pi P : A \to A \to *\Pi a : A.*) : \Box$$

6.3 Systém λC

Funkce přiřadí množině A a predikátu P na A negaci P.

$$\vdash (\lambda A : *\lambda P : A \to *\lambda a : A.\neg(Pa)) : (\Pi A : *.(A \to *) \to (A \to *)) : \Box$$

Funkce pro množinu a predikát vracející výrok, že každý prvek množiny splňuje predikát:

$$\text{ALL} \equiv (\lambda A : *\lambda P : A \to *.\Pi a : A.Pa)$$

Platí:

$$A : *, P : A \to * \vdash \text{ALL } AP : *$$

Máme:

$$(\text{ALL } AP) =_\beta (\Pi a : A.Pa)$$

Například máme:

$$A : *, P : A \to *, a : A \vdash (\lambda f : (\text{ALL } AP).fa) : (\text{ALL } AP \to Pa)$$