1 Wstęp

1.1 Skąd pomysł?

Bezpieczeństwo

Pierwotny pomysłodawca: Werner Damm w 1982.

Wprowadzone przez Knapika, Niwińskiego i Urzyczyna na Conference on Foundations of Software Science and Computation Structures (FoSSaCS) 2002, na seminarium o algorytmach na nieskończonych drzewach generowanych przez gramatyki wyższego rzędu.

1.2 Podstawowe definicje

Typy w rachunku λ

Definicja 1. T jest typem wtw.

$$T ::= o \mid T \to T$$

gdzie o nazywamy typem bazowym.

Notacja

Definicja 2.

$$A \to B \to C := A \to (B \to C)$$

Definicja 3.

$$A^n \to B = \begin{cases} B & \mid n = 0 \\ A \to (A^{n-1} \to B) & \mid n > 0 \end{cases}$$

Definicja 4.

$$(A_1, A_2, \ldots, A_n) := A_1 \to A_2 \to \ldots \to A_n$$

gdzie A_n niekoniecznie jest typem bazowym.

Notacja

Definicja 5.

$$\begin{aligned} 0 &\equiv o \\ (k+1) &\equiv k \to o \end{aligned}$$

Przykład 6.

$$\begin{array}{rcl} 0 & \equiv & o \\ 1 & \equiv & o \to o \\ 2 & \equiv & (o \to o) \to o \\ 3 & \equiv & ((o \to o) \to o) \to o \\ & \vdots & \end{array}$$

Rząd typu

Definicja 7. $ord(T)$ jest rzędem typu T wtw.

$$ord(T) = \begin{cases} 0 & | \; T = o \\ \max\{ord(A) + 1, ord(B)\} & | \; T = A \to B \end{cases}$$

Drzewo typu

Definicja 8. $tree(T)$ jest drzewem typu T wtw.

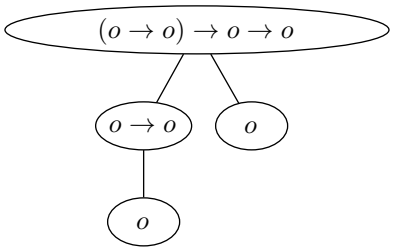
$$tree(T) := \begin{cases} o & | \; T = o \\ \begin{array}{c} \quad tree(B) \\ \diagup \\ tree(A) \end{array} & | \; T = A \to B \end{cases}$$

Spostrzeżenie 9. $ord(T) = height(tree(T))$

Drzewo typu

Przykład 10.

$tree((o \to o) \to (o \to o)) =$



Typ homogeniczny

Definicja 11. Typ $T = A_1 \to A_2 \to \ldots \to A_n \to o$ jest homogeniczny wtw. $ord(A_1) \geq ord(A_2) \geq \ldots \geq ord(A_n)$

Podstawienie capture-permitting

Definicja 12. $M\{N/x\}$ jest podstawieniem capture-permitting N pod x w M wtw.

$$\begin{array}{rcll} x\{N/x\} & \equiv & N & \\ y\{N/x\} & \equiv & y & \mid x \not\equiv y \\ (M_1M_2)\{N/x\} & \equiv & (M_1\{N/x\})(M_2\{N/x\}) & \\ (\lambda x.M)\{N/x\} & \equiv & \lambda x.M & \\ (\lambda y.M)\{N/x\} & \equiv & \lambda y.M\{N/x\} & \mid x \not\equiv y \end{array}$$

Przykład 13.

$$\begin{array}{rcl} \lambda y.x\{y/x\} & \equiv & \lambda y.y \\ \lambda z.x\{y/x\} & \equiv & \lambda z.y \end{array}$$

Podstawienie

Definicja 14. $M[N/x]$ jest podstawieniem N pod x w M wtw.

$$\begin{array}{rcll} x[t/x] & \equiv & t & \\ y[t/x] & \equiv & y & \mid x \not\equiv y \\ (M_1M_2)[t/x] & \equiv & (M_1[t/x])(M_2[t/x]) & \\ (\lambda x.M)[t/x] & \equiv & \lambda x.M & \\ (\lambda y.M)[t/x] & \equiv & \lambda y.M[z/y][t/x] & \mid x \not\equiv y \wedge z : \text{nowa zmienna} \end{array}$$

Przykład 15.

$$\begin{array}{rcl} \lambda y.x[y/x] & \equiv & \lambda z'.y \\ \lambda z.x[y/x] & \equiv & \lambda z'.y \end{array}$$

2 Bezpieczeństwo

2.1 Bezpieczne gramatyki

Bezpieczna gramatyka

Definicja 16. Gramatyka wyższego rzędu G o homogenicznych typach niterminali jest niebezpieczna wtw. istnieje produkcja $Fz_1z_2 \ldots z_n \to e$, gdzie e zawiera podterm, który:

- pojawia się w roli argumentu;
- zawiera parametr typu o niższym rzędzie, niż on sam.

Bezpieczna gramatyka

Definicja 17 (równoważna)**.** Zakładając homogeniczność typów:

- term t rzędu k jest niebezpieczny wtw. zawiera parametr rzędu $< k$;
- wystąpienie niebezpiecznego termu t w termie t' jest bezpieczne wtw. t pojawia się w kontekście $\ldots(ts)\ldots$;
- gramatyka jest bezpieczna wtw. żaden niebezpieczny term nie pojawia się w sposób niebezpieczny po prawej stronie żadnej produkcji.

Bezpieczna gramatyka

Przykład 18.

$$\begin{array}{rl} f & : (o,o,o) \\ g,h & : (o,o) \\ a,b & : o \end{array}$$

$$\begin{array}{rcl} S^o & \to & Fgab \\ F^{((o,o),o,o,o)}\phi^{(o,o)}x^o y^o & \to & f(F(\underline{F\phi x})y(hy))(f(\phi x)y) \end{array}$$

$$\begin{array}{rcl} S^o & \to & G(ga)b \\ G^{(o,o,o)}z^o y^o & \to & f(G(Gzy)(hy))(fzy) \end{array}$$

$F\phi x$ jest niebezpieczny, bo pojawia się w roli argumentu do F i posiada x^o, gdy sam jest typu (o,o).

2.2 Bezpieczny rachunek λ

Bezpieczny rachunek λ

Definicja 19. System w rachunku λ jest bezpieczny wtw. zmienne wolne pojawiające się w termie mają rząd niemniejszy, niż sam term.

Przykład 20 (termy Kiersteada).

$$\begin{array}{rcl} M_1 & = & \lambda f^{((o,o),o)}.f(\lambda x^o.f(\lambda y^o.y)) \\ M_2 & = & \lambda f^{((o,o),o)}.f(\lambda x^o.f(\underline{\lambda y^o.x})) \end{array}$$

Bezpieczny rachunek λ

Definicja 21.

$$(\mathsf{var})\ \frac{}{x : A \vdash_{\mathsf{s}} x : A} \qquad (\mathsf{const})\ \frac{}{\vdash_{\mathsf{s}} f : A}\ f \in \Xi \qquad (\mathsf{wk})\ \frac{\Gamma \vdash_{\mathsf{s}} M : A}{\Delta \vdash_{\mathsf{s}} M : A}\ \Gamma \subset \Delta$$

$$(\mathsf{app}_{\mathsf{as}})\ \frac{\Gamma \vdash_{\mathsf{asa}} M : A \to B \quad \Gamma \vdash_{\mathsf{s}} N : A}{\Gamma \vdash_{\mathsf{asa}} M\,N : B} \qquad (\delta)\ \frac{\Gamma \vdash_{\mathsf{s}} M : A}{\Gamma \vdash_{\mathsf{asa}} M : A}$$

$$(\mathsf{app})\ \frac{\Gamma \vdash_{\mathsf{asa}} M : A \to B \quad \Gamma \vdash_{\mathsf{s}} N : A}{\Gamma \vdash_{\mathsf{s}} M\,N : B} \qquad ord\,B \leq ord\,\Gamma$$

$$(\mathsf{abs})\ \frac{\Gamma, x_1 : A_1, \ldots, x_n : A_n \vdash_{\mathsf{asa}} M : B}{\Gamma \vdash_{\mathsf{s}} \lambda x_1^{A_1} \ldots x_n^{A_n}.M : (A_1, \ldots, A_n, B)} \qquad ord(A_1, \ldots, A_n, B) \leq ord\,\Gamma$$

Bezpieczny rachunek λ

Przykład 22.

$$M_1 = \lambda f^2.f(\lambda x^0.f(\lambda y^0.y))$$

$$\dfrac{\dfrac{\dfrac{}{f : 2 \vdash_{\mathsf{s}} f : 2}(\mathsf{var})}{f : 2 \vdash_{\mathsf{asa}} f : 2}(\delta) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f : 2 \vdash_{\mathsf{s}} f : 2}(\mathsf{var})}{f : 2, x : 0 \vdash_{\mathsf{s}} f : 2}(\mathsf{wk})}{f : 2, x : 0 \vdash_{\mathsf{asa}} f : 2}(\delta) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{y : 0 \vdash_{\mathsf{s}} y : 0}(\mathsf{var})}{y : 0 \vdash_{\mathsf{asa}} y : 0}(\delta)}{y : 0 \vdash_{\mathsf{s}} \lambda y^0.y : 1}(\mathsf{abs})}{f : 2, x : 0 \vdash_{\mathsf{s}} \lambda y^0.y : 1}(\mathsf{wk})}{f : 2, x : 0 \vdash_{\mathsf{s}} f(\lambda y^0.y) : 0}(\mathsf{app})}{f : 2 \vdash_{\mathsf{s}} \lambda x^0.f(\lambda y^0.y) : 1}(\delta)}{f : 2 \vdash_{\mathsf{s}} f(\lambda x^0.f(\lambda y^0 : y)) : 0}(\mathsf{app}) \qquad \dfrac{f : 2 \vdash_{\mathsf{s}} f(\lambda x^0.f(\lambda y^0 : y)) : 0}{f : 2 \vdash_{\mathsf{asa}} f(\lambda x^0.f(\lambda y^0 : y)) : 0}(\delta)}{\vdash_{\mathsf{s}} \lambda f^2.f(\lambda x^0.f(\lambda y^0.y)) : 3}(\mathsf{abs})$$

Prawie-bezpieczeństwo

Definicja 23. Term t jest prawie bezpieczny wtw. t jest prawie bezpieczną aplikacją lub termem w formie $\lambda x_1^{A_1} x_2^{A_2} \ldots x_n^{A_n}.M$ gdzie $n \geq 1$ oraz M jest prawie bezpieczną aplikacją.

Spostrzeżenie 24. Termy prawie bezpieczne nie są koniecznie bezpieczne, ale mogą się takie stać. Czy to przez abstrakcję zmiennych, czy przez aplikację.

Bezpieczny rachunek λ

Lemat 25 (o nieprzechwytywaniu zmiennych)**.** *Jeśli* $\Gamma, x : B \vdash_{\mathsf{s}} M : A$ *oraz* $\Gamma \vdash_{\mathsf{s}} N : B$*, nie zajdzie przechwytywanie zmiennych wykonując capture-permitting podstawienie* $M\,[N/x]$.

Dowód Udowodnimy to indukcją po strukturze M.

(var), (const) i (app) są trywialne.

(abs) Załóżmy, że $M \equiv \lambda\overline{y}.R$, gdzie $\overline{y} = y_1 \ldots y_p$. Jeśli $x \in \overline{y}$, $M\,[N/x] = M$, czyli nic się nie dzieje.

No to załóżmy, że $x \notin \overline{y}$. Z kwestii o prawie-bezpieczeństwie, $R = M_1 \ldots M_m$ dla jakiegoś $m \geq 1$. M_1 nie jest aplikacją, $\forall_i M_i$ jest bezpieczne. Tak więc $M\,[N/x] \equiv$

$\lambda\overline{y}.M_1\,[N/x]\ldots M_m\,[N/x]$. Jako że przechwycenie może się wydarzyć jedynie, gdy obydwa poniższe warunki są spełnione:

1. x jest wolne w M_i
2. jakaś zmienna y_i dla $1 \leq i \leq p$ jest wolne w N.

to z niezmiennika, jeśli zachodzi 2., znaczy że $ord(y_i) \geq ord(N) = ord(x)$, a skoro $x \notin \overline{y}$, warunek 1. implikuje, że x jest wolne w $\lambda\overline{y}.R$, więc znowu, $ord(x) \geq ord(\lambda\overline{y}.R) \geq 1 + ord(y_i) > ord(y_i)$. To daje sprzeczność. □

β-redukcja

Przykład 26.

$$\begin{aligned} w, z &: o \\ f &: (o, o, o) \end{aligned}$$

$$(\lambda x^o y^o.fxy)zw \to_\beta (\underline{\lambda y^o.fzy})w \to_\beta fzw$$

β_s-redukcja

Definicja 27. Redeks jest bezpieczny wtw. jest prawie bezpieczną aplikacją w formie:

$$(\lambda x_1^{A_1} x_2^{A_2} \ldots x_n^{A_n}.M)N_1N_2\ldots N_l$$

gdzie $l, n \geq 1$ i M jest prawie bezpieczną aplikacją.

Definicja 28.

$$\begin{aligned} \beta_s \quad &= \quad \{\ (\lambda x_1^{A_1} \ldots x_n^{A_n}.M)N_1 \ldots N_l \mapsto \lambda x_{l+1}^{A_{l+1}} \ldots x_n^{A_n}.M\left[\overline{N}/x_1 \ldots x_l\right] \mid n > l\} \\ &\cup \quad \{\ (\lambda x_1^{A_1} \ldots x_n^{A_n}.M)N_1 \ldots N_l \mapsto M\,[N_1 \ldots N_n/\overline{x}]\,N_{n+1} \ldots N_l \mid n \leq l\}\ . \end{aligned}$$

β_s-redukcja

Spostrzeżenie 29. Dla $l < n$, bezpieczny redeks ma natępujące drzewo wywodu:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dots}{\Gamma', \overline{x} : \overline{A} \vdash_{\mathsf{s}} M : (A_{n+1}, \ldots, A_l, B)}}{\Gamma' \vdash_{\mathsf{s}} \lambda x_1^{A_1} \ldots x_n^{A_n}.M : (A_1, \ldots, A_l, B)}\,(\mathsf{abs})}{\Gamma \vdash_{\mathsf{s}} \lambda x_1^{A_1} \ldots x_n^{A_n}.M : (A_1, \ldots, A_l, B)}\,(\mathsf{wk})}{\Gamma \vdash_{\mathsf{asa}} \lambda x_1^{A_1} \ldots x_n^{A_n}.M : (A_1, \ldots, A_l, B)}\,(\delta) \qquad \dfrac{\dots}{\Gamma \vdash_{\mathsf{s}} N_1 : A_1}}{\Gamma \vdash_{\mathsf{asa}} (\lambda x_1^{A_1} \ldots x_n^{A_n}.M)N_1 : (A_2, \ldots A_l, B)}\,(\mathsf{app}_{\mathsf{as}})$$

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash_{\mathsf{asa}} (\lambda x_1^{A_1} \ldots x_n^{A_n}.M)N_1 \ldots N_{l-1} : (A_l, B)}\,(\mathsf{app}_{\mathsf{as}}) \qquad \dfrac{\dots}{\Gamma \vdash_{\mathsf{s}} N_l : A_l}}{\Gamma \vdash_{\mathsf{s}} (\lambda x_1^{A_1} \ldots x_n^{A_n}.M)N_1 \ldots N_l : B}\,(\mathsf{app})$$

β_s-redukcja

Lemat 30 (Podstawianie zachowuje bezpieczeństwo)**.** *Niech* $\Gamma \vdash_{\mathsf{s}} N : A$*. Wtedy:*

1. $\Gamma, x : A \vdash_{\mathsf{s}} M : B \implies \Gamma \vdash_{\mathsf{s}} M\,[N/x] : B$
2. $\Gamma, x : A \vdash_{\mathsf{asa}} M : B \implies \Gamma \vdash_{\mathsf{asa}} M\,[N/x] : B$

Dowód Niech $\Gamma \vdash_{\mathsf{s}} N : A$. Udowodnimy obydwa punkty równolegle przez indukcję po drzwie wywodu $\Gamma, x : A \vdash_{\mathsf{s}} M : B$ lub $\Gamma, x : A \vdash_{\mathsf{asa}} M : B$.

Baza, czyli (var) i (const) działają trywialnie. (δ) praz (wk) również.

(abs) Mamy, że $\Gamma, x : A \vdash_{\mathsf{s}} \lambda\overline{y}^{\overline{C}}.Q \equiv M : (\overline{C}, D)$. Załóżmy, że x należy do $\overline{y}$. Zatem podstawienie nie jest wepchnięte do lambdy, więc wszystko działa trywialnie. W przeciwnym przypadku, załóżmy, że $\Gamma, x : A, \overline{y} : \overline{C} \vdash_{\mathsf{asa}} Q : D$. Aplikując założenie indukcyjne 2. na tym termie w kontekście, otrzymujemy, że $\Gamma, \overline{y} : \overline{C} \vdash_{\mathsf{asa}} Q\,[N/x] : D$, a dzięki reguły abs, dostajemy, że: $\Gamma \vdash_{\mathsf{s}} \lambda\overline{y}^{\overline{C}}.Q\,[N/x] : (\overline{C}, D)$.

($\mathsf{app_{as}}$) Mamy, że $M \equiv M_0 M_1 \dots M_p$, gdzie $p \geq 1$ oraz $\Gamma \vdash_{\mathsf{s}} M_k : A_k$ dla $1 \geq k \geq p$. Z założenia indukcyjnego mamy, że $\Gamma \vdash_{\mathsf{s}} M_k\,[N/x] : A_k$ dla każdego k. Reguły ($\mathsf{app_as}$) wystarczają, żeby zakończyć.

(app) Analogicznie.

Zatem podstawienie zachowuje typy.

□

β_s-redukcja

Lemat 31 (β_s-redukcja zachowuje bezpieczeństwo)**.** *Niech* $M_1 \beta_s M_2$*. Wtedy:*

- M_2 *jest prawie bezpieczne;*
- *jeśli* M_1 *jest bezpieczne, to* M_2 *również.*

β_s-redukcja

Definicja 32. Bezpieczna β-redukcja ($\rightarrow_{\beta_s}$) jest najmniejszą relacją, że jeśli $M_1 \beta_s M_2$ oraz $C[M]$ jest bezpiecznym termem w kontekście $C[-]$, to $C[M_1] \rightarrow_{\beta_s} C[M_2]$.

Lemat 33 (β_s-redukcja zachowuje bezpieczeństwo)**.** *Jeśli* $\Gamma \vdash_{\mathsf{s}} M_1 : A$ *oraz* $M_1 \rightarrow_{\beta_s} M_2$*, to* $\Gamma \vdash_{\mathsf{s}} M_2 : A$

Dowód Indukcja, jak w przypadku relacji β_s. □

η-długa forma normalna

Definicja 34. η-długa forma normalna termu $M : (A_1, \ldots, A_n, o)$, $\lceil M \rceil$ jest zdefiniowana następująco:

$$\begin{array}{rcl} \lceil x \rceil & \equiv & \lambda.x \\ \lceil \lambda x^\tau.N \rceil & \equiv & \lambda x^\tau.\lceil N \rceil \\ \lceil xN_1 \ldots N_m \rceil & \equiv & \lambda\overline{\varphi}^{\overline{A}}.x\lceil N_1 \rceil \ldots \lceil N_m \rceil \lceil \varphi_1 \rceil \ldots \lceil \varphi_n \rceil \\ \lceil (\lambda x^\tau.N)N_1 \ldots N_p \rceil & \equiv & \lambda\overline{\varphi}^{\overline{A}}.(\lambda x^\tau.\lceil N \rceil)\lceil N_1 \rceil \ldots \lceil N_m \rceil \lceil \varphi_1 \rceil \ldots \lceil \varphi_n \rceil \end{array}$$

gdzie $m \geq 0$, $p \geq 1$, $\overline{\varphi} = \varphi_1 \ldots \varphi_n$, gdzie wszystkie φ_i są nowymi zmiennymi.

Przykład 35.

$$\begin{array}{rcl} \lceil x^o \rceil & \equiv & \lambda.x \\ \lceil \lambda f^{(o,o,o)}.fx \rceil & \equiv & \lambda f^{(o,o,o)}\varphi_1^o.fx\varphi_1 \\ \lceil (\lambda f^{(o,o,o)}x^o y^o.fxy)sa \rceil & \equiv & \lambda\varphi_1^o(\lambda f^{(o,o,o)}x^o y^o.fxy)sa\varphi_1 \end{array}$$

η-długa forma normalna

Lemat 36. *Term jest bezpieczny wtw. jego η-długa forma normalna jest bezpieczna.*

Dowód Dowód przechodzi przez długie bezpieczeństwo, w które nie chce nam się zagłębiać. □

2.3 Siła ekspresji

Numerale Churcha

Definicja 37. Każda liczba $n \in \mathbb{N}$ może być zakodowana jako:

$$\overline{n}^I := \lambda s^{(o,o)}z^o.s^n z$$

gdzie $I = ((o,o),o,o)$

Definicja 38. p-arna funkcja $f : \mathbb{N}^p \to \mathbb{N}$ dla $p \geq 0$ jest reprezentowana przez term $F : (I^p, I)$ wtw. $\forall m_i \in \mathbb{N}$:

$$F\overline{m_1} \ldots \overline{m_p} =_\beta \overline{f(m_1, \ldots, m_p)}$$

Numerale Churcha

Twierdzenie 39 (Schwichtenberg, 1976)**.** *Funkcje numeryczne wyrażalne za pomocą numerali Churcha w prosto-typowalnym rachunku λ to dokładnie wielomiany wielu zmiennych rozszerzone o funkcję warunkową.*

Twierdzenie 40. *Funkcje numeryczne wyrażalne za pomocą numerali Churcha w bezpiecznym rachunku λ to dokładnie wielomiany wielu zmiennych.*

Dowód Dodawanie: $\overline{n+m} = \lambda\alpha^{(o,o)}x^o.(\overline{n}\alpha)(\overline{m}ax)$. Bezpieczne.

Mnożenie: $\overline{nm} = \lambda\alpha^{(o,o)}.\overline{n}(\overline{m}\alpha)$. Bezpieczne.

Złożenie: $G \equiv g : \mathbb{N}^n \to \mathbb{N}$, $F_1, \ldots, F_n \equiv f_1, \ldots, f_n : \mathbb{N}^p \to \mathbb{N}$. $(x_1, \ldots, x_p) \mapsto g(f_1(x_1, \ldots, x_p), \ldots, f_n(x_1, \ldots, x_p)) \equiv \lambda c_1 \ldots c_p.G(F_1c_1 \ldots c_p) \ldots (F_nc_1 \ldots c_p)$.

Zatem wszystkie wielomiany wielu zmiennych są reprezentowalne. Pytanie, co z ifem.

Załóżmy, że mamy term U w jego η-długiej formie normalnej.

Niech $\mathcal{N}^\tau_\Sigma$ oznacza zbiór bezpiecznych η-długich β-normalnych termów typu τ z wolnymi zmiennymi w Σ oraz $\mathcal{A}^\tau_\Sigma$ oznacza zbiór β-normalnych termów typu τ o wolnych zmiennych z Σ w formie $\varphi s_1 \ldots s_m$, dla jakiegoś $\varphi : (A_1, \ldots, A_m, o)$, gdzie $m \geq 0$ oraz $\forall_{1\leq i\leq m} s_i \in \mathcal{N}^{A_i}_\Sigma$.

Zauważmy, że zbiór $\mathcal{A}^o_\Sigma$ zawiera tylko termy bezpieczne, ale dla dowolnego τ, może być różnie. Niech Σ oznacza alfabet $\{x, y : I, z : o, \alpha : o \to o\}$. Za pomocą konstrukcji Zaionca, możemy otrzymać następujące równania indukujące gramatykę nad zbiorem terminali $\Sigma \cup \{\lambda xy\alpha z., \lambda z.\}$, generujące dokładnie termy $\mathcal{N}^{(I,I,I)}_\emptyset$:

$$\begin{array}{rcl}
\mathcal{N}^{(I,I,I)}_\emptyset & \to & \lambda xy\alpha z.\mathcal{A}^o_\Sigma \\
\mathcal{A}^o_\Sigma & \to & z \mid \mathcal{A}^{(o,o)}_\Sigma \mathcal{A}^o_\Sigma \\
\mathcal{A}^{(o,o)}_\Sigma & \to & \alpha \mid \mathcal{A}^I_\Sigma \mathcal{N}^{(o,o)}_\Sigma \\
\mathcal{N}^{(o,o)}_\Sigma & \to & \lambda z.\mathcal{A}^o_\Sigma \\
\mathcal{A}^I_\Sigma & \to & x \mid y
\end{array}$$

Kluczowa jest reguła czwarta. Gdybyśmy nie wymagali bezpieczeństwa, prawa strona by wyglądała następująco: $\lambda w^o.\mathcal{A}^{(o,o)}_{\Sigma\cup\{w:o\}}$. Bezpieczeństwo wymaga od nas wyabstrachowania wszystkich wolnych zmiennych typu o, jako że w termie może się pojawić tylko jedna taka zmienna.

Funkcję $f : \mathbb{N}^2 \to \mathbb{N}$ reprezentujemy jako:

- $\Sigma \vdash_{\mathsf{st}} F : o$ wtw. $\forall m, n \in \mathbb{N}, F\,[\overline{m}, \overline{n}/x, y] =_\beta \alpha^{\overline{f(m,n)}}z$;
- $\Sigma \vdash_{\mathsf{st}} G : (o, o)$ wtw. $G\,[\overline{m}, \overline{n}/x, y] =_\beta \lambda z.\alpha^{\overline{f(m,n)}}z$;
- $\Sigma \vdash_{\mathsf{st}} H : I$ wtw. $H\,[\overline{m}, \overline{n}/x, y] =_\beta \lambda\alpha z.\alpha^{\overline{f(m,n)}}z$;

Teraz przez indukcję po regułach gramatyki dostajemy, że dowolny term wygenerowany przez gramatykę reprezentuje jakiś wielomian.

Baza:

- Term x oraz y reprezentuje funkcję rzutowania, odpowiednio $(m,n) \mapsto m$ oraz $(m,n) \mapsto n$;
- Term α oraz z reprezentują funkcje stałe, odpowiednio $(m,n) \mapsto 1$ oraz $(m,n) \mapsto 0$.

Krok: Pierwsza i czwarta reguła są trywialne dla $F \in \mathcal{A}^o_\Sigma$, termy $\lambda z.F$ oraz $\lambda xy\alpha.F$ reprezentują te same funkcje, co F. Rozważamy teraz drugą i trzecią regułę. Widzimy, że dla $m, p, p' \geq 0$ dostajemy:

1. $\overline{m}(\lambda z.\alpha^p z) =_\beta \lambda z.\alpha^{m \cdot p} z$;
2. $(\lambda z.\alpha^p z)(\alpha^{p'} z) =_\beta \alpha^{p+p'} z$

Załóżmy, że $F \in \mathcal{A}^I_\Sigma$ oraz $G \in \mathcal{N}^{(o,o)}_\Sigma$ reprezentują odpowiednio funkcje f oraz g. Zatem FG z 1. reprezentuje funkcję $f \cdot g$. Jeśli $F \in \mathcal{A}^{(o,o)}_\Sigma$ oraz $G \in \mathcal{N}^o_\Sigma$ reprezentują f oraz g, to z 2. reprezentują funkcję $f + g$.

Stąd U reprezentuje jakiś wielomian, dla każdego $m, n \in \mathbb{N}$ mamy, że $U\overline{mn} =_\beta \lambda\alpha z.\alpha^{p(m,n)} z$, gdzie $p(m,n) = \Sigma_{0 \leq k \leq d} m^{i_k} n^{j_k}$ dla jakichś $i_k, j_k \geq 0$ oraz $d \geq 0$. □

Spostrzeżenie 41. Operator warunkowy $C : I \to I \to I \to I$, który by spełniał:

$$Ctyz \to_\beta \begin{cases} y & | \; t \to_\beta \overline{0} \\ z & | \; t \to_\beta \overline{n+1} \end{cases}$$

nie jest definiowalny w bezpiecznym rachunku λ.

Przykład 42. Term $\lambda FGH\alpha x.F(\underline{\lambda y.G\alpha x})(H\alpha x)$, zaproponowany przez Schwichtenberga, nie jest bezpieczny.

Spostrzeżenie 43. To, że nie da się stworzyć operatora warunkowego za pomocą numerali Chucha, nie znaczy, że nie jest to w ogóle możliwe. Można zawsze zareprezentować operator warunkowy za pomocą innego kodowania liczb naturalnych. Jedną z opcji jest poradzić sobie tworząc przeliczalnie wiele baz reprezentacji liczb. Taka technika jest użyta przy reprezentowaniu poprzednika w prosto-typowanym rachunku λ.

Spostrzeżenie 44. Można kodować binarny operator warunkowy następująco: kodujemy boole termami typu $B = (o,o,o)$. Dwie wartości prawdziwe są reprezentowane jako $\lambda x^o y^o.x$ oraz $\lambda x^o y^o.y$, a operator warunkowy jest dany termem $\lambda F^B G^B H^B x^o y^o.F(Gxy)(Hxy)$.

Spostrzeżenie 45. Możliwe jest również definiowanie operatora warunkowego, niczym operatora warunkowego C w rachunku lambda drugiego rzędu. Liczby naturalne reprezentowane są jako $\overline{n} \equiv \Lambda t.\lambda s^{t \to t} z^t.s^n(z)$ typu $J \equiv \Delta t.(t \to t) \to (t \to t)$, a operator warunkowy reprezentowany jako $\lambda F^J G^J H^J.FJ(\lambda u^J.G)H$. To, czy term jest bezpieczny, czy nie, nie może być stwierdzone w tym momencie, jako że nie pojęcie bezpieczeństwa termów o typach drugiego rzędu nie jest zbadane.

Funkcje na słowach

Zaionc badał problemy dla funkcji na słowach, później na drzewach, a następnie funkcjach na wolnych algebrach.

Definicja 46 (alfabet binarny)**.** $\Sigma = \{a, b\}$

Definicja 47 (numeral)**.** $\mathbf{k} := a \dots a \mid |\mathbf{k}| = k$

Definicja 48 (c)**.** $c(n, k) : (\Sigma^*)^n \to \Sigma^* := \mathbf{k}$

Definicja 49 (katenacja)**.** $app(x, y) : (\Sigma^*)^2 \to \Sigma^*$:= katenacja x oraz y.

Funkcje na słowach

Definicja 50 (podmiana)**.** $sub(x, y, z) : (\Sigma^*)^3 \to \Sigma^*$:= słowo powstałe przez podmianę w słowie x wszystkich wystąpień a przez y oraz wystąpień b przez z. Formalnie:

$$\begin{aligned} sub(\epsilon, y, z) &= \epsilon \\ sub(ax, y, z) &= app(y, sub(x, y, z)) \\ sub(bx, y, z) &= app(z, sub(x, y, z)) \end{aligned}$$

Przykład 51.

$$\begin{aligned} sub(abab, a, b) &= abab \\ sub(abab, b, a) &= baba \\ sub(abab, aba, a) &= abaaabaa \end{aligned}$$

Funkcje na słowach

Definicja 52 (ucięcie prefiksu)**.** $cut_a(x) : \Sigma^* \to \Sigma^*$:= maksymalny prefiks x zawierający same a. Formalnie:

$$\begin{aligned} cut_a(\epsilon) &= \epsilon \\ cut_a(ax) &= app(a, cut_a(x)) \\ cut_a(bx) &= \epsilon \end{aligned}$$

Przykład 53.

$$\begin{aligned} cut_a(\epsilon) &= \epsilon \\ cut_a(baaaaaa) &= \epsilon \\ cut_a(aaabaabbbab) &= aaa \end{aligned}$$

Funkcje na słowach

Definicja 54 (rzutowanie)**.** $\pi_k(x_1, \dots, x_k, \dots x_n) : (\Sigma^*)^n \to \Sigma^* = x_k$, gdzie $1 \leq k \leq n$

Definicja 55 (funkcja stała)**.** $cst_w(x) : \Sigma^* \to \Sigma^* = w$

Funkcje na słowach

Definicja 56 (niepustość)**.** $\overline{sq}(x) : \Sigma^* \to \Sigma^* = \left\{ \begin{array}{ll} \mathbf{0} & \mid x = \epsilon \\ \mathbf{1} & \mid x \neq \epsilon \end{array} \right. = cut_a(app(sub(x, b, b), a)$

Definicja 57 (pustość)**.** $sq(x) : \Sigma^* \to \Sigma^* = \overline{sq}(\overline{sq}(x))$

Definicja 58 (występowanie)**.** $occ_l(x) : \Sigma^* \to \Sigma^* = \left\{ \begin{array}{ll} \mathbf{1} & \mid l \in x \\ \mathbf{0} & \mid l \notin x \end{array} \right. = sq(sub(x, l, \epsilon))$

Funkcje na słowach

Definicja 59 (typ słowa binarnego)**.** $\mathbf{B} = (o \to o) \to (o \to o) \to o \to o$

Intuicja **B** jest następująca: **B** przyjmuje katenatory a i b oraz konstruktor ϵ i generuje sobie słowa.

Istnieje bijekcja pomiędzy domkniętymi termami typu **B** a słowami nad Σ.

Przykład 60.

$$\begin{array}{rcl} \underline{\epsilon} & = & \lambda u^{o\to o} v^{o\to o} x^o.x \\ \underline{a} & \equiv & \lambda u^{o\to o} v^{o\to o} x^o.ux \\ \underline{abaa} & \equiv & \lambda u^{o\to o} v^{o\to o} x^o.u(v(u(ux))) \end{array}$$

Funkcje na słowach

Twierdzenie 61 (Zaionc, 1987)**.** *Zbiór słów na słowach definiowalnych w rachunku λ to minimalny zbiór zawierający*

1. *funkcje stałe cst_w*
2. *rzutowania π_k*
3. *katenację app*
4. *ucięcie prefiksu cut_a*

domknięty na złożenia.

Funkcje na słowach

Przykład 62.

$$\begin{aligned}
\pi_k &\equiv \lambda x_1 \dots x_n.x_i \\
cst_w &\equiv \lambda x_1 \dots x_n.\underline{w} \\
app &\equiv \lambda cduvx.cuv(duvx) \\
sub &\equiv \lambda xdeuvx.c(\lambda y.duvy)(\lambda y.euvy)x \\
cut_a &\equiv \lambda cuvx.cu(\lambda y.x)x \\
cut_b &\equiv \lambda cuvx.c(\lambda y.x)vx \\
sq &\equiv \lambda cuvx.c(\lambda y.ux)(\lambda y.ux)x \\
\overline{sq} &\equiv \lambda cuvx.c(\lambda y.x)(\lambda y.x)(ux) \\
occ_a &\equiv \lambda cuvx.c(\lambda y.ux)(\lambda y.y)x \\
occ_b &\equiv \lambda cuvx.c(\lambda y.y)(\lambda y.ux)x
\end{aligned}$$

Można zwrócić uwagę, że tylko *app* oraz *sub* są bezpiecznie zdefiniowane. Okazuje się, że *app* i *sub* stanowią bazę do wygenerowania wszystkich definiowalnych funkcji w bezpiecznym rachunku λ.

Funkcje na słowach

Twierdzenie 63. *Funkcje na słowach definiowalne w bezpiecznym rachunku λ to mimimalny zadany przez:*

1. *funkcje stałe cst_w*
2. *rzutowania π_k*
3. *katenację app*
4. *podmianę sub*

domknięty na złożenia.

Dowód Dowód jest podobny do dowodu twierdzenia Zaionca. W pierwszą stronę jest prosto, bo wszystko pokazaliśmy w przykładzie. W drugą jest sporo roboty. □

2.4 Złożoność

Elementarna rekursja

Definicja 64 (Funkcje elementarnie rekursywne)**.**

$$\begin{aligned}
\text{ELEMENTARY} &= \text{EXPTIME} \cup 2\,\text{EXPTIME} \cup 3\,\text{EXPTIME} \cup \dots \\
&= \text{DTIME}(2^n) \cup \text{DTIME}(2^{2^n}) \cup \text{DTIME}(2^{2^{2^n}}) \cup \dots
\end{aligned}$$

Spostrzeżenie 65.

$$\text{EXPTIME} \subsetneq \text{ELEMENTARY} \subsetneq \text{PRec} \subsetneq \text{Rec}$$

Złożoność

Twierdzenie 66 (Statman, 1979)**.** *$\beta\eta$-równość dwóch typowalnych λ-termów pierwszego rzędu nie jest* ELEMENTARY

True Quantified Boolean Formula

Definicja 67. (True Quantified Boolean Formula) SAT z możliwością wstawiania kwantyfikatorów dla każdej zmiennej.

Twierdzenie 68. TQBF *jest* PSPACE-*complete.*

Twierdzenie 69. *$\beta\eta$-równość jest* PSPACE-*hard.*

Dowód Przez utworzenie konstrukcji, która sprowadza TQBF do $\beta\eta$-równości bezpiecznych termów.

$$\begin{aligned} \sigma_0 &\equiv o \\ \sigma_{n+1} &\equiv \sigma_n \to \sigma_n \end{aligned}$$

$$B_n \equiv \sigma_n \to o \to o \to o$$

$ord(B_n) = n + 1$

$$\underline{i}_{n+1} \equiv \lambda x^{\sigma_n}.x : \sigma_{n+1}$$

$\vdots$

□

Spostrzeżenie 70. PSPACE-hard jest jakimś dolnym oszacowaniem, może być więcej, ale nie wiadomo.

3 Co dalej?

3.1 Ciekawostki

Języki na słowach

Twierdzenie 71 (Damm, Goerdt)**.** *Języki generowane przez bezpieczne gramatyki rzędu n odpowiadają kolejnym klasom automatów z zagnieżdżonym stosem, tj. językom regularnym, bezkontekstowym, indeksowanym, . . .*

Przy czym mało wiadomo o wyższych klasach.

Drzewa

Twierdzenie 72 (Knapik z ekipą)**.** *Twierdzenia monadycznej logiki drugiego rzędu (MSO) na drzewach generowanych przez bezpieczne deterministyczne gramatyki są rozstrzygalne.*

Definicja 73. Monadyczna logika drugiego rzędu (MSO, monadic second-order logic): logika drugiego rzędu, gdzie predykaty są jednoargumentowe. Równoważnie, predykaty mogą być aplikowane do zbiorów.

Grafy

Twierdzenie 74 (Caucal)**.** *Twierdzenia MSO są rozstrzygalne dla grafów generowanych przez bezpieczne gramatyki dowolnego skończonego rzędu.*

Twierdzenie 75 (Hague)**.** *Twierdzenia MSO dla grafów generowanych przez niebezpieczne gramatyki jest nierozstrzygalne, a twierdzenia μ-rachunku są n-EXPTIME zupełne.*

Definicja 76. Grafy generowane przez gramatyki: grafy konfiguracji push-down automatu rzędu n.

3.2 Bibliografia

Bibliografia

Literatura

[1] William Blum, *The Safe Lambda Calculus*, praca doktorska, Oxford University Computing Laboratory, Michaelmas 2008

[2] William Blum and C.-H. Luke Ong, *The Safe Lambda Calculus*, Logical Methods in Computer Science, Vol. 5 (1:3) 2009, str. 1–38