11. Wykład 11: Rachunek λ. Obliczenia i obliczalność.

Rachunek λ jest systemem pozornie bardzo prostym. Abstrakcja i aplikacja wydają się trywialnymi operacjami, i może się zdawać, że niczego ciekawego nie da się z nich uzyskać. Okazuje się jednak, że siła wyrazu tego rachunku jest tak duża jak moc obliczeniowa maszyn Turinga.

Zaczniemy od dość paradoksalnej własności rachunku lambda. Każde równanie stałopunktowe ma w nim rozwiązanie. Odpowiedzialny za to jest na przykład taki term:

$$Y := \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)).$$

Główna własność termu Y jest taka: Dla dowolnego termu F zachodzi β-równość:

$$F(Y(F)) =_\beta Y(F). \tag{1}$$

Rzeczywiście:

$$Y(F) =_\beta (\lambda x.F(xx))(\lambda x.F(xx)) =_\beta F((\lambda x.F(xx))(\lambda x.F(xx)) =_\beta F(Y(F)).$$

Termy o własności (**??**) nazywamy **kombinatorami punktu stałego.**

Stwierdzenie 1. *Dla dowolnego termu M istnieje taki term N, że $N =_\beta M[x := N]$.*

Dowód. Wystarczy przyjąć $N = Y(\lambda x.M)$. □

Przykłady: (1) Istnieje taki term M, że $Mxy =_\beta MyxM$. Można przyjąć $M = Y(\lambda mxy.myxm)$. Wtedy $M =_\beta \lambda xy.MyxM$, skąd także $Mxy =_\beta MyxM$.

(2) Istnieje taki term M, że dla dowolnego N zachodzi $MN =_\beta MNN$, na przykład $M = Y(\lambda mx.mxx)$.

Teraz pokażemy jak w rachunku λ można zinterpretować pewne proste konstrukcje. Zaczniemy od wartości logicznych

$$\text{true} = \lambda xy.x \text{ oraz } \text{false} = \lambda xy.y$$

i instrukcji warunkowej

$$\text{if } P \text{ then } Q \text{ else } R = PQR.$$

Łatwo sprawdzić, że if true then Q else $R \twoheadrightarrow_\beta Q$ oraz if false then Q else $R \twoheadrightarrow_\beta R$.

Następna konstrukcja to para uporządkowana. Przyjmiemy

$$\langle M, N\rangle = \lambda x.xMN;$$

$$\pi_i = \lambda x_1 x_2.x_i \text{ dla } i = 1, 2.$$

Jak należy się spodziewać, mamy

$$\langle M1, M2\rangle \pi_i \twoheadrightarrow_\beta M_i.$$

Zauważmy jednak, że nie zachodzi równość

$$\langle M\pi_1, M\pi_2\rangle =_\beta M,$$

tj. nasza para uporządkowana nie jest surjektywna .

Dodanie do rachunku lambda surjektywnej pary uporządkowanej powoduje utratę własności Churcha-Rossera. Oznacza to w szczególności, że w (beztypowym) rachunku lambda nie można takiej operacji zdefiniować.

Liczby naturalne reprezentujemy w rachunku lambda jako tzw. **liczebniki Churcha:**

$$c_n = \lambda fx.f^n(x),$$

gdzie notacja $f^n(x)$ oznacza oczywiście term $f(f(\ldots(x)\ldots))$, w którym f występuje n razy. Zwykle zamiast c_n będziemy pisać n, co jest mniej precyzyjne ale wygodne. Więc na przykład:

$$0 = \lambda fx.x;$$

$$1 = \lambda fx.fx;$$

$$2 = \lambda fx.f(fx),$$

i tak dalej. Zauważmy, że $1 =_{\beta\eta} I$, ale $1 \neq_{\beta} I$.

Powiemy, że funkcja częściowa $f : \mathbb{N}^k \multimap \mathbb{N}$ (tzn. relacja f określona na iloczynie kartezjańskim $A \times B$, gdzie A i B są dowolnymi zbiorami, taka że każdy element z A jest w relacji z najwyżej jednym elementem z B:

$$\forall a \in A \forall b_1, b_2 \in B a f b_1 \wedge a f b_2 \Rightarrow b_1 = b_2)$$

jest **definiowalna** w beztypowym rachunku λ (λ-definiowalna) jeżeli istnieje term zamknięty F, spełniający dla dowolnych liczebników $n_1, \ldots, n_k \in \mathbb{N}$ następujące warunki:

- Jeżeli $f(n_1, \ldots, n_k) = m$, to $Fn_1 \ldots n_k =_{\beta} m$;
- Jeżeli $f(n_1, \ldots, n_k)$ jest nieokreślone, to $Fn_1 \ldots n_k$ nie ma postaci normalnej.

Mówimy oczywiście, że F **definiuje** lub **reprezentuje** funkcję f w rachunku λ. W przypadku, gdy dodatkowo dla wszystkich $n_1, \ldots, n_k \in \mathbb{N}$ spełniony jest warunek

- Jeżeli $f(n_1, \ldots, n_k)$ jest określone, to $Fn_1 \ldots n_k$ ma własność silnej normalizacji,

to powiemy, że F **dobrze definiuje** funkcję f , którą nazwiemy **dobrze definiowalną.** Kilka przykładów termów definiujących pewne funkcje podamy poniżej.

Przykłady:

- Następnik: succ $= \lambda nfx.f(nfx)$;
- Dodawanie: add $= \lambda mnfx.mf(nfx)$;
- Mnożenie: mult $= \lambda mnfx.m(nf)x$;
- Potęgowanie: exp $= \lambda mnfx.mnfx$;
- Test na zero: zero $= \lambda m.m(\lambda y.\text{false})\text{true}$;
- Funkcja $k-$argumentowa stale równa zeru: $Z_k = \lambda m_1 \ldots m_k.0$;
- Rzut $k-$argumentowy na $i-$tą współrzędną: $\Pi_i = \lambda m_1 \ldots m_k.m_i$.

Przypomnijmy, że **funkcje częściowo rekurencyjne** tworzą najmniejszą klasę funkcji częściowych nad $\mathbb{N}$ zawierającą **funkcje bazowe**:

- następnik, $\text{succ}(n) = n + 1$;
- rzuty, $\Pi^i_k(n_1, \ldots, n_k) = n_i$;
- funkcje stale równe zeru, $Z_k(n_1, \ldots, n_k) = 0$,

i zamkniętą ze względu na składanie, rekursję prostą i operację minimum. Na wszelki wypadek, przypomnijmy też znaczenie tych terminów.

(1) Funkcja $f : \mathbb{N}^k \multimap \mathbb{N}$ powstaje przez **składanie funkcji** $h : \mathbb{N}^\ell \multimap \mathbb{N}$ z funkcjami $g_1, \ldots, g_\ell : \mathbb{N}^k \multimap \mathbb{N}$, gdy
   - $\text{Dom}(f) = \{\vec{n} : \vec{n} \in \text{Dom}(g_i) \text{ dla wszystkich } i, \text{ oraz } (g_1(\vec{n}), \ldots, g_\ell(\vec{n})) \in \text{Dom}(h)\}$;
   - $f(\vec{n}) = h(g_1(\vec{n}), ..., g_\ell(\vec{n}))$, dla dowolnego $\vec{n} \in \text{Dom}(f)$.

(2) Funkcja $f : \mathbb{N}^{k+1} \multimap \mathbb{N}$ powstaje przez **rekursję prostą** z funkcji $h : \mathbb{N}^{k+2} \multimap \mathbb{N}$ i funkcji $g : \mathbb{N}^k \multimap \mathbb{N}$, jeżeli dla dowolnego $n \in \mathbb{N}$ i dowolnego $\vec{n} \in \mathbb{N}^k$ spełnione są równania
   - $f(0, \vec{n}) = g(\vec{n})$;
   - $f(n + 1, \vec{n}) = h(f(n, \vec{n}), n, \vec{n})$,

   przy czym równanie uważamy za spełnione, gdy obie strony są określone i równe, lub obie strony są nieokreślone. Wartość wyrażenia $h(f(n, \vec{n}), n, \vec{n})$ jest określona wtedy i tylko wtedy gdy $(n, \vec{n}) \in \text{Dom}(f)$ oraz $(f(n, \vec{n}), n, \vec{n}) \in \text{Dom}(h)$.

(3) Funkcja $f : \mathbb{N}^k \multimap \mathbb{N}$ powstaje z funkcji $h : \mathbb{N}^{k+1} \multimap \mathbb{N}$ przez zastosowanie **operacji minimum**, co zapisujemy $f(\vec{n}) = \mu y[h(\vec{n}, y) = 0]$, gdy dla dowolnego $\vec{n} \in \mathbb{N}^k$:
- Jeśli istnieje takie m, że $h(\vec{n}, m) = 0$ oraz wszystkie wartości $h(\vec{n}, i)$ dla $i < m$ są określone i różne od zera, to $f(\vec{n}) = m$.
- Jeśli takiego m nie ma, to $f(\vec{n})$ jest nieokreślone.

Klasa **funkcji pierwotnie rekurencyjnych** to najmniejsza klasa funkcji całkowitych (tzn. po prostu funkcji, ale żeby nie pozajączkowały nam się z funkcjami częściowymi używanymi w tym rozdziale, będziemy mówili o funkcjach całkowitych) nad N zawierająca funkcje bazowe i zamknięta ze względu na składanie i rekursję prostą.

Przykłady: Szczególne funkcje pierwotnie rekurencyjne to **funkcja pary**

$$p(m, n) = \frac{(m + n)(m + n + 1)}{2 + m},$$

i dwie funkcje **left** i **right** , takie że dla dowolnego n,

$$n = p(\text{left}(n), \text{right}(n)),$$

czyli funkcje odwrotne do funkcji pary.

Mamy następujące:

Twierdzenie 1 (o postaci normalnej Kleene'go)**.** *Każdą funkcję częściowo rekurencyjną można przedstawić w postaci*

$$f(\vec{n}) = \text{left}(\mu y[g(\vec{n}, y) = 0]),$$

gdzie g jest funkcją pierwotnie rekurencyjną.

Udowodnimy teraz, że każda funkcja częściowo rekurencyjna jest definiowalna w rachunku λ. Zaczynamy od najłatwiejszego.

Lemat 1. *Funkcje bazowe są lambda-definiowalne.*

Dowód. Wynika to wprost z jednego z dyskutowanych przykładów. □

Lemat 2. *Jeśli funkcja całkowita f powstaje przez składanie $\lambda-$definiowalnych funkcji całkowitych, to też jest $\lambda-$definiowalna.*

Dowód. Wynika to wprost z faktu, że mamy tu do czynienia z funkcjami! □

Lemat 3. *Jeśli funkcja całkowita f powstaje przez rekursję prostą z $\lambda-$definiowalnych funkcji całkowitych, to też jest $\lambda-$definiowalna.*

Dowód. Załóżmy, że f jest zdefiniowana równaniami

$$\begin{aligned} f(0, \vec{n}) &= g(\vec{n}); \\ f(n + 1, \vec{n}) &= h(f(n, \vec{n}), n, \vec{n}), \end{aligned}$$

i że funkcje g i h są definiowalne odpowiednio za pomocą termów G i H. Zdefiniujmy pomocnicze termy

$$\begin{aligned} \text{Step} &= \lambda p.\langle \text{succ}(p\pi_1), H(p\pi_2)(p\pi_1)x_1 \dots x_m \rangle; \\ \text{Init} &= \langle 0, Gx_1 \dots x_m \rangle. \end{aligned}$$

Funkcja f jest wtedy definiowalna termem

$$F = \lambda x x_1 ... x_m . x \text{StepInit} \pi_2.$$

Ta definicja wyraża następujący algorytm obliczania wartości funkcji f : generujemy ciąg par

$$(0, a_0), (1, a_1), \ldots, (n, a_n),$$

gdzie $a_0 = g(n_1, \ldots, n_m)$, każde a_{i+1} to $h(a_i, i, n_1, \ldots, n_m)$ oraz $a_n = f(n, n_1, \ldots, n_m)$. □

Wniosek 1. *Funkcje pierwotnie rekurencyjne są $\lambda-$definiowalne.*

Ponieważ mamy twierdzenie Kleene'go, więc pozosta je już (prawie) tylko pokazać $\lambda-$definiowalność funkcji określonych przez minimum. Można do tego wykorzystać kombinator punktu stałego $\mathbf{Y}$, rozwiązując odpowiednie równanie stałopunktowe. My to zrobimy trochę delikatniej...

Twierdzenie 2. *Wszystkie funkcje częściowo rekurencyjne są $\lambda-$definiowalne.*

Dowód. Niech $f(\vec{n}) = \text{left}(\mu y[g(\vec{n}, y) = 0])$, gdzie g jest funkcją pierwotnie rekurencyjną. Na mocy poprzedniego wniosku, zarówno g jak left są $\lambda-$definiowalne pewnymi termami G i L. Określimy pomocniczy term

$$W = \lambda y.if\text{zero}(\vec{G}xy)\text{then}\lambda w.Ly\text{else}\lambda w.w(\text{succ}y)w.$$

Funkcja f jest definiowana termem

$$F = \lambda\vec{x}.W0W.$$

Rzeczywiście, przypuśćmy najpierw, że $\mu y[g(\vec{n}, y) = 0] = n$, oraz $\text{left}(n) = r$. Wtedy mamy taki ciąg redukcji:

$$F\vec{n} \twoheadrightarrow_\beta \overline{W}0\overline{W} \twoheadrightarrow_\beta \overline{W}1\overline{W} \twoheadrightarrow_\beta \ldots \twoheadrightarrow_\beta \overline{W}n\overline{W} \twoheadrightarrow_\beta Ln \twoheadrightarrow_\beta r,$$

gdzie $\overline{W}$ oznacza wynik podstawienia $\vec{n}$ na $\overline{x}$ w termie W.

Jeśli minimum jest nieokreślone, to znaczy, że wszystkie wartości $g(\vec{n}, m)$ są określone i różne od zera, bo funkcja g jest całkowita. Wtedy mamy nieskończony ciąg redukcji

$$F\vec{n} \twoheadrightarrow_\beta \overline{W}0\overline{W} \twoheadrightarrow_\beta \overline{W}1\overline{W} \twoheadrightarrow_\beta \ldots \twoheadrightarrow_\beta \overline{W}n\overline{W} \twoheadrightarrow_\beta \ldots$$

Ten ciąg jest quasi-lewostronny, a zatem na mocy postaci normalnej nie ma. □

Co to wszystko ma wspólnego z maszynami Turinga...? Okazuje się, że "obliczeniowo" rachunek λ jest tak samo dobry. Przypomnijmy, że (deterministyczną, jednotaśmową) **maszynę Turinga** nad alfabetem $\mathcal{A}$ można zdefiniowaś jako algebrę $\mathcal{M} = \langle \Delta, Q, \delta, q_0, q_a, q_r \rangle$, gdzie:

- Δ jest skończonym alfabetem, zawierającym $\mathcal{A}$ oraz symbol $B \notin \mathcal{A}$ (blank);
- Q jest skończonym zbiorem stanów;
- $q_0 \in Q$ jest stanem początkowym;
- $q_a \in Q$ jest stanem akceptującym;
- $q_r \in Q$ jest stanem odrzucającym;
- $\delta : (Q \setminus \{q_a, q_r\}) \times \Delta \to \Delta \times Q \times \{-1, 0, +1\}$ jest funkcją przejścia.

Zakładając, że zbiory Δ i Q są rozłączne, można zdefiniować **konfigurację maszyny** jako trójkę postaci (q, i, w), gdzie $q \in Q$, $i \in \mathbb{N}$ oraz $w \in \Delta^*$, przy czym utożsamiamy konfiguracje (q, i, w) oraz (q, i, wB).

Interpretacja tej definicji jest następująca. Taśma maszyny jest nieskończona w prawo. Na początku taśmy zapisane jest słowo w, dalej w prawo są same blanki, a głowica maszyny zna jduje się w pozycji i. Konfigurację postaci $\mathcal{C}_w = (q_0, 0, w)$, gdzie $w \in A^*$, nazywamy **początkową**. Konfiguracje akceptujące (odp. odrzucające) są zaś postaci (q_a, i, w) (odp. (q_r, i, w)).

Jeśli $\mathcal{C} = (q, i, w)$ oraz $w = a_0 \ldots a_k$, to **następna** konfiguracja $\mathcal{C}'$ jest określona tak:

- Jeśli $\delta(q, a) = (b, p, +1)$ to $\mathcal{C}' = (p, i + 1, w[i \leftarrow b])$;
- Jeśli $\delta(q, a) = (b, p, 0)$ to $\mathcal{C}' = (p, i, w[i \leftarrow b])$;

- Jeśli $\delta(q,a) = (b,p,-1)$ to $\mathcal{C}' = (p, i-1, w[i \leftarrow b])$.

Piszemy $\mathcal{C} \rightarrow_{\mathcal{M}} \mathcal{C}'$, a symbolem $\twoheadrightarrow_{\mathcal{M}}$ oznaczamy przechodnio-zwrotne domknięcie relacji $\rightarrow_{\mathcal{M}}$. Jeżeli $\mathcal{C}_w \twoheadrightarrow_{\mathcal{M}} \mathcal{C}'$, gdzie $\mathcal{C}'$ jest konfiguracją akceptującą (odp. odrzucającą) to mówimy, że maszyna **akceptuje** (odp. **odrzuca**) słowo w. Maszyna **zatrzymuje się** dla wejścia w, wtedy i tylko wtedy, gdy słowo w jest akceptowane lub odrzucane.

Obliczenie rozpoczynające się od konfiguracji $\mathcal{C}_0$ to skończony lub nieskończony ciąg konfiguracji

$$C_0 \rightarrow_{\mathcal{M}} C_1 \rightarrow_{\mathcal{M}} C_2 \rightarrow_{\mathcal{M}} \dots$$

Obliczenie skończone jest **akceptujące** lub **odrzucające**, w zależności od ostatniego stanu. Oczywiście maszyna deterministyczna ma obliczenie nieskończone rozpoczynające się od $\mathcal{C}_w$ wtedy i tylko wtedy, gdy nie zatrzymuje się dla wejścia w.

Przez $L(\mathcal{M})$ oznaczamy język złożony ze wszystkich słów akceptowanych przez maszynę $\mathcal{M}$. Zbiór słów $L \subset \mathcal{A}^*$ jest **częściowo obliczalny** wtedy i tylko wtedy, gdy $L = L(\mathcal{M})$ dla pewnej maszyny $\mathcal{M}$. Jeśli dodatkowo maszyna odrzuca dokładnie te słowa, które nie należą do L, to mówimy, że L jest **obliczalny**.

Jeśli $\mathcal{A} = \{1\}$, a każdą liczbę naturalną n zinterpretujemy jako ciąg jedynek długości n, to język $L(\mathcal{A})$ można uważać za podzbiór $\mathbb{N}$. Możemy też użyć maszyny Turinga do definiowania funkcji $f_{\mathcal{M}} : \mathbb{N} \multimap \mathbb{N}$. Jeśli $\mathcal{M}$ zatrzymuje się dla wejścia n, to za wartość funkcji $f_{\mathcal{M}}$ przyjmujemy pozycję głowicy w odpowiedniej konfiguracji końcowej. W przeciwnym razie wartość $f_{\mathcal{M}}$ uważamy za nieokreśloną.

Twierdzenie 3. *Funkcja częściowa $f : \mathbb{N}^k \multimap \mathbb{N}$ jest częściowo obliczalna , wtedy i tylko wtedy, gdy jest postaci $f_{\mathcal{M}}$ dla pewnej maszyny $\mathcal{M}$. Każda funkcja (częściowo) rekurencyjna jest też (częściowo) obliczalna.*