P114
Konstrukce

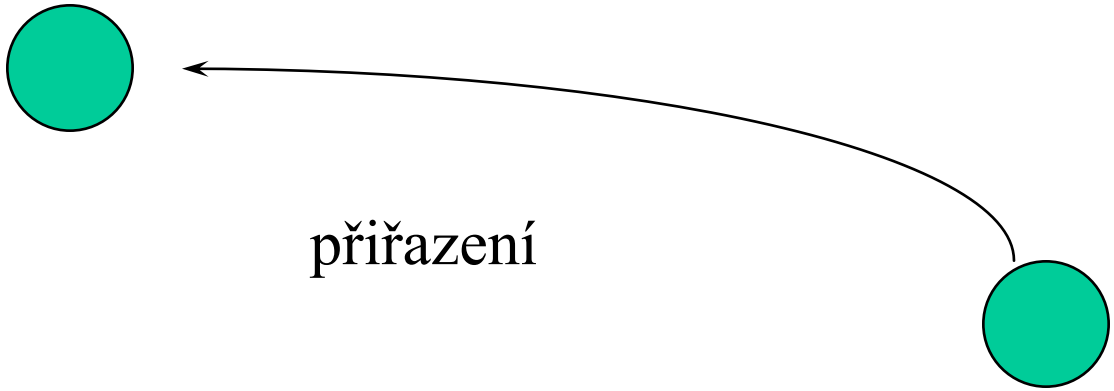
užití λ - kalkulu

5

Témata

- TIL s jednoduchou teorií typů
- atomické konstrukce
- konstrukce aplikace
- konstrukce abstrakce
- konstrukce n-tice a projekce
- ekvivalence konstrukcí
- uzavřené a otevřené konstrukce

Elementární konstrukt



Přiřazení - zobrazení - funkce:

je to předpis = procedura, který říká jaký výsledek (výstup) je přiřazen k danému vstupu

funkce (filozoficky)

- bazální pojem: pojem funkce
- skoro vše, o čem přemýšlíme jsou funkce (až na P,N, časové okamžiky, individua)
- způsob, jak přemýšlíme: něco něčemu přiřazujeme, tj. základním konstruktem přemýšlení je funkce
- jsme zvyklí jednotliviny, v něčem si podobné, „sypat“ do samostatných kontejnerů a přemýšlet a vyjadřovat se pomocí těchto kontejnerů = typů

funkce jako procedura

- procesní pohled
- (parciální) funkce je „výpočetní“ resp. „vyhodnocovací“ pravidlo/procedura, které poskytne buď *nic* nebo *výsledek* z množiny M, jestliže jí na vstupu zadáme hodnoty parametrů z M_1x...xM_n
- *deklar* x_1,..., x_n::M_1,..., M_n *deklar* y::M (*tělo proc*)
- x_i ... formální parametry, y ... výsledek

jak zadat konkrétní funkci ?

- Máme dánu bázi **B**. Objekty nad **B** jsou
  - prvky množin tvořících **B** (bázových množin)
  - všechny funkce definované nad **B**
- Vezměme funkci f /(BA) a prvek a z A. Nechť f(a) = b, b je prvkem B. b je hodnotou f na a.
- Funkci f - objekt typu (BA) - a objekt a typu A lze použít jako způsob určení jiného objektu - objektu b/B.
- Jak sestrojit (zadat) konkrétní funkci? Sestrojit typ (množinu všech funkcí) umíme ...
- Jak obecně zadávat objekty, abychom s nimi mohli pracovat?

klíčový „pojem“: konstrukce

- návod k určitým intelektuálním krokům, pomocí kterých porozumíme významu jazykových výrazů, tj.
  - porozumíme tomu, co ony reprezentují
  - identifikujeme přesně to, co ony (někdy ne zcela jednoznačně) označují
- srovnej s KTO v přednášce č. 4

konstrukce v TIL s jtt (1)

- Termínem **T-konstrukce** rozumíme specifický způsob určení (zadání) nějakého T-objektu.
- Je-li objekt zadán nějakou konstrukcí, řekneme, že konstrukce konstruuje tento objekt.
- Konstrukce souvisí s „procedurou identifikace“ něčeho v rámci všeho ostatního
- POZOR: konstrukce objektu je zcela něco jiného než objekt sám.
- Nechť $\underline{f}(\underline{a}) = \underline{b}$. Z objektu $\underline{b}$ nelze žádným způsobem poznat, že byl zadán právě konstrukcí $\underline{f}(\underline{a})$.

konstrukce v TIL s jtt (2)

- TIL s jtt = TIL s jednoduchou teorií typů (původní varianta TIL, která byla v 80-tých létech použita jako základ metody HIT datového modelování)
- v TIL s jtt sama konstrukce není objektem zájmu, ale pouze prostředek k identifikaci objektů nad EB
- v tomto (jednodušším) pojetí v konstrukcích přímo vystupují objekty
- později (DM2) ukážeme korekci tohoto přístupu použitím TIL s rtt (= TIL s rozvětvenou teorií typů) zavedením zásadního triku - tzv. ***triviální konstrukce*** nebo-li běžněji ***konstrukce trivializace***
- odkaz na TIL s jtt budeme v dalším vynechávat

konstrukce (3)

- Nechť f(a) = b. Říkáme, že objekty f a a se **vyskytují** v konstrukci objektu b. Konstrukce jsou složeniny objektů.
- V konstrukci se některé objekty mohou vyskytnout víckrát.
- V konstrukcích vystupují přímo **objekty**, nikoli neinterpretované symboly, kterým by objekty byly přiřazovány nějakou interpretací. V tom spočívá **transparentní pojetí**.
- Jedním z cílů je reprezentace informací v počítačích: proto se pohybujeme v dualismu transparentní - formální.

konstrukce, valuace (4)

- **Nevlastní konstrukce**: neexistuje objekt, který by byl konstruován touto konstrukcí.
- Nechť f je nedefinovaná na a. Potom konstrukce f(a) nezadává žádný objekt, a říkáme, že tato konstrukce je nevlastní.
- Představa konstrukce jako procedury na počítači: dosazením konkrétních argumentů za formální parametry (**valuací**) procedura
  - buď něco „spočítá“ = konstruuje nějaký objekt
  - nebo abortuje = je při tomto dosazení nevlastní

proměnné, valuace (5)

- Nechť pro každý typ T nad **B** máme přiřazenu spočetnou množinu V^T tak, že V^T je disjunktní s každým typem nad **B** (speciálně tedy V^T je disjunktní s množinou T), a pro každé dva různé typy T_1, T_2 platí V^{T1} je disjunktní s V^{T2}. Prvky množiny V^T nazýváme **T-proměnnými**. **Proměnná** je T-proměnná pro nějaký typ T.
- Totální funkce *v* z množiny všech proměnných do množiny všech objektů se nazývá **valuace**, jestliže pro jakoukoli T-proměnnou x platí, že *v*x je T-objekt.
- valuace znamená dosazení hodnot za všechny proměnné při zachování „typové kázně“

konstrukce atomické (6)

- Nechť X je nějaký T-objekt. Tento objekt je zadán sám sebou. Říkáme, že X je **T-konstrukce** a nazýváme ji **T-atom**. Nechť *v* je libovolná valuace. T-konstrukce X **v-konstruuje** objekt X.
- Nechť X je nějaká T-proměnná. Jako taková může konstruovat libovolný T-objekt. Opět řekneme, že X je **T-konstrukce** a nazveme ji **T-atom**. Nechť *v* je libovolná valuace. T-konstrukce X **v-konstruuje** objekt *v*X.
- T-atom nemůže nikdy být nevlastní.

konstrukce „n-tice“
(tuple construction)

- A_i, i = 1,..., n, jsou (ne nutně různé) T_i-konstrukce
- Pak $(A_1, ..., A_n)$ je konstrukce konstruující objekty typu $(T_1, ..., T_n)$
- Nechť *v* je libovolná valuace:
  - konstrukce $(A_1, ..., A_n)$ je *v*-nevlastní, jestliže některé A_i je *v*-nevlastní
  - jsou-li všechny A_i *v*-vlastní, nechť $\underline{A}_i$ jsou jimi *v*-konstruované objekty: potom $(A_1, ..., A_n)$ *v*-konstruuje objekt $(\underline{A}_1, ... , \underline{A}_n)$

konstrukce projekce

- Nechť B je n-ticová konstrukce konstruující objekty typu $(T_1, ..., T_n)$, kde T_i již nejsou n-ticové typy.
- Potom $B_{(i)}$, i = 1, ..., n, jsou T_i-konstrukce (projekce na i-tou složku)
- Nechť *v* je libovolná valuace:
  - je-li B *v*-nevlastní, pak každé $B_{(i)}$ je *v*-nevlastní
  - v opačném případě B *v*-konstruuje objekt $\underline{B}/(T_1,...,T_n)$ a $B_{(i)}$ *v*-konstruuje objekt $\underline{B}_{(i)}/T_i$ -- tzv. i-tou složku objektu $\underline{B}$.

konstrukce aplikace (7)

- Nechť T, T_1, ... , T_n jsou jakékoli typy nad **B**. Nechť A je ($TT_1...T_n$)-konstrukce, B_1 je T_1-konstrukce, ..., B_n je T_n-konstrukce.
  - A je konstrukce, která zadává objekty, jež jsou funkce ($T \leftarrow (T_1,...,T_n)$)
  - B_i jsou konstrukce, jež zadávají objekty typu T_i, o kterých dále nic nevíme
- [$AB_1...B_n$] je **T-konstrukce** zvaná **aplikace** A na B_1, ..., B_n.
  - zadání funkce lze aplikovat na zadání jiných objektů
  - objekt (funkci) nemohu aplikovat na jiné objekty !!! (srov. s čím pracujeme např. v integrálním počtu)

konstrukce aplikace (8)

- Nechť *v* je libovolná valuace. Nechť A v-konstruuje (objekt) funkci $\underline{A}/(TT_1...T_n)$ a nechť pro každé *i*, B_i v-konstruuje objekt $\underline{B_i}/T_i$. Mohou nastat dva případy:
  - funkce $\underline{A}$ je na n-tici $(\underline{B_1}, ... , \underline{B_n})$ nedefinována: potom řekneme, že konstrukce $[AB_1...B_n]$ je **v-nevlastní**
  - funkce $\underline{A}$ na n-tici $(\underline{B_1}, ... , \underline{B_n})$ je definována: potom řekneme, že konstrukce $[AB_1...B_n]$ **v-konstruuje T-objekt**, který je hodnotou funkce $\underline{A}$ na n-tici $(\underline{B_1}, ... , \underline{B_n})$

konstrukce aplikace (9)

- V ostatních případech, když buď
  - neexistuje objekt v-konstruovaný A nebo
  - pro nějaké *i* neexistuje objekt v-konstruovaný B_i
  - (a stačí alespoň něco z toho)
- nelze konstrukci $[AB_1...B_n]$ přiřadit žádný v-konstruovaný objekt a říkáme, že je tato konstrukce v-nevlastní
- PŘÍKLADY
  - z matematiky
  - ze života

konstrukce aplikace (9a) --příklady

- sčítání v oboru reálných čísel :: ((Tim, Tim) → Tim)
  [+ 5 2] konstruuje Tim-objekt 7
  [+ 5 x] v-konstruuje Tim-objekt 5+x
- dělení v oboru reálných čísel :: ((Tim, Tim) → Tim)
  [: 6 2] konstruuje Tim-objekt 3
  [: 6 0] je v-nevlastní pro všechna v, nekonstruuje nic
  [+ 5 [: 6 0]] je rovněž v-nevlastní pro všechna v
- relace > v oboru reálných čísel :: ((Tim, Tim) → Bool)
  [> 6 2] konstruuje Bool-objekt Pravda
  [> 2 6] konstruuje Bool-objekt Nepravda
  [> 6 [: 6 0]] je v-nevlastní pro všechna v
  [> x 2] v-konstruuje Bool-objekt Pravda pro ty valuace v, které přiřadí x číslo větší než 2 a Nepravda pro všechny ostatní valuace v

konstrukce aplikace (9b) --příklady

- ZAM' / (Wrd → (Tim → (Univ → Bool)))
  [[[ZAM' w] t] Novák] konstruuje v daném světě w a čase t Pravda, jestliže individuum s „nálepkou" Novák je zaměstnancem, a konstruuje Nepravda, jestliže není;
  [[ZAM' w] t] konstruuje v daném světě w a čase t množinu individuí, která mají tu čest právě v tomto w a t býti zaměstnanci
- PlatZam / (Wrd → (Tim → (ZAM → PLAT)))
  [[PlatZam w] t] konstruuje tabulku (dvousloupcovou) platů zaměstnanců, která platí v daném světě w a čase t;
  [[[PlatZam w] t] Novák] konstruuje tu hodnotu z množiny PLAT, která je přiřazena předchozí tabulkou Novákovi;
  [[[[PlatZam w] t] Novák] 8000] konstruuje P nebo N podle toho, zda Novák má či nemá plat 8000.

konstrukce aplikace (9c) --příklady

- RektorMU / (Wrd → (Tim → Univ))
  [[RektorMU w] t] konstruuje Univ-objekt označený nálepkou, která je jménem rektora MU v daném w a daném čase t;
  v čase t=1912 a v aktuálním světě je v-nevlastní
  v čase t= 2001 konstruovala v aktuálním světě Jiřího Zlatušku
- RektorZDSBotanicka / (Wrd → (Tim → Univ))
  [[RektorZDSBotanicka w] t] je v-nevlastní pro všechny *v*
  (v žádném možném světě není žádné individuum rektorem ZDŠ - to je naše domluva na významu slova rektor a na významu slova ZDŠ)
- RektorZDS / (Wrd → (Tim → (Univ → Bool))
  [[RektorZDS w] t] v-konstruuje {} pro všechny valuace *v*

konstrukce abstrakce (10)

- Nechť T, T_1, ... , T_n jsou jakékoli typy nad **B**. Nechť A je T-konstrukce. Nechť x_1, ... , x_n jsou navzájem různé proměnné, x_i je T_i-proměnná, značíme $x_i :: T_i$.
- Definujeme, že „$\lambda\ x_1 ... x_n$ (A)“ je (TT_1 ... T_n)-konstrukce, zvaná (lambda-) abstrakce A vzhledem k proměnným x_1, ... , x_n.
- Všimněme si, že to připomíná zápis:
  procedure X_1 ... X_n (**tělo procedury)**,
  který z „holého“ algoritmu nějakého výpočtu vyrábí (opakovatelně použitelnou a jako celek použitelnou) funkci
- lambda-abstrakce vyrábí (objekty) funkce z n-tic (T_1,..,T_n) do T.

konstrukce abstrakce (11)

- Jaké objekty lambda-abstrakce konstruuje?
- Příprava na odpověď:
- Nechť *v* je libovolná valuace. Označme $v(x_1:=X_1, \ldots, x_n:=X_n)$, kde X_i jsou T_i-objekty, valuaci která se od *v* liší pouze tím, že proměnným x_i (i=1,...,n) přiřazuje objekty po řadě X_i/T_i. Všem ostatním proměnným přiřazuje to co valuace *v*.
- Nechme X_i probíhat všechny objekty z T_i, tj. postupně je „dosazujeme“ do $v(x_1:=X_1, \ldots, x_n:=X_n)$

konstrukce abstrakce (11a)

- Definujme funkci F / ($TT_1...T_n$) takto:
- F(X_1,..., X_n) je nedefinováno, jestliže A je *v*(x_1:=X_1,...,x_n:=X_n)-nevlastní
- F(X_1,..., X_n) je T-objekt *v*(x_1:=X_1,...,x_n:=X_n)-konstruovaný A, jestliže A není *v*(x_1:=X_1,...,x_n:=X_n)-nevlastní.

- Potom říkáme, že $\lambda\ x_1...x_n$ (A) v-konstruuje F

konstrukce abstrakce (12)

- lamda-abstrakce nemůže být nikdy nevlastní, může maximálně dávat všude nedefinovanou funkci
- to že $\lambda\, x_1...x_n\, (A)$ je $(TT_1...T_n)$-konstrukce se zapisuje $\lambda\, x_1...x_n\, (A) :: (TT_1...T_n)$
- říkáme, že je to $x_1,...,x_n$ - uzávěr otevřené konstrukce A
- **volná proměnná**: lze jí valuací *v* udělit libovolnou hodnotu
- **uzavřená konstrukce**: neobsahuje volné proměnné
- nechť A obsahuje nejvýše proměnné $x_1, ..., x_n$. Potom $\lambda\, x_1...x_n\, (A)$ je uzavřená konstrukce

konstrukce abstrakce - příklady

- otevřená konstrukce: x+5
  k ní uzavřená konstrukce: λx (x+5)
- *v* je valuace, která x přiřadí 3 a y přiřadí 4:

λy [> x y]	*v*-konstruuje (Bool Tim)-objekt -- třídu čísel menších než 3
λx [> x y]	*v*-konstruuje třídu čísel větších než 4
λxy [> x y]	konstruuje (nezávisle na *v*) relaci „>“ (Bool Tim Tim)-objekt
λx λy [> x y]	konstruuje funkci, která každému číslu m (dosazenému za x) přiřadí třídu čísel menších než m
λy λx [> x y]	konstruuje funkci, která každému číslu m (dosazenému za y) přiřadí třídu čísel větších než m

konstrukce abstrakce - příklady

- Nechť x::ZAM, y::PLAT, w::Wrd, t::Tim
  PlatZam::(((PLAT ZAM)Tim)Wrd)
- λw λt λx λy[[[[PlatZam w]t]x]y] konstruuje co?
- poněvadž datové funkce nad **EB** jsou (vzhledem k w, t) totální, lze uplatnit Shönfinkelovu redukci, a ekvivalentně psát:
  λwt λx λy[[[PlatZam (w, t)]x]y]
- tato konstrukce ale konstruuje funkce typu
  ((((Bool PLAT) ZAM)(Tim,Wrd))
- jak vyjádřit, že množina λy[[[PlatZam (w, t)]x]y]
  je pro dané (w,t) a dané x vždy jednoprvková ?
- funkce „singularizátor“ -- viz dále

podkonstrukce

- Nechť A je konstrukce.
- (1) A je podkonstrukce A
- (2) Je-li A tvaru $[BB_1...B_n]$,
  pak B, B_1, ..., B_n jsou podkonstrukce A.
- (3) Je-li A tvaru $\lambda\, x_1...x_n\, B$,
  pak B je podkonstrukce A.
- (4) Jeli A tvaru $(B_1, ..., B_n)$,
  pak B_1, ..., B_n jsou podkonstrukce A.
- (5) Je-li C podkonstrukce B a B je podkonstrukce A,
  pak C je podkonstrukce A.
- Nic jiného, než je řečeno v (1) až (5) není podkonstrukce

vázané výskyty a volné proměnné

- Mějme konstrukci $\lambda\, x_1 ... x_n$ (A) (1)

- všechny výskyty proměnných $x_1, ..., x_n$ v (1) jsou vázané výskyty těchto proměnných v konstrukci (1)
- Nechť C je podkonstrukcí B. Výskyty proměnných, které jsou vázanými výskyty v C, jsou vázanými výskyty v B.
  To platí pro všechna taková B, že C je podkonstrukcí B.
- Výskyty proměnných, které nejsou vázanými výskyty v dané konstrukci, nazýváme **volnými výskyty .**
- Proměnné, které mají alespoň jeden volný výskyt v A, nazýváme **volné proměnné** konstrukce A.

ekvivalence konstrukcí

- konstrukce C_1 je ekvivalentní s konstrukcí C_2, jestliže pro libovolnou valuaci *v* a libovolný objekt A platí C_1 v-konstruuje A právě tehdy, když C_2 v-konstruuje A .
- Srov. princip extenzionality
- ekvivalentní transformace konstrukcí /MaPaZla - str.78/
  pravidlo ***alfa***-redukce:
  Jestliže konstrukce C obsahuje vázané i volné výskyty proměnné x, pak všechny vázané výskyty x můžeme přejmenovat na libovolné jméno y takové, které se nevyskytuje v konstrukci C. Výsledkem je konstrukce ekvivalentní s C.
- transparentní formulace pravidla ***beta***-redukce z lambda kalkulu:

pravidlo ***beta***-redukce:

- konstrukce A::T nechť obsahuje volné proměnné $x_1::T_1,\ldots, x_n::T_n$
- $\lambda\, x_1 \ldots x_n(A) :: (T T_1 \ldots T_n)$
- definujeme operaci $A(x_1, \ldots, x_n := B_1, \ldots, B_n)$:
  - ta v konstrukci A nahradí každý výskyt proměnné x_i konstrukcí $B_i::T_i$
  - a je přípustná pouze tehdy, když nahrazení nevede k tomu, že některá z volných proměnných konstrukce B_i se po nahrazení stane vázanou proměnnou v B_i

pravidlo ***beta***-redukce - pokračování

- Potom pro každou valuaci *v*:
  konstrukce
  $A(x_1, ..., x_n := B_1, ..., B_n)$
  a konstrukce
  $[(\lambda\, x_1 ... x_n\, (A))B_1 ... B_n]$
  v-konstruují týž T-objekt nebo jsou obě
  v-nevlastní

otevřené a uzavřené konstrukce

- Nechť A je konstrukce, která obsahuje volné proměnné. Potom A se nazývá ***otevřená konstrukce***.
- Nechť A je konstrukce, která neobsahuje žádnou volnou proměnnou. Potom A se nazývá ***uzavřená konstrukce***.

- To co konstruují otevřené konstrukce, je vždy závislé na valuaci. Uzavřené konstrukce konstruují objekty nezávisle na valuaci.
- Uzavřené konstrukce jsou vhodné pro modelování těch objektů, které chceme vnímat jako celé funkce, tj. kde pracujeme s funkcí jako takovou, nikoli s jejími jednotlivými hodnotami v závislosti na konkrétních argumentech
- Otevřené konstrukce jsou vhodné pro modelování toho, co je přirozené vnímat jako závislé na konkrétní valuaci: databázový stroj (obecně stroj = „engine“) který umožňuje manipulovat s daty

PŘÍKLADY

– logických operátorů
– matematických funkcí
– konstrukce množin
– funkcí ze života
– jak je to s nevlastností ekvivalentních konstrukcí ?

kvantifikátory, singularizátor

- X::T, B je Bool- konstrukce
- Λ^T::((T $\rightarrow$ Bool) $\rightarrow$ Bool) je tzv. obecný kvantifikátor:
  [Λ^T λx B] = Pravda, když λx B konstruuje právě T, jinak,
  tj. když λx B konstruuje T′ $\subset$ T, je [Λ^T λx B] = Nepravda
- V^T::((T $\rightarrow$ Bool) $\rightarrow$ Bool) je tzv. existenční kvantifikátor:
  [V^T λx B] = Pravda, když λx B konstruuje neprázdnou
  podmnožinu T, jinak Nepravda
- I^T::((T $\rightarrow$ Bool) $\rightarrow$ T) je tzv. singularizátor:
  [I^T λx B] = t, když λx B konstruuje jednoprvkovou
  podmnožinu {t}$\subseteq$ T, jinak je nedefinován
- místo [Λ^T λx B] píšeme $\forall$x (B)
  místo [V^T λx B] píšeme $\exists$x (B)
  místo [I^T λx B] píšeme ιx (B)

reálný svět

- Mějme atribut (viz přednáška č. 3)
  OdbDodZbozi =
  odběratelé (#ODB) kterým daný dodavatel (#DOD)
  dodává dané zboží (#ZBOZI)
  :: (...((DOD, ZBOZI) → (ODB → Bool)))
- Jak vypadá jeho konstrukce:
  x::DOD, z::ZBOZI, y::ODB, w,t jako obvykle:

$$\lambda wt\ (\lambda xz\ (\lambda y\ ([[OdbDodZbozi_{wt}\ (x,z)]\ y])))$$

jak je to s nevlastností ekvivalentních konstrukcí ?

- Nechť C_1 je ekvivalentní s C_2. Nechť v je libovolná valuace.
- C_1 je v-nevlastní právě tehdy, když C_2 je v-nevlastní.

- Proveďte důkaz.

otázky

- co je to matematika ?
- je matematika více o objektech nebo o konstrukcích
- co nám více pomůže při domlouvání? objekty nebo konstrukce?
- jsou v reálném světě objekty? jsou v něm konstrukce?
- jak je to s objekty a konstrukcemi v ideálních světech: jeden IdSvet versus dva a více IdSvetu, vytváření společné kultury
- kultura lidí vs. kultura lidí + matematických strojů