EVALUATION OF OPEN EXPRESSIONS

Per Martin-Löf

Symposium on Programming, Types, and Languages dedicated to Bengt Nordström in connection with his 60th birthday, Chalmerska Huset, Göteborg, 19 March 2009

Compute the value of x^2+3y!

1) Does not make sense, because no values have been assigned to the variables x and y

2) $x \cdot x + ((y+y)+y)$

---

ordinary computation

vs.

normalization

The informal, or intuitive, semantics of type theory makes it evident that closed expressions of ground type evaluate to head normal form.

Metamathematics, either the method of computability or the method of normalization by evaluation, is needed to show that expressions which are open or of higher type can be reduced to normal form.

Main question: Would it be possible to modify the informal semantics in such a way that it becomes evident that all expressions, also those which are open or of higher type, can be reduced to full normal form?

Comparison with the notion of general recursive function

e Gödel number

$(\forall m)(\exists n)\, T(e,m,n)$ true

$$\begin{cases} f : N \to N \\ (\forall m)\, T(e,m,f(m)) \text{ true} \end{cases}$$

Two ways of evaluating the general recursive function with Gödel number e for the argument m: Either use the system of equations encoded by e with input m, or evaluate $U(f(m))$!

Heyting

Skolem

Copenhagen 2008

Method of computability
convertibility (Tait)
reducibility (Girard)

Gödel 1941 CW, Vol. III, p. 188
Tait 1967 JSL
M-L 1971 2nd Scand. Logic Symp. (Oslo 1970)

The method of computability was turned into an intuitionistic model construction in

M-L 1975 3rd Scand. Logic Symp. Uppsala 1973
1975 Logic Colloquium '73 Bristol

$A \longrightarrow A^* = A^{*I}, A^{*II}$

A^{*I}: normal formula; A^{*II}: computability predicate

$a \longrightarrow a^* = a^{*I}, a^{*II}$

a^{*I}: normal term of type A^{*I}; a^{*II}: proof of $A^{*II}(a^{*I})$

Normalization by evaluation

U. Berger & H. Schwichtenberg, An inverse of the evaluation functional for typed λ-calculus, 1991

---

a closed

$nf(\ulcorner a \urcorner) = \downarrow [\![\ulcorner a \urcorner]\!] = \downarrow a^*$

$[\![\ulcorner a \urcorner]\!] = a^*$ semantic object

$\downarrow$ = reification

$a = a(x_1, \dots, x_n)$ open

$$nf(\ulcorner a \urcorner) = \downarrow [\![\ulcorner a \urcorner]\!](\ulcorner x_1 \urcorner = c^*_{x_1}, \dots, \ulcorner x_n \urcorner = c^*_{x_n})$$
$$= \downarrow \underbrace{a^*(x_1^* = c^*_{x_1}, \dots, x_n^* = c^*_{x_n})}_{r}$$
$$a^*(c^*_{x_1}, \dots, c^*_{x_n})$$

$a^* = a^*(x_1^*, \dots, x_n^*)$

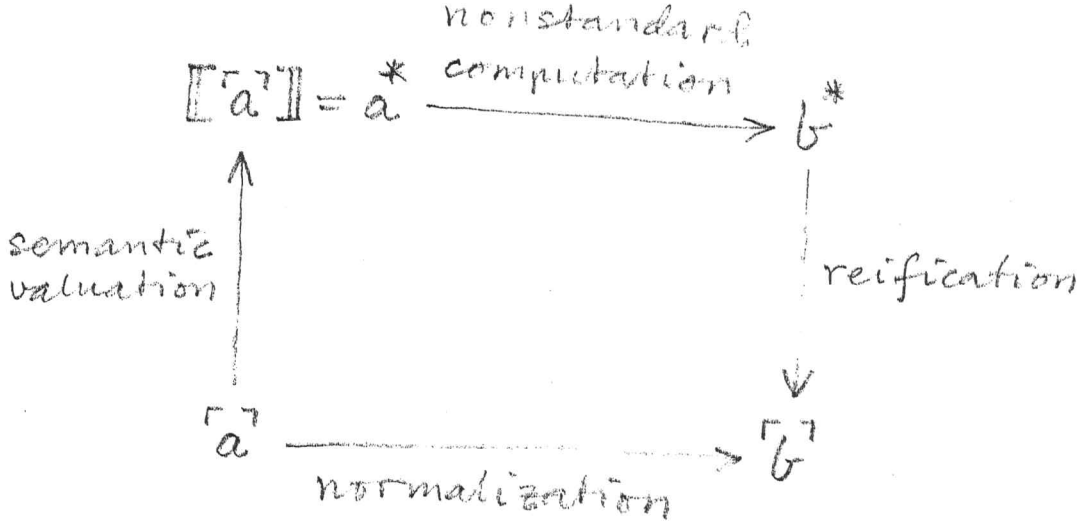
Using the method of computability made into an intuitionistic model construction, we can put

$$nf(\ulcorner a \urcorner) = [\![\ulcorner a \urcorner]\!]' = a^{*\prime}$$

Thus, with this choice of model, reification is defined by

$$\downarrow = {}' = \text{left projection}$$

When the method of computability is formulated as an intuitionistic model construction, it becomes an example of normalization by evaluation, although the model it uses is different from the one used by Ulrich Berger and Helmuth Schwichtenberg.



Reification rules

$$\frac{\alpha : \mathrm{dtype}^*}{\downarrow\alpha : \mathrm{ntype}} \qquad \frac{a : \mathrm{obj}^*(\alpha)}{\downarrow a : \mathrm{nterm}(\downarrow\alpha)}$$

$$\downarrow \mathrm{set}^* = \ulcorner \mathrm{set} \urcorner$$

$$\downarrow \mathrm{elem}^*(A) = \ulcorner \mathrm{elem} \urcorner(\downarrow A)$$

$$\downarrow \mathrm{fun}^*(\alpha, (x^*)\beta) = \ulcorner \mathrm{fun} \urcorner(\downarrow\alpha, \ulcorner x \urcorner \downarrow \beta(x^* = c_x^*))$$

$$\downarrow (x^*)b = \ulcorner x \urcorner \downarrow b(x^* = c_x^*)$$

or

$$\downarrow b = \ulcorner x \urcorner \downarrow b(c_x^*)$$

$$\downarrow c^*(a_1, \dots, a_n) = \ulcorner c \urcorner(\downarrow a_1, \dots, \downarrow a_n)$$

$$\downarrow c_x^*(a_1, \dots, a_n) = \ulcorner x \urcorner(\downarrow a_1, \dots, \downarrow a_n)$$

$$\downarrow f^*(a_1, \dots, a_n, a) = \ulcorner f \urcorner(\downarrow a_1, \dots, \downarrow a_n, \downarrow a)$$

↑ neutral

Ex. $(\lambda y)(x + s(y))$ open

↑ free variable parameter

$$\mathrm{nf}(\ulcorner (\lambda y)(x + s(y)) \urcorner)$$

$$= \downarrow [\![\ulcorner (\lambda y)(x + s(y)) \urcorner]\!](\ulcorner x \urcorner = c_x^*)$$

$$= \downarrow \lambda^*((y^*)(x^* +^* s^*(y^*)))(x^* = c_x^*)$$

$$= \downarrow \lambda^*((y^*)(c_x^* +^* s^*(y^*)))$$

$$= \ulcorner \lambda \urcorner(\downarrow(y^*)(c_x^* +^* s^*(y^*)))$$

$$= \ulcorner \lambda \urcorner(\ulcorner y \urcorner \downarrow(c_x^* +^* s^*(c_y^*)))$$

$$= \ulcorner \lambda \urcorner(\ulcorner y \urcorner \downarrow s^*(c_x^* +^* c_y^*))$$

$$= \ulcorner \lambda \urcorner(\ulcorner y \urcorner \ulcorner s \urcorner(\downarrow(c_x^* +^* c_y^*)))$$

$$= \ulcorner \lambda \urcorner(\ulcorner y \urcorner \ulcorner s \urcorner(\downarrow c_x^* \ulcorner + \urcorner \downarrow c_y^*))$$

$$= \ulcorner \lambda \urcorner(\ulcorner y \urcorner \ulcorner s \urcorner(\ulcorner x \urcorner \ulcorner + \urcorner \ulcorner y \urcorner))$$

Cf. the following direct computation of the open expression $(\lambda y)(x+s(y))$!

$\lambda((y)(x+s(y)))$

$= \lambda((y)(x+s(y)))$

$= \lambda((y)(x+s(y)))$

$= \lambda((y)\, s(x+y))$

$= \lambda((y)\, s(x+y))$

$= \lambda((y)\, s(x+y))$

$= \lambda((y)\, s(x+y))$

$\underbrace{c_{x_1} : \alpha_1, \ldots, c_{x_m} : \alpha_m}_{\Gamma}, \underbrace{x_{m+1} : \alpha_{m+1}, \ldots, x_n : \alpha_n}_{\Phi}$

Γ context of constants

Φ context of variables

$\Gamma = * =$ world

$\Gamma \leq \Delta$ accessibility relation

$$\begin{cases} \alpha : type^*(\Phi) \\ \alpha = \beta : type^*(\Phi) \\ a : obj^*(\alpha)(\Phi) \\ a = b : obj^*(\alpha)(\Phi) \end{cases}$$

$$\begin{cases} \alpha : type(\Gamma; \Phi) \\ \alpha = \beta : type(\Gamma; \Phi) \\ a : obj(\alpha)(\Gamma; \Phi) \\ a = b : obj(\alpha)(\Gamma; \Phi) \end{cases}$$

hybrid system theory

~~$\mathcal{E}_x$~~ x

~~$T, x \in T$~~ T, x

The four forms of judgement

$$\begin{cases} \alpha : \mathrm{type}\ (T) \\ \alpha = \beta : \mathrm{type}\ (T) \\ a : \mathrm{obj}(\alpha)\ (T) \\ a = b : \mathrm{obj}(\alpha)\ (T) \end{cases}$$

need new meaning explanations validating the thinning, or monotonicity, rules

$$\frac{\alpha : \mathrm{type}\ (T) \quad T \leq \Delta}{\alpha : \mathrm{type}\ (\Delta)}$$

etc.

$$\frac{\alpha : \mathrm{type}\ (T) \quad \beta : \mathrm{type}\ (T, x : \alpha)}{\mathrm{fun}(\alpha, (x)\beta) : \mathrm{type}\ (T)}$$

$(x : \alpha)\beta$

$f : \mathrm{fun}(\alpha, (x)\beta)\ (T)$ means that, for $\Delta \geq T$, $f(a) : \beta(x = a)(\Delta)$ provided that $a : \alpha(\Delta)$, and $f(a) = f(c) : \beta(x = a)(\Delta)$ provided that $a = c : \alpha(\Delta)$.

$$\frac{f : \mathrm{fun}(\alpha, (x)\beta)\ (T) \quad T \leq \Delta}{f : \mathrm{fun}(\alpha, (x)\beta)\ (\Delta)}$$

$$\frac{f : \mathrm{fun}(\alpha, (x)\beta)\ (T) \quad a : \alpha(T)}{f(a) : \beta(x = a)(T)}$$

$$\frac{f : \mathrm{fun}(\alpha, (x)\beta)\ (T) \quad a = c : \alpha(T)}{f(a) = f(c) : \beta(x = a)(T)}$$

Coquand 1985
Mitchell & Moggi 1987
Coquand & Gallier 1990

Three meaning explanations for the form of judgement
$f : \mathrm{fun}(\alpha, (x)\beta)(T)$

1) $f\gamma(a) : \beta(\gamma, x = a)$
for $\gamma : T$ and $a : \alpha\gamma$

2) $fp_T^\Delta(a) : \beta(p_T^\Delta, x = a)(\Delta)$
for $\Delta \geq T$ and $a : \alpha p_T^\Delta(\Delta)$

3) $f\gamma(a) : \beta(\gamma, x = a)$
for any Δ, $\gamma : \Delta \to T$, $a : \alpha\gamma(\Delta)$

The condition 3) includes 1) and 2) as special cases. Also, 2) is related to 3) in the same way as lawless sequences are related to choice sequences.

$X : (x_1 : \alpha_1, \ldots, x_n : \alpha_n)\mathrm{set}$
one of the variable declarations in the context T

$$\frac{a_1 : \alpha_1(T) \quad \vdots \quad a_n : \alpha_n(x_1 = a_1, \ldots, x_{n-1} = a_{n-1})(T)}{\underbrace{X(a_1, \ldots, a_n)}_{\text{neutral}} : \mathrm{set}(T)}$$

$x : (x_1 : \alpha_1, \ldots, x_n : \alpha_n)A$

$$\frac{a_1 : \alpha_1(T) \quad \vdots \quad a_n : \alpha_n(x_1 = a_1, \ldots, x_{n-1} = a_{n-1})(T)}{\underbrace{x(a_1, \ldots, a_n)}_{\text{neutral}} : A(x_1 = a_1, \ldots, x_n = a_n)(T)}$$

$$\frac{\begin{array}{l} a_1 : \alpha_1\,(T) \\ \quad\vdots \\ a_n : \alpha_n(x_1 = a_1, \ldots, x_{n-1} = a_{n-1})\,(T) \\ a : A(x_1 = a_1, \ldots, x_n = a_n)\,(T)\ \text{neutral} \end{array}}{\underbrace{F(a_1, \ldots, a_n, a)}_{\text{neutral}} : \text{set}\,(T)}$$

Ex. $T(a) : \text{set}\,(T)$ is neutral for neutral $a : U\,(T)$.

$$\frac{\begin{array}{l} a_1 : \alpha_1\,(T) \\ \quad\vdots \\ a_n : \alpha_n(x_1 = a_1, \ldots, x_{n-1} = a_{n-1})\,(T) \\ a : A(x_1 = a_1, \ldots, x_n = a_n)\,(T)\ \text{neutral} \end{array}}{\underbrace{f(a_1, \ldots, a_n, a)}_{\text{neutral}} : C(x_1 = a_1, \ldots, x_n = a_n, x = a)\,(T)}$$

Ex. $a + b : N\,(T)$ is neutral for neutral $b : N\,(T)$.

Old and new parsing of the basic forms of judgement

space ↓

$$\begin{cases} \alpha : \text{type}\,|\,(T) \\ \alpha = \beta : \text{type}\,|\,(T) \\ a : \text{obj}(\alpha)\,|\,(T) \\ a = b : \text{obj}(\alpha)\,|\,(T) \end{cases}$$

$$\begin{cases} \alpha \mid \text{type}(T) \\ \alpha = \beta \mid \text{type}(T) \\ a \mid \text{obj}(\alpha)(T) \\ a = b \mid \text{obj}(\alpha)(T) \end{cases}$$

subject/predicate gramm.

object/category ont.