INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Inteligentní systémy (TIL)

Přednáška 4

Marie Duží

http://www.cs.vsb.cz/duzi/

Pravidlo β-transformace

- Základní výpočtové pravidlo λ-kalkulů a funkcionálních programovacích jazyků
- určuje, jak provést operaci *aplikaci funkce f na argument a* za účelem získání hodnoty funkce *f* na *a*.

Př.: $[\lambda x\, [^0{+}\; x\; ^01]\; ^03]$ – chci hodnotu funkce následníka na čísle 3:

β-redukce (někdy také λ-*redukce*) „jménem“:

$[\lambda x\, [^0{+}\; x\; ^01]\; ^03] \Rightarrow [^0{+}\; ^03\; ^01]$ (= ^{04})

β-rozvinutí (nebo také λ-*rozvinutí*):

$[^0{+}\; ^03\; ^01] \Rightarrow [\lambda x\, [^0{+}\; x\; ^01]\; ^03]$

$[^0{+}\; ^03\; ^01] \Rightarrow [\lambda y\, [^0{+}\; ^03\; y]\; ^01]$

$[^0{+}\; ^03\; ^01] \Rightarrow [\lambda xy\, [^0{+}\; x\; y]\; ^03\; ^01]$

- Redukce obecně: $[[\lambda x_1 \ldots x_m\; Y]\; D_1 \ldots D_m] \vdash Y(D_i/x_i)$

β-conversion: $[\lambda x\ C(x)\ A] \vdash C(A/x)$

- *Procedure* of applying the function presented by $\lambda x\ C(x)$ to an argument presented by *A.*
- The fundamental computational rule of λ-calculi and functional programming languages
- The fundamental inference rule of HOL

‘**by name**’; the *procedure A* is substituted for all the occurrences of *x*

- **not operationally equivalent**

‘**by value**’; the *value* presented by *A* is substituted for all the occurrences of *x*

β-conversion: $[\lambda x\, C(x)\, A] \mid\!\!— C(A/x)$

- In programming languages the difference between ‘by value’ and ‘by name’ revolves around the programmer’s choice of *evaluation strategy*.
  - Algol’60: “call-by-value” and “call-by-name”
  - Java: manipulates objects “by name”, however, procedures are called “by-value”
  - Clean and Haskell: “call-by-name”
- Similar work has been done since the early 1970s; for instance, Plotkin (1975) proved that the two strategies are not operationally equivalent.
- *Chang & Felleisen (2012)’s call-by-need reduction by value.* But their work is couched in an untyped λ-calculus.

$[\lambda x\, C(x)\, A] \mid\!\!—\, C(A/x)$

- *Conversion by name* → three problems.

1. conversion of this kind is *not guaranteed to be an equivalent transformation* as soon as partial functions are involved.
2. even in those cases when β-reduction is an equivalent transformation, it can yield a *loss of analytic information* of which function has been applied to which argument
3. In practice *less efficient* than ‘by value’

Problems with β-reduction 'by name'

1) non-equivalence

$$[\lambda x\ [\lambda y\ [^0\!+\ x\ y]]\ [^0 Cotg\ ^0\pi]]$$

is an ***improper*** construction; it does not construct anything, because there is no value of the cotangent function at π

but its β-reduced Composition

$$[\lambda y\ [^0\!+\ [^0 Cotg\ ^0\pi]\ y]]$$

constructs a degenerate function

- The improper construction $[^0 Cotg\ ^0\pi]$ has been drawn into the intensional context of the Closure $[\lambda y\ [^0\!+\ x\ y]]$.

β-conversion by name:
2) loss of analytic information

$[\lambda x\ [x + 1]\ 3]$

β $\rightarrow$ $[3 + 1]$

$[\lambda y\ [3 + y]\ 1]$

which function has been applied to which argument?

No ‘backward path’. Does it matter?

Problems with β-reduction

2) Loss of analytic information

- "John loves his wife, and so does Peter"
  → **exemplary husbands** (sloppy reading)
- "loving one's own wife" vs. "loving John's wife"

$L^{own}\ (John)$: $\lambda w \lambda t\ [\lambda x\ [{}^0Love_{wt}\ x\ [{}^0Wife\_of_{wt}\ \mathbf{x}]]\ {}^0John]$

$L^{John}\ (John)$: $\lambda w \lambda t\ [\lambda x\ [{}^0Love_{wt}\ x\ [{}^0Wife\_of_{wt}\ {}^0\mathbf{John}]]\ {}^0John]$

Both β-reduce to $\boldsymbol{L^{John}\ (John)}$:

$\lambda w \lambda t\ [{}^0Love_{wt}\ {}^0John\ [{}^0Wife\_of_{wt}\ {}^0\mathbf{John}]]$

- "so does Peter"
- Peter loves *John's* wife → **trouble on the horizon**

β-conversion by name: loss of info

(1) $\lambda w \lambda t\, [\lambda x\, [{}^0Love_{wt}\; x\; [{}^0Wife\_of_{wt}\; {}^0John]]\; {}^0John]$

(2) $\lambda w \lambda t\, [\lambda x\, [{}^0Love_{wt}\; x\; [{}^0Wife\_of_{wt}\; x]]\; {}^0John]$

(3) $\lambda w \lambda t\, [{}^0Love_{wt}\; {}^0John\; [{}^0Wife\_of_{wt}\; {}^0John]]$

It is uncontroversial that the contractum (3) can be equivalently expanded back both to (1) and (2).

The problem is, of course, that there is no way to reconstruct *which* of (1), (2) would be the correct redex

Does it matter?

- HOL tools are broadly used in automatic theorem checking and applied as interactive proof assistants.
- The underlying logic is usually a version of *simply typed* λ-calculus of *total functions*.
- However, there is another application → *natural language processing* → hyperintensional logic is needed so that the underlying *inference machine is neither over-inferring* (that yields inconsistencies) *nor under-inferring* (that causes lack of knowledge).
- agents' attitudes like knowing, believing, seeking, solving, designing, etc., because attitudinal sentences are part and parcel of our everyday vernacular.

Hyperintensionality

- was born out of a negative need, to block invalid inferences
  - Carnap (1947, §§13*ff*); there are contexts that are neither extensional nor intensional (attitudes)
  - Cresswell; any context in which substitution of necessary equivalent terms fails is hyperintensional
- Yet, which inferences are valid in hyperintensional contexts?
- How hyper are hyperintensions? → procedural isomorphism
- Which contexts are hyperintensional?
- *TIL definition is positive*: a context is *hyperintensional* if the very meaning *procedure* is an object of predication; TIL is a ***hyperintensional, partial typed*** λ***-calculus***

β-*reduction by value*

$$[\lambda x\ C(x)\ A] \vdash C(A/x)$$

underspecified:

- How to execute *C*(*A*/*x*)?

a) 'by name': *construction A* is substituted for *x* → problems

b) 'by value': execute *A* first, and only if it does not fail, substitute the produced *value* for *x* – *substitution method* → bingo, no problems !!! ☺

Substitution 'by value'

$$[\lambda x\, F(x)\, A] = {}^2[{}^0Sub\, [{}^0Tr\, A]\, {}^0x\, {}^0F(x)]$$

1. *A: execute A in order to obtain the value a; if A* is *v-improper, then the whole Composition is v-improper (stop); else*:
2. $[{}^0Tr\, A]$: *obtain Trivialization of ("pointer at"*) the argument *a*
3. $[{}^0Sub\, [{}^0Tr\, A]\, {}^0x\, {}^0F]$: *substitute this Trivialization for x into 'the body' F*
4. ${}^2[{}^0Sub\, [{}^0Tr\, A]\, {}^0x\, {}^0F]$: *execute the result*

Substitution 'by value'

$Sub/(*_n *_n *_n *_n)$ operuje na *konstrukcích* takto:

$[^0Sub\ C_1\quad C_2\quad C_3]$

co za_co kam

Nechť C_1 *v-konstruuje konstrukci* D_1,

C_2 *v-konstruuje konstrukci* D_2,

C_3 *v-konstruuje konstrukci* D_3,

konstruuje konstrukci *D*, která vznikne korektní substitucí D_1 za D_2 do D_3

$Tr/(*_n\ \alpha)$ *v-konstruuje Trivializaci* α*-objektu*

$[^0Tr\ x]$ *v-konstruuje Trivializaci objektu v-konstruovaného proměnnou x, x je* ***volná***

^0x *konstruuje x* bez ohledu na valuaci, proměnná *x* je **o-vázaná**

Substitution 'by value'

Příklad

$[{}^0Sub\ [{}^0Tr\ {}^0\pi]\ {}^0x\ {}^0[{}^0Sin\ x]]$

konstruuje *konstrukci* $[{}^0Sin\ {}^0\pi]$

${}^2[{}^0Sub\ [{}^0Tr\ {}^0\pi]\ {}^0x\ {}^0[{}^0Sin\ x]]$

konstruuje *hodnotu* funkce *Sinus* na π,
tj. číslo 0

$[{}^0Sub\ [{}^0Tr\ y]\ {}^0x\ {}^0[{}^0Sin\ x]]$

$v(\pi/y)$-konstruuje *konstrukci* $[{}^0Sin\ {}^0\pi]$

Substitution method; *broadly applied*

- Application of a function to an argument (β-reduction by value)
- Existential quantification into hyperintensional contexts
- Hyperintensional attitudes *de re*
- Anaphoric preprocessing
- Topic/focus articulation; presuppositions; active vs. passive

Substitution method; *broadly applied*

de re attitudes

*Tilman believes **of** the Pope that **he** is wise*

$\lambda w \lambda t\ [{}^0Believe_{wt}\ {}^0Tilman\ {}^2[{}^0\mathbf{Of}\ [{}^0Tr\ {}^0Pope_{wt}]\ {}^0he\ {}^0[\lambda w^* \lambda t^*\ [{}^0Wise_{w^*t^*}\ he]]]$

Of = *Sub* operates on the (hyper)intensional context of “that he is wise”

Substitution method; *broadly applied*

Quantifying into …

- Tom is seeking the last decimal of π

There is a number such that Tom is seeking its last decimal

- $\lambda w \lambda t\ [{}^0 Seek^*_{wt}\ {}^0 Tom\ {}^0[{}^0 Last\_Dec\ {}^0\pi]]$

$\lambda w \lambda t\ [{}^0\exists \lambda x\ [{}^0 Seek^*_{wt}\ {}^0 Tom\ [{}^0 Sub\ [{}^0 Tr\ x]\ {}^0 y\ {}^0[{}^0 Last\_Dec\ y]]]]$

How hyper are hyperintensions → procedural isomorphism

- Maybe that it is philosophically wise to adopt several notions of procedural isomorphism.
- It is not improbable that several degrees of hyperintensional individuation are called for, depending on which sort of discourse happens to be analysed.
- What appears to be synonymous in an ordinary vernacular might not be synonymous in a professional language like the language of logic, mathematics, computer science or physics.

Procedural isomorphism

- Ordinary vernacular – no variables →

(A1'''): α-conversion + β-conversion by value

+ *restricted* β-conversion by name;

$[\lambda x\ [{}^0{+}\ x\ {}^0 0]\ y] \rightarrow [{}^0{+}\ y\ {}^0 0]$

+ pairs of simple synonyms

- Programming language – variables matter →

(A0'): α-conversion + pairs of simple synonyms