Tłumaczenie λY-termów na automaty CPDA

Celem wykładu jest pokazanie, że dla każdego λY-termu można skonstruować automat ze stosem wyższego rzędu z paniką. W zapisaniu tego tłumaczenia pomoże wprowadzenie maszyny Krivine'a. Jest to maszyna umożliwiająca obliczanie drzewa generowanego przez λY-term.

Maszyna Krivine'a

Ramką nazwiemy parę (M, ρ), gdzie M jest λY-termem, a ρ *środowiskiem*, czyli funkcją częściową ze zmiennych w ramki. Typem ramki (M, ρ) nazwiemy typ termu M. *Konfiguracja* maszyny Krivine'a jest postaci (M, ρ, Σ), gdzie (M, ρ) jest ramką (nazywaną *główną* ramką konfiguracji), natomiast Σ to stos (czyli po prostu lista) ramek. Stos Σ może być pusty. Nie należy mylić stosu Σ w konfiguracji maszyny Krivine'a ze stosem automatu CPDA, który stworzymy za chwilę.

Niech M_0 będzie zamkniętym λY-termem typu o. Powiemy, że ramka (M, ρ) jest *poprawna dla* M_0, gdy

- M jest podtermem M_0 oraz
- dla każdej zmiennej $x \in \mathrm{FV}(M)$ ramka $\rho(x)$ jest zdefiniowana, ma ten sam typ co x i jest poprawna dla M_0.

Konfiguracja[1] $(M, \rho, R_k \ldots R_1)$ jest *poprawna dla* M_0, gdy

- ramki $(M, \rho), R_1, \ldots, R_k$ są poprawne dla M_0 oraz
- jeśli M ma typ $\alpha_k \to \ldots \to \alpha_1 \to o$, to R_i dla $1 \le i \le k$ ma typ α_i.

Zdefiniujmy przez indukcję term reprezentowany przez ramkę (poprawną). Ramka (M, ρ) reprezentuje term $M[N_1/x_1, \ldots, N_m/x_m]$, gdzie $x_1, \ldots, x_m$ to zmienne wolne termu M, natomiast N_i dla $1 \le i \le m$ to term reprezentowany przez $\rho(x_i)$. Konfiguracja $(M, \rho, R_k \ldots R_1)$ reprezentuje term $N\, N_k \ldots N_1$, gdzie N to term reprezentowany przez (M, ρ), natomiast N_i, dla $1 \le i \le k$, to term reprezentowany przez R_i. Powyższa definicja ma sens dla jedynie dla poprawnych konfiguracji i ramek: poprawność zapewnia, że typy w podstawieniu $N_1/x_1, \ldots, N_m/x_m$ i w aplikacji $N\, N_k \ldots N_1$ zgadzają się. Widzimy, że ramka reprezentuje zawsze term zamknięty, a konfiguracja – term zamknięty typu o.

Aby wyznaczyć drzewo generowane przez term M_0 za pomocą maszyny Krivine'a postępujemy w następujący sposób. Zaczynamy z konfiguracji $(M_0, \emptyset, \varepsilon)$ (gdzie $\emptyset$ to środowisko o pustej dziedzinie, a ε to pusty stos). Następnie, w zależności od postaci termu w głównej ramce konfiguracji wykonujemy jedną z następujących reguł:

$$\begin{aligned}
(\lambda x.M, \rho, R\,\Sigma) &\to (M, \rho[x \mapsto R], \Sigma)\,, \\
(M\,N, \rho, \Sigma) &\to (M, \rho, (N, \rho)\,\Sigma)\,, \\
(Y x.M, \rho, \Sigma) &\to (M, \rho[x \mapsto (Y x.M, \rho)], \Sigma)\,, \\
(x, \rho, \Sigma) &\to (M, \rho', \Sigma) \qquad \text{jeśli } \rho(x) = (M, \rho')\,.
\end{aligned}$$

Jeśli dojdziemy do konfiguracji postaci $(a\, M_1 \ldots M_r, \rho, \varepsilon)$, to tworzymy korzeń drzewa etykietowany symbolem a; aby uzyskać i-te poddrzewo korzenia dla $1 \le i \le r$, kontynuujemy obliczenie z konfiguracji (M_i, ρ, ε). Jeśli nie da się dojść do konfiguracji powyższej postaci, to przyjmujemy, że drzewo składa się z pojedynczego wierzchołka o etykiecie ω. Widzimy, że reguły maszyny Krivine'a są w pełni deterministyczne: każda konfiguracja ma jednego następnika (z wyjątkiem konfiguracji dla symbolu, która „wczytuje" literę a i ma $\mathit{rank}(a)$ następników).

Lemat. *Drzewo uzyskane w powyższy sposób jest drzewem generowanym przez* M_0.

Dowód. Zauważmy, że jeśli startujemy z konfiguracji $(M_0, \emptyset, \varepsilon)$, to wszystkie uzyskane po drodze konfiguracje będą poprawne dla M_0. Reguły dla zmiennej i aplikacji nie zmieniają reprezentowanego termu. Reguła dla λ-abstrakcji odpowiada wykonaniu czołowej β-redukcji na reprezentowanym termie, a reguła dla operatora Y odpowiada wykonaniu czołowej δ-redukcji na reprezentowanym termie.

Pozostaje zobaczyć, że z każdej konfiguracji po skończonej liczbie kroków dojdziemy do reguły wykonującej redukcję (czyli nie nastąpi nieskończone obliczenie nie zmieniające reprezentowanego termu). W tym celu zdefiniujmy rozmiar ramki (M, ρ) jako rozmiar termu M plus suma rozmiarów ramek $\rho(x)$ po wszystkich $x \in \mathrm{FV}(M)$. Łatwo przekonać się, że obie reguły nie wykonujące redukcji, czyli te dla aplikacji i dla zmiennej, zmniejszają tak zdefiniowany rozmiar głównej ramki w konfiguracji (nie wliczamy tutaj rozmiaru ramek na stosie, bo to popsułoby rozumowanie dla reguły dla aplikacji). □

[1] Ramki na stosie numerujemy od prawej, aby ramka znajdująca się na samym spodzie stosu miała numer 1; wygodniej jest, gdy stos rośnie w lewą stronę.

Reprezentacja konfiguracji maszyny Krivine'a przez stos CPDA

Niech M_0 będzie zamkniętym λY-termem typu o. Naszym celem jest stworzenie automatu CPDA generującego to samo drzewo co M_0. Na początku, dla ułatwienia, pokażemy jak uzyskać taki automat rzędu $\mathrm{comp}(M_0)+1$. Później zobaczymy jak poprawić podaną konstrukcję tak, aby uzyskany automat był rzędu o jeden mniejszego, czyli $\mathrm{comp}(M_0)$.

Niech $m_0 = \mathrm{comp}(M_0)+1$. Dla dolnego termu N przez $na(N)$ oznaczmy liczbę argumentów, na które czeka N, czyli taką liczbę, że typ N to $\alpha_{na(N)} \to \ldots \to \alpha_1 \to o$. Niech $k_{\max}$ będzie maksimum z $na(N)$ po wszystkich podtermach N termu M_0. Widzimy, że podczas obliczania drzewa generowanego przez M_0, na stosie konfiguracji maszyny Krivine'a nie będzie nigdy więcej niż $k_{\max}$ ramek.

Na stosie definiowanego automatu CPDA będziemy używać symboli

$$(x,i), (i,N), (x,N), col_l, \bot$$

gdzie N jest podtermem M_0, x jest zmienną występującą w M_0, l jest liczbą ze zbioru $\{2,\ldots,m_0\}$ oraz i jest liczbą ze zbioru $\{1,\ldots,k_{\max}\}$.

Intuicyjne znaczenie symboli stosowych jest następujące. Symbol (x,N) mówi, że wartością zmiennej x jest term N. Symbol (i,N) mówi, że termem w i-tej ramce stosu konfiguracji maszyny Krivine'a jest N. Symbol (x,i) mówi, że wartością zmiennej x jest term, który był w i-tej ramce stosu konfiguracji maszyny Krivine'a. Symbol col_l mówi, że szukając stosu konfiguracji maszyny Krivine'a powinniśmy wykonać operację $\mathsf{collapse}$ rzędu l. Symbol $\bot$ jest używany jedynie jako symbol początkowy, znajdujący się na spodzie stosu.

Oznaczmy $inv(l) = m_0 - l + 1$. Podamy teraz formalną definicję dwóch funkcji: *val* i *find*. Wartość $val(x,S)$ określa ramkę przypisaną do zmiennej x w środowisku reprezentowanym przez stos S, natomiast wartość $find(i,S)$ określa i-tą ramkę na stosie konfiguracji maszyny Krivine'a reprezentowanej przez stos S. Tutaj S jest stosem automatu CPDA, $top(S)$ oznacza najwyższy symbol na stosie S, natomiast $\mathsf{pop}_1(S)$ i $\mathsf{collapse}_l(S)$ oznaczają wynik wynikania operacji pop_1 bądź $\mathsf{collapse}_l$, odpowiednio, na stosie S.

$$val(x,S) = \begin{cases} (N, \mathsf{pop}_1(S)) & \text{jeśli } top(S) = (x,N), \\ find(i, \mathsf{pop}_1(S)) & \text{jeśli } top(S) = (x,i), \\ \text{nieokreślona} & \text{jeśli } top(S) = \bot, \\ val(x, \mathsf{pop}_1(S)) & \text{w przeciwnym przypadku.} \end{cases}$$

$$find(i,S) = \begin{cases} (N, \mathsf{pop}_1(S)) & \text{jeśli } top(S) = (i,N), \\ (N, S') & \text{jeśli } top(S) = col_l, (N,S') = find(i, \mathsf{collapse}_l(S)), \text{ oraz } l < inv(ord(N)), \\ \text{nieokreślona} & \text{jeśli } top(S) = col_l, (N,S') = find(i, \mathsf{collapse}_l(S)), \text{ oraz } l \geq inv(ord(N)), \\ \text{nieokreślona} & \text{jeśli } top(S) = \bot, \\ find(i, \mathsf{pop}_1(S)) & \text{w przeciwnym przypadku.} \end{cases}$$

Stos S reprezentuje środowisko $\rho[S]$ zdefiniowane przez indukcję jako

$$\rho[S](x) = (N, \rho[S']) \qquad \text{jeśli } val(x,S) = (N,S').$$

Ponadto stos S reprezentuje także i-tą ramkę na stosie konfiguracji maszyny Krivine'a $\gamma_i[S]$, zdefiniowaną jako

$$\gamma_i[S] = (N, \rho[S']) \qquad \text{jeśli } find(i,S) = (N,S').$$

Zanim opiszemy działanie automatu, poczyńmy dwa spostrzeżenia. Po pierwsze, ponieważ funkcje *val* i *find* używają w swojej definicji jedynie operacji pop_1 i $\mathsf{collapse}_l$, to łatwo przekonać się, że wykonanie operacji push_k nie ma wpływu na ich wynik.

Lemat. *Zachodzi* $\rho[S] = \rho[\mathsf{push}_k(S)]$ *oraz* $\gamma_i[S] = \gamma_i[\mathsf{push}_k(S)]$ *dla dowolnego* $k \in \{2,\ldots,m_0\}$.

Ponadto widzimy, że jeśli znaleziony term jest rzędu k, to podczas szukania go wykonywaliśmy jedynie operacje pop_1 i $\mathsf{collapse}_l$ dla $l < inv(k)$, które modyfikują wyłącznie najwyższy stos rzędu $inv(k)-1$.

Lemat. *Jeśli* $val(x, \mathsf{push}_{inv(k)}(S)) = (N,S')$ *i* $k = ord(N) > 0$ *to* $\mathsf{pop}_{inv(k)}(S') = S$.

Symulacja reguł maszyny Krivine'a przez operacje automatu CPDA

Automat CPDA $\mathcal{A}$ odpowiadający termowi M_0 będzie miał symbole stosowe jak opisano powyżej, natomiast jego stanami będą podtermy termu M_0 (oraz pewne stany pomocnicze, których nie wymieniamy jawnie). Konfiguracja (M,S) tego automatu reprezentuje konfigurację $(M, \rho[S], \gamma_{na(M)}[S] \ldots \gamma_1[S])$ maszyny Krivine'a. Widzimy, że konfiguracja początkowa $(M_0, S_\bot)$, gdzie $S_\bot$ to pusty stos (ściślej: stos zawierający jedynie symbol początkowy $\bot$), reprezentuje początkową konfigurację maszyny Krivine'a, czyli $(M_0, \emptyset, \varepsilon)$.

Podamy teraz reguły przejścia automatu $\mathcal{A}$ odpowiadające regułom maszyny Krivine'a. Będziemy to robić tak, aby prawdziwy był następujący lemat.

Lemat. *Jeśli konfiguracja (M,S) automatu $\mathcal{A}$ reprezentuje konfigurację C maszyny Krivine'a, z której można przejść do konfiguracji C', to z (M,S) automat $\mathcal{A}$ może przejść (jedynie) do pewnej konfiguracji reprezentującej C'. W szczególnym przypadku, gdy $M = a\,M_1\,\ldots\,M_r$, automat $\mathcal{A}$ może wczytać literę a i przejść do krotki r konfiguracji, z których i-ta reprezentuje i-ty następnik konfiguracji C.*

Zaczniemy od prostych przypadków; najtrudniejszy przypadek zmiennej zostawiamy sobie na koniec. Załóżmy, że stan automatu to M, a jego stos to S.

1. Jeśli $M = \lambda x.K$, to przechodzimy do stanu K i stosu $S' = \mathsf{push}_1^{(x,\gamma_{na(M)})}(S)$. Zobaczmy, że $\rho[S'](y) = \rho[S](y)$ dla $y \neq x$ oraz $\gamma_i[S'] = \gamma_i[S]$ dla $i \leq na(K) = na(M) - 1$, natomiast $\rho[S'](x) = \gamma_{na(M)}[S]$. Zatem (K,S') reprezentuje konfigurację $(K, \rho[S][x \mapsto \gamma_{na(M)}[S]], \gamma_{na(K)}[S]\ \ldots\ \gamma_1[S])$, osiąganą w wyniku wykonania jednego kroku przez maszynę Krivine'a.

2. Jeśli $M = K\,L$, to przechodzimy do stanu K i stosu $S' = \mathsf{push}_1^{(\gamma_{na(K)},L)}(S)$. Zobaczmy, że $\rho[S'] = \rho[S]$ oraz $\gamma_i[S'] = \gamma_i[S]$ dla $i \leq na(M) = na(K) - 1$, natomiast $\gamma_{na(K)}[S'] = (L, \rho[S])$. Zatem (K,S') reprezentuje konfigurację $(K, \rho[S], (L,\rho[S])\,\gamma_{na(K)}[S]\ \ldots\ \gamma_1[S])$, osiąganą w wyniku wykonania jednego kroku przez maszynę Krivine'a.

3. Jeśli $M = Yx.K$, to przechodzimy do stanu K i stosu $S' = \mathsf{push}_1^{(x,M)}(S)$. Zobaczmy, że $\rho[S'](y) = \rho[S](y)$ dla $y \neq x$ oraz $\gamma_i[S'] = \gamma_i[S]$ dla $i \leq na(M) = na(K)$, natomiast $\rho[S'](x) = (M, \rho[S])$. Zatem (K,S') reprezentuje konfigurację $(K, \rho[S][x \mapsto (M,\rho[S])], \gamma_{na(K)}[S]\ \ldots\ \gamma_1[S])$, osiąganą w wyniku wykonania jednego kroku przez maszynę Krivine'a.

4. Jeśli $M = a\,N_1\,\ldots\,N_r$, to wczytujemy literę a i w i-tym następniku, dla $1 \leq i \leq r$, przechodzimy do stanu N_i nie zmieniając stosu.

5. Jeśli $M = x$ oraz $ord(x) = 0$, to obliczamy $(N,S') = val(x,S)$ i przechodzimy do stanu N i stosu S'. Widać, że obliczanie $val(x,S)$ da się zaimplementować za pomocą automatu CPDA. Warunek $ord(x) = 0$ implikuje, że $na(x) = na(N) = 0$. Para (N,S') reprezentuje więc konfigurację $(N, \rho[S'], \varepsilon)$, osiąganą w wyniku wykonania jednego kroku przez maszynę Krivine'a.

6. Na koniec załóżmy, że $M = x$ oraz $ord(x) > 0$. Obliczamy wówczas $(N,S') = val(x, \mathsf{push}_{inv(ord(x))}(S))$, a następnie $S'' = \mathsf{push}_1^{col_{inv(ord(x))}}(S')$. Główna ramka reprezentowanej konfiguracji to $(N, \rho[S'']) = (N, \rho[S']) = \rho[\mathsf{push}_{inv(ord(x))}(S)](x) = \rho[S](x)$. Pozostaje dowieźć, że wartości ramek na stosie pozostają bez zmian, czyli że $\gamma_i[S''] = \gamma_i[S]$ dla każdego $i \in \{1,\ldots,na(x)\}$. Oznaczmy $(N_i,S_i) = find(i,S)$. Mamy $\mathsf{collapse}_{inv(ord(x))}(S'') = \mathsf{pop}_{inv(ord(x))}(S'') = S$ (pierwsza równość zachodzi, bo na szczycie S'' jest nowo wstawiony element, a druga dzięki obserwacji z poprzedniego podrozdziału). Na szczycie S'' leży $col_{inv(ord(x))}$, będziemy więc mieli $find(i,S'') = find(i, \mathsf{collapse}_{inv(ord(x))}(S'')) = find(i,S)$ (czyli $\gamma_i[S''] = \gamma_i[S]$) o ile tylko $inv(ord(x)) < inv(ord(N_i))$. Ale jeśli zmienna x (czyli term M) ma typ $\beta = \alpha_{na(x)} \to \ldots \to \alpha_1 \to o$, to term N_i ma typ α_i; ponieważ zaś $ord(\alpha_i) < ord(\beta)$, to $inv(ord(x)) < inv(ord(N_i))$.

Ponieważ stworzony automat $\mathcal{A}$ symuluje krok po kroku działanie maszyny Krivine'a na termie M_0, to wygeneruje on to samo drzewo co term M_0.

Zmniejszanie rzędu automatu

Powyżej, dla dowolnego λY-termu skonstruowaliśmy równoważny automat CPDA rzędu o jeden większego niż złożoność termu. Teraz pokażemy jak poprawić rząd o jeden, czyli jak uzyskać automat CPDA o rzędzie równym złożoności termu. Skorzystamy w tym celu z postaci kanonicznej termów.

Przypomnijmy z poprzedniego wykładu, że λY-term jest w postaci kanonicznej, jeśli wszystkie zmienne wolne dowolnego jego podtermu postaci $Yx.(\ldots)$ to zmienne z związane przez jakieś otaczające wyrażenie postaci $Yz.(\ldots)$ (a nie $\lambda z.(\ldots)$). Przypomnijmy również, że dla dowolnego λY-termu istnieje λY-term w postaci kanonicznej mający tę samą złożoność i generujący to samo drzewo.

Niech więc M_0 będzie zamkniętym λY-termem typu o w postaci kanonicznej. Załóżmy (bez straty ogólności), że każda zmienna związana w M_0 ma inną nazwę. W tej sytuacji możemy zmienić regułę maszyny Krivine'a dla zmiennych związanych przez Y na następującą:

$$(x,\rho,\Sigma) \to (Yx.P,\rho,\Sigma) \qquad \text{jeśli } Yx.P \text{ jest podtermem } M_0.$$

Oryginalnie reguła dla zmiennej zmieniała główną ramkę na $\rho(x)$. Wiemy jednak, że $\rho(x)$ zostało zdefiniowane dla otaczającego operatora wiążącego x, czyli dla podtermu $Yx.P$; zatem $\rho(x) = (Yx.P,\rho')$. W zasadzie powinniśmy także

zmienić środowisko w głównej ramce z ρ na ρ', ale ponieważ M_0 jest w postaci kanonicznej, to nic się nie stanie jeśli tego nie zrobimy; wiemy bowiem, że $Yx.P$ nie ma zmiennych wolnych wiązanych przez lambdę, a wartości środowiska dla zmiennych wolnych wiązanych przez Y przestały nas właśnie obchodzić. Ponadto teraz do środowiska nie musimy już w ogóle wstawiać wartości dla zmiennych wolnych wiązanych przez Y, czyli regułę dla Y możemy zmienić na następującą:

$$(Yx.P, \rho, \Sigma) \to (P, \rho, \Sigma)$$

Łatwo dostosować definicję automatu $\mathcal{A}$ do naszych nowych reguł. W obu przypadkach wystarczy zmiana stanu (z x na $Yx.P$ i z $Yx.P$ na P), bez jakiejkolwiek modyfikacji stosu.

Przypomnijmy, że zmienne wprowadzane przez lambdę mają rząd co najwyżej $m_0 - 2$ (bo $ord(x) < ord(\lambda x.P) \leq \mathrm{comp}(M_0) = m_0 - 1$). Zatem w efekcie naszej modyfikacji stara reguła dla zmiennej nie będzie nigdy wywoływana dla zmiennych rzędu $m_0 - 1$. W związku z tym, nie będziemy wykonywać operacji $\mathsf{push}_{inv(m_0-1)}$, czyli push_2 (a przy okazji także operacji $\mathsf{collapse}_2$). W efekcie każdy stos rzędu 2 będzie zawierał tylko jeden stos rzędu 1. Możemy więc zmniejszyć rząd naszego automatu do $m_0 - 1$, zamieniając wszystkie operacje push_k na push_{k-1} oraz $\mathsf{collapse}_k$ na $\mathsf{collapse}_{k-1}$, gdzie $k \geq 3$.

Rozmiar

Widzimy, że liczba stanów $\mathcal{A}$ oraz liczba jego symboli stosowych zależy w sposób wielomianowy od liczby podtermów oryginalnego λY-termu M_0, od liczby $k_{\max}$ oraz od liczby nazw zmiennych występujących w M_0. Nazw zmiennych w M_0 jest nie więcej niż różnych podtermów. Natomiast ograniczeniem na $k_{\max}$ są rozmiary typów podtermów M_0. Zatem rozmiar $\mathcal{A}$ jest wielomianowy względem rozmiaru grafowego termu M_0 (zdefiniowanego jako suma, po wszystkich różnych podtermach M_0, rozmiarów typów tych podtermów).

Przypomnijmy, że na wykładzie 3 pokazaliśmy jak zamienić schemat rekurencyjny $\mathcal{S}$ na λY-term M_0 (w postaci kanonicznej) generujący to samo drzewo. Składając to tłumaczenie z niniejszym, możemy dostać automat CPDA generujący to samo drzewo co $\mathcal{S}$ i będący tego samego rzędu. W dodatku rozmiar wynikowego automatu będzie wielomianowy względem rozmiaru $\mathcal{S}$.