

### Typy a formule

```
• K = (\lambda xy.x)^{\alpha \beta \alpha}
```

• 
$$S = \lambda xyz.xz(yz)^{(\delta \quad \epsilon \quad \theta)} \quad (\delta \quad \epsilon) \quad \delta \quad \theta$$

Hilbertov axiomatický systém:

Modus ponens:



## Curry-Howardov izomorfizmus

S K K = I  
S:
$$(\delta \epsilon \theta) (\delta \epsilon) (\delta \theta)$$
  
K:α β α  
K:

$$(\delta \quad \epsilon \quad \theta) = (\alpha \quad \beta \quad \alpha)$$

$$\bullet \quad \delta = \alpha, \epsilon = \beta$$

δ=θ

$$K = (\lambda xy.x)a \quad \beta \quad a \\ S = \lambda xyz.xz(yz)(\delta \quad \epsilon \quad \theta) \quad (\delta \quad \epsilon) \quad \delta \quad \theta$$



# Curry-Howardov izomorfizmus

SKK:(A A)

### Polymorfický (druhorádový) λ-kalkul, System F<sub>2</sub> (Girard-Reynold)

V jednoducho-typovanom λ-kalkule doménou premenných sú funkcie, v druho-rádovom λ-kalkule doménou premenných sú typy:

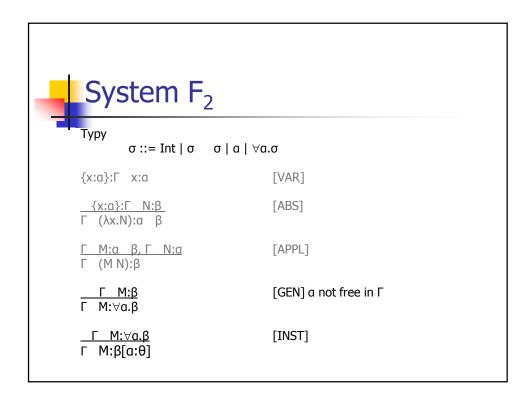
Тур

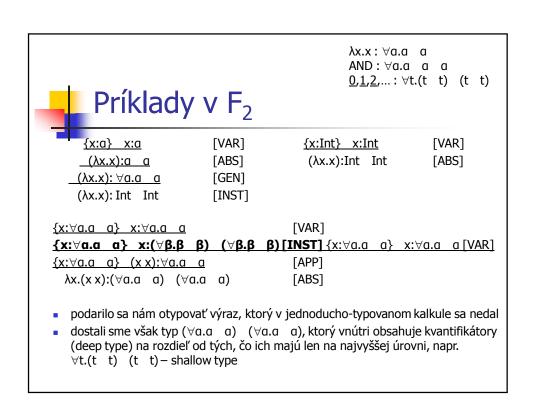
$$\sigma ::= Int \mid \sigma \quad \sigma \mid \alpha \mid \forall \alpha.\sigma \quad - v\check{s}eobecne kvantifikovaný typ$$

- λx.x: ∀a.a a
- NOT: ∀a.a a
- K=TRUE ( $\lambda x.\lambda y.x$ ):  $\forall a.\forall \beta.a$   $\beta$   $\alpha$ , FALSE ( $\lambda x.\lambda y.y$ ):  $\forall a.\forall \beta.a$   $\beta$
- 0,1,2 ( $\lambda f.\lambda x.f(fx)$ ),...:  $\forall a.(a \ a) \ a \ a$

### Dôsledky:

- v takomto  $\lambda$ -kalkule vieme otypovať aj to, čo v  $F_1$  nevieme ( $\omega$ )
- inferencia v takomto kalkule je nerozhodnuteľný problém ☺, 1970-1994
- hierarchia  $F_1$ ,  $F_2$ ,  $F_3$ , ...  $F_{\omega}$
- dependent type (t:Type, t)







### Let polymorfizmus Hindley-Millner

chceme zakázať kvantifikáciu typov vnútri typového výrazu (zakázať deep type)

Туру

$$\begin{split} \sigma &::= \psi \mid \forall \alpha. \sigma \\ \psi &::= \text{Int} \mid \psi \quad \psi \mid \alpha \end{split}$$

premenná viazaná λ výrazom nemôže byť polymofrického typu, napr.

 $f(\lambda x.x)$  where  $f = \lambda g.[...(g 1) ....(g True) ...], alebo v Haskelli:$ 

- idd =  $(\x->x)$
- foo = f idd where  $f = \g->(if (g True) then (g 2) else (g 1))$

Nahradíme pravidlo [GEN] pravidlom [LET]

<u>Γ Μ:β, Γ, x:∀α.β Ν:δ</u>

[LET] a in  $\beta$ , a not free in  $\Gamma$ 

 $\Gamma$  let x=M in N : δ