P. Urzyczyn: Materiały do wykładu z semantyki[1]

Rachunek lambda CBN i CBV

Rachunek lambda często uważamy za abstrakcyjny język programowania funkcyjnego. Jednak ewaluacja wyrażenia w rzeczywistych językach funkcyjnych różni się istotnie od beta-redukcji. Po pierwsze, mamy wtedy określoną deterministyczną strategię redukcji. Po drugie, zwykle nie dokonuje się ewaluacji „pod lambdą": wyrażenie zdolne do pobrania argumentu (zgłaszające prompt) jest uważane za „gotową funkcję" i nie podlega ewaluacji aż do chwili dostarczenia parametru aktualnego. Dlatego lepszymi modelami języków funkcyjnych są takie wersje rachunku lambda, w których każdą abstrakcję uważamy za *wartość*. Jeden z takich modeli to *leniwy rachunek lambda*, nazywany też *rachunkiem CBN*. Opiszemy trzy rodzaje semantyki operacyjnej dla termów zamkniętych tego języka. Będą to:

- Semantyka tranzycyjna, określona za pomocą maszyny abstrakcyjnej. Modeluje ona proces obliczenia przez opisanie zmian stanu maszyny.
- Semantyka strukturalna (SOS[2]), w dwóch smakach:
  - Semantyka małych kroków, która definiuje pojedynczy krok ewaluacji „pośrednio", przez indukcję ze względu na strukturę ewaluowanego wyrażenia;
  - Semantyka dużych kroków, która definiuje od razu relację $M \Downarrow V$ pomiędzy wyrażeniem M i jego wartością V.

Małe kroki dla CBN

Relacja $\to$ w zbiorze termów to najmniejsza taka relacja, że $(\lambda x A)B \to A[B/x]$, oraz jeśli $A \to B$ to także $AC \to BC$. Zapisujemy to w skrócie w postaci takich reguł:

$$(\lambda x A)B \to A[B/x] \qquad\qquad \frac{A \to B}{AC \to BC}$$

Duże kroki dla CBN

Relacja $M \Downarrow V$, gdzie M jest dowolnym termem zamkniętym, a V jest *wartością*, czyli abstrakcją, jest określona regułami:

$$V \Downarrow V \qquad\qquad \frac{M \Downarrow \lambda x A \qquad A[N/x] \Downarrow V}{MN \Downarrow V}$$

Udowodnimy, że te dwie definicje są równoważne. Poniżej mowa o termach zamkniętych.

Lemat 0.1 *Jeśli $M \Downarrow V$, to $M \twoheadrightarrow V$.*

Dowód: Indukcja ze względu na definicję $M \Downarrow V$. W kroku indukcyjnym zakładamy, że $MN \Downarrow V$, bo $M \Downarrow \lambda x A$ oraz $A[N/x] \Downarrow V$. Te dwie asercje mają krótsze wyprowadzenia, więc z założenia indukcyjnego wnioskujemy, że $M \twoheadrightarrow \lambda x A$ oraz $A[N/x] \twoheadrightarrow V$. Ostatecznie otrzymujemy redukcję $MN \twoheadrightarrow (\lambda x A)N \to A[N/x] \twoheadrightarrow V$. ∎

[1] Pierwotna wersja tego materiału powstała we współpracy z M. Zielenkiewiczem.
[2] *Structural Operational Semantics*

Lemat 0.2 *Jeżeli $M \twoheadrightarrow M' \Downarrow V$, to $M \Downarrow V$.*

Dowód: Dowód jest przez indukcję ze względu na długość ciągu redukcji $M \twoheadrightarrow M'$. Oczywiście najważniejszy jest przypadek jednego kroku. Niech więc $M \to M'$. Użyjemy indukcji ze względu na definicję relacji $\to$.

Przypadek 1: $M = (\lambda x\, A)B \to A[B/x] = M' \Downarrow V$. Mamy wtedy takie wyprowadzenie:

$$\frac{\lambda x\, A \Downarrow \lambda x\, A \qquad A[B/x] \Downarrow V}{M \Downarrow V}$$

Przypadek 2: $M = AC \to BC \Downarrow V$, bo $A \to B$. Chcemy pokazać $AC \Downarrow V$. Skoro $BC \Downarrow V$, to $B \Downarrow \lambda x\, D$ oraz $D[C/x] \Downarrow V$. Z „wewnętrznego" założenia indukcyjnego dla $A \to B$, wnioskujemy, że $A \Downarrow \lambda x\, D$, a ponieważ $D[C/x] \Downarrow V$, więc $AC \Downarrow V$. ∎

Wniosek 0.3 *Dla dowolnych M i V, warunki $M \Downarrow V$ i $M \twoheadrightarrow V$ są równoważne.*

Uproszczona maszyna Krivine'a

Maszyna abstrakcyjna opisana poniżej różni się trochę od oryginalnej maszyny J.-L. Krivine'a. Głównie dlatego, że pomijamy tu całkowicie sprawę reprezentacji termów i otoczeń. Tymczasem „prawdziwa" maszyna Krivine'a posługuje się termami w notacji De Bruijna (indeksy zamiast zmiennych), obejmuje konkretną implementację otoczeń i algorytmy ich obsługi.

W definicji maszyny występują dwa pojęcia zdefiniowane przez wzajemną rekursję:

- *Otoczenie*, to funkcja częściowa, która zmiennym przypisuje domknięcia.
- *Domknięcie*, zwane też *klozurą* (ang. closure), to para $\langle M, E\rangle$, gdzie x jest zmienną, oraz E jest takim otoczeniem, że $\mathrm{FV}(M) \subseteq \mathrm{Dom}(E)$.

Ta definicja jest poprawna (dobrze ufundowana), bo otoczenie może być puste.

Konfiguracja maszyny Krivine'a to niepusta lista domknięć, która może być widziana jako trójka postaci $\langle M, E\rangle :: S$. Mamy tu term M do ewaluacji w otoczeniu E, i „stos" S. Konfiguracja początkowa ma postać $\langle M, \varnothing\rangle :: \mathit{nil}$, gdzie M jest termem zamkniętym, a konfiguracja końcowa powinna być postaci $\langle V, E\rangle :: \mathit{nil}$.

Ewaluacja aplikacji odbywa się metodą „push-enter", tj. polega na odłożeniu argumentu na stos (push) i wejściu w ewaluację operatora (enter). Maszyna ma takie instrukcje:

$$\begin{aligned} \langle x, E\rangle :: S &\Rightarrow E(x) :: S; \\ \langle M\,N, E\rangle :: S &\Rightarrow \langle M, E\rangle :: \langle N, E\rangle :: S; \\ \langle \lambda x\, M, E\rangle :: \Delta :: S &\Rightarrow \langle M, E[x \mapsto \Delta]\rangle :: S. \end{aligned}$$

Zdefiniujemy teraz *znaczenie domknięcia* (M, E), czyli *znaczenie termu* M w otoczeniu E.

$$\mathrm{Real}(M, E) = M[\mathrm{Real}(E(y))/y]_{y \in \mathrm{Dom}(E)}$$

Powyżej chodzi oczywiście o jednoczesne podstawienie na wszystkie zmienne $y \in \mathrm{Dom}(E)$. Definicja M^E ma sens także wtedy, gdy (M, E) nie jest domknięciem, tj. $E(y)$ nie jest określone dla jakiegoś $y \in \mathrm{FV}(M)$. Taka zmienna y pozostaje wtedy wolna w M^E.

Lemat 0.4 (Poprawność maszyny) *Jeśli $M^E \Downarrow V$, to $\langle M, E\rangle :: S \twoheadrightarrow \langle W, F\rangle :: S$, gdzie W jest wartością i $W^F = M^E$.*

Dowód: Indukcja ze względu na definicję relacji $\Downarrow$. Szczegóły pomijamy. ∎

W dowodzie następnego lematu użyjemy pomocniczej relacji $\Downarrow_k$, która jest zdefiniowana tak:

$$V \Downarrow_0 V \qquad\qquad \frac{M \Downarrow_k \lambda x\, A \qquad A[N/x] \Downarrow_\ell V}{MN \Downarrow_{k+\ell} V}$$

Jeśli $M \Downarrow_i V$ dla pewnego $i \leq k$, to napiszemy $M{\Downarrow_k}$. W przeciwnym razie $M{\not\Downarrow_k}$.

Lemat 0.5 *Jeśli $M^E \not\Downarrow_k$, to maszyna uruchomiona w stanie $\langle M, E\rangle :: nil$ wykonuje więcej niż k kroków.*

Dowód: Indukcja ze względu na k. Szczegóły pomijamy. ∎

Wniosek 0.6 *Jeśli M jest termem zamkniętym, to $M \Downarrow V$ wtedy i tylko wtedy, gdy maszyna Krivine'a ma obliczenie $\langle M, \varnothing\rangle \twoheadrightarrow \langle W, E\rangle$, gdzie $W^E = V$.*

Rachunek lambda w wersji CBV

W rachunku leniwym przekazywanie parametrów odbywa się „przez nazwę", tj. parametr aktualny jest przekazywany do procedury bez ewaluacji. W rzeczywistych językach często mamy do czynienia z przekazywaniem parametrów „przez wartość". Wtedy parametr aktualny musi sam być wartością. Taką strategię ewaluacji modeluje rachunek lambda w wersji CBV. Oto reguły semantyki dużych kroków dla takiego rachunku:

$$V \Downarrow V \qquad\qquad \frac{M \Downarrow \lambda x\, A \qquad N \Downarrow W \qquad A[W/x] \Downarrow V}{MN \Downarrow V}$$

Symbole V i W oznaczają tu wartości, czyli abstrakcje.

Definicja semantyki małych kroków dla CBV wymaga pojęcia *kontekstu ewaluacyjnego*. Kontekst ewaluacyjny $E[\,]$, z jedną „dziurą" oznaczaną przez $[\,]$, może być postaci:

$$[\,] \qquad\qquad V E[\,] \qquad\qquad E[\,] M,$$

Fakt 0.7 *Każdy term zamknięty M, który nie jest wartością, można w dokładnie jeden sposób przedstawić w postaci $M = E[R]$, gdzie E jest kontekstem ewaluacyjnym, a R jest beta-redeksem.*

Dowód: Ćwiczenie. ∎

Dzięki jednoznaczności rozkładu, poniższe reguły ewaluacji „przez wartość" określają deterministyczną strategię redukcji:

$$(\lambda x\, A)V \to A[V/x] \qquad\qquad \frac{A \to B}{E[A] \to E[B]}$$

Fakt 0.8 *Warunki $M \Downarrow V$ i $M \twoheadrightarrow V$ są równoważne.*

Dowód: Podobny do dowodu Wniosku 0.3. ∎

Maszyna SECD

Maszyna SECD została opisana przez P.J. Landina w roku 1965 i jest jedną z najstarszych maszyn abstrakcyjnych. Stan maszyny opisywany jest przez czwórkę $\langle S, E, C, D\rangle$, a nazwy składowych *Stack*, *Environment*, *Code* i *Dump*, tłumaczą jednocześnie znaczenie skrótu. Mamy więc:

- Stack – czyli listę domknięć postaci $\langle V, E\rangle$;
- Environment – czyli otoczenie;
- Code – czyli listę „operatorów" (operatorami są termy i symbol @);
- Dump – czyli listę trójek postaci $\langle S, E, C\rangle$.

Idea jest taka: Wyrażenie przeznaczone do ewaluacji w otoczeniu E, umieszczone początkowo w C, jest rozkładane na części (od prawej). Każda część w postaci domknięcia jest umieszczana na stosie. Kiedy na początku listy C pojawi się symbol @, maszyna ewaluuje operator z początku stosu, używając następnego domknięcia ze stosu jako argumentu. Odbywa się to przez rekurencyjne wywołanie całego procesu (po to jest składowa D). Ta metoda ewaluacji nazywana jest „eval-apply". Instrukcje maszyny są takie:

$$\begin{aligned}
\langle S, E, x :: C, D\rangle &\Rightarrow \langle E(x) :: S, E, C, D\rangle \\
\langle S, E, \lambda x M :: C, D\rangle &\Rightarrow \langle\langle \lambda x M, E\rangle :: S, E, C, D\rangle \\
\langle S, E, MN :: C, D\rangle &\Rightarrow \langle S, E, N :: M :: @ :: C, D\rangle \\
\langle\langle \lambda x M, E\rangle :: \Delta :: S, E', @ :: C, D\rangle &\Rightarrow \langle nil, E[x \mapsto \Delta], M :: nil, \langle S, E', C\rangle :: D\rangle \\
\langle \Delta :: S, E', nil, \langle S', E', C'\rangle :: D\rangle &\Rightarrow \langle \Delta :: S', E', C', D\rangle
\end{aligned}$$

Konfiguracja początkowa ma postać $\langle nil, \varnothing, M :: nil, nil\rangle$, a końcowa $\langle \Delta :: nil, \varnothing, nil, nil\rangle$. Mamy teraz dwa lematy podobne do lematów 0.4 i 0.5.

Lemat 0.9 *Jeśli $M^E \Downarrow V$, to $\langle S, E, M :: C, D\rangle \twoheadrightarrow \langle\langle W, F\rangle :: S, E, C, D\rangle$, gdzie $W^F = V$.*

Lemat 0.10 *Jeśli $M \not\Downarrow_k$ to maszyna* SECD*, uruchomiona w konfiguracji $\langle S, E, M :: C, D\rangle$, wykonuje co najmniej k kroków.*

Wniosek 0.11 *Jeśli M jest termem zamkniętym, to $M \Downarrow V$ wtedy i tylko wtedy, gdy maszyna* SECD *ma obliczenie $\langle nil, \varnothing, M :: nil, nil\rangle \twoheadrightarrow \langle\langle W, F\rangle, \varnothing, nil, nil\rangle$.*

Semantyka denotacyjna dla CBN i CBV

Modelami standardowej teorii rachunku lambda są zupełne porządki częściowe spełniające „równanie" postaci $D \cong [D \to D]$. W przypadku rachunku CBN jest trochę inaczej, bo

nie każdy element dziedziny powinien być interpretowany jako funkcja. Na przykład trzeba rozróżniać termy Ω i $\lambda x\,\Omega$. Pierwszy chcemy interpretować jako element najmniejszy $\bot$, a drugi jako funkcję stałą $\lambda\!\!\lambda d.\,\bot$. Interesuje więc nas rozwiązanie innego równania, a mianowicie $D \cong [D \to D]_\bot$, gdzie $A_\bot$ oznacza zbiór A z dodatkowym elementem najmniejszym $\bot$. (Dotychczasowy element najmniejszy w A przestaje pełnić tę funkcję.) Rozwiązanie takiego równania można uzyskać powtarzając konstrukcję modelu $\mathcal{D}_\infty$ z odpowiednimi modyfikacjami (zaczynamy od jednoelementowego zbioru D_0 i definiujemy D_{n+1} jako $[D_n \to D_n]_\bot$). Tak otrzymany model nie jest jednak w pełni abstrakcyjny ze względu na obserwacyjną równoważność odpowiednią dla CBV (taką gdzie zamiast o istnienie czołowej postaci normalnej pytamy o redukcję do wartości.) Modele w pełni abstrakcyjne można uzyskać w grach.

W przypadku rachunku CBV, oprócz rozróżnienia Ω i $\lambda x\,\Omega$ trzeba jeszcze uwzględnić konieczność ewaluacji argumentu. Nie interesują nas więc dowolne funkcje ciągłe, ale tylko „rzetelne", tj. funkcje o własności $f(\bot) = \bot$. Jeśli $[D \to_\bot D]$ oznacza przestrzeń funkcji rzetelnych, to równanie dziedzinowe przyjmuje postać $D \cong [D \to_\bot D]_\bot$. Metodą Scotta otrzymamy dobry model, który znowu nie będzie w pełni abstrakcyjny.

Ćwiczenia:

1. Zdefiniować maszynę SECD w wersji CBV.
2. Czy wnioski 0.6 i 0.11 można wzmocnić żądając aby $W = V$ i $F = \varnothing$?
3. Zdefiniować projekcję z $[D \to D]_\bot$ na D oraz metodę „podnoszenia" projekcji (i, j) z D' na D do projekcji z $[D' \to D']_\bot$ na $[D \to D]_\bot$.