Rachunek Lambda i Języki Programowania

Małgorzata Biernacka

Instytut Informatyki UWr

Wykład 4

Plan

Plan

Plan

Rozszerzenia rachunku lambda: stałe

- Składnia (następnik jako stała):

$$t := \ldots \mid n \mid \mathtt{succ}$$

- Pojęcie redukcji:

$$\mathtt{succ}\ n \longrightarrow n + 1$$

- Wartości:

$$v := \ldots \mid n \mid \mathtt{succ}$$

- Strategia – w CBN dodajemy regułę aplikatywną dla aplikacji nastepnika (w CBV – bez zmian)

$$\frac{r \longrightarrow_n s}{\mathtt{succ}\ r \longrightarrow_n \mathtt{succ}\ s}$$

- Za pomocą kontekstów – dodajemy nowy kontekst:

$$C := \ldots \mid C[\mathtt{succ}\ [\,]]$$

Rozszerzenia rachunku lambda: konstrukcja warunkowa

- Składnia (`if` jako forma specjalna):

$$t := \ldots \mid \texttt{\#t} \mid \texttt{\#f} \mid \texttt{if}\ t\ t\ t$$

- Pojęcie redukcji:

$$\texttt{if}\ \texttt{\#t}\ r\ s \longrightarrow r$$

$$\texttt{if}\ \texttt{\#f}\ r\ s \longrightarrow s$$

- Wartości:

$$v := \ldots \mid \texttt{\#f} \mid \texttt{\#t}$$

- Strategia – dodajemy następującą regułę (w CBN i CBV):

$$\frac{r \longrightarrow s}{\texttt{if}\ r\ t\ u \longrightarrow \texttt{if}\ s\ t\ u}$$

- Za pomocą kontekstów – dodajemy nowy kontekst:

$$C := \ldots \mid C[\texttt{if}\ [\,]\ \texttt{t}\ \texttt{u}]$$

Rozszerzenia rachunku lambda: konstrukcja `let`

- Składnia (`let` jako forma specjalna):

$$t := \ldots \mid \texttt{let}\, x = t\, \texttt{in}\, t$$

- Pojęcie redukcji:

$$\texttt{let}\, x = v\, \texttt{in}\, r \to r\,[v/x]$$

- Strategia – dodajemy następującą regułę (w CBN i CBV):

$$\frac{r \to s}{\texttt{let}\, x = r\, \texttt{in}\, u \to \texttt{let}\, x = s\, \texttt{in}\, u}$$

- Za pomocą kontekstów – dodajemy nowe konstruktory kontekstu:

$$C := \ldots \mid C[\texttt{let}\, x = [\,]\, \texttt{in}\, t]$$

Rozszerzenia rachunku lambda: definicje funkcji rekurencyjnych

- Składnia:

$$t := \ldots \mid \lambda^f x.t$$

- Wartości: $v := \ldots \mid \lambda^f x.t$
- Pojęcie redukcji (uogólniona β-redukcja):

$$(\lambda^f x.t)\, r \to t\ [r/x]\ [\lambda^f x.t/f]$$

Plan

Rachunek domknięć

- Modelowanie ewaluacji w maszynach ze środowiskiem na poziomie rachunku
- Reguły redukcji odzwierciedlają "opóźnione podstawienie"
- Rachunek pośredni między rachunkiem lambda a implementacjami używającymi środowiska i domknięć
- Istnieją różne wersje rachunku, w szczególności dla silnej normalizacji i dla ewaluacji

Rachunek domknięć dla ewaluacji ($\lambda\rho$)

- Składnia:

$$\begin{array}{ll} \text{(Term)} & t ::= i \mid \lambda t \mid t\ t \\ \text{(Domknięcie)} & c ::= t[s] \\ \text{(Podstawienie)} & s ::= \bullet \mid c \cdot s \end{array}$$

- Termy – jak w rachunku lambda z indeksami de Bruijna
- Domknięcia – syntaktyczna kategoria oznaczająca term z podstawieniem do wykonania
- Podstawienie (środowisko) – lista domknięć
- Pozycja na liście oznacza indeks zmiennej, pod którą podstawiamy

Rachunek domknięć dla ewaluacji ($\lambda\rho$), c.d.

- Semantyka CBN:
  - Wartości:

$$v ::= (\lambda t)[s]$$

  - System wnioskowania dla ewaluacji (semantyka dużych kroków):

$$(\mathrm{Eval}) \qquad \frac{t_1[s] \xrightarrow{\rho}{}^{*} (\lambda t_1')[s']}{(t_1\ t_2)[s] \xrightarrow{\rho} t_1'[(t_2[s]) \cdot s']}$$

$$(\mathrm{Var}) \qquad i[c_1 \cdots c_m] \xrightarrow{\rho} c_i \ \text{ if } i \leq m$$

  - Redukcje są wykonywane tylko na domknięciach
  - Słaba redukcja – nie ma obsługi termów wewnątrz lambda abstrakcji
  - Semantyka małych kroków nie jest wyrażalna w tym języku

Semantyka redukcyjna dla rachunku domknięć ($\lambda\widehat{\rho}$)

- Rozszerzamy składnię o konstrukcję pozwalającą wyrażać pośrednie wyniki obliczeń
- Składnia języka $\lambda\widehat{\rho}$:

(Term)	$t ::= i \mid t\ t \mid \lambda t$
(Domknięcie)	$c ::= t[s] \mid c\ c$
(Podstawienie)	$s ::= \bullet \mid c \cdot s$

- Aplikacja domknięć – modeluje pośredni wynik obliczeń

Semantyka redukcyjna dla rachunku domknięć, c.d.

- Wartości:

$$v ::= (\lambda t)[s]$$

- Pojęcie redukcji:

$$(\text{Beta}) \qquad ((\lambda t)[s])\ c \xrightarrow{\widehat{\rho}} t[c \cdot s]$$

$$(\text{App}) \qquad (t_1\ t_2)[s] \xrightarrow{\widehat{\rho}} (t_1[s])\ (t_2[s])$$

$$(\text{Var}) \qquad i[c_1 \cdots c_m] \xrightarrow{\widehat{\rho}} c_i \text{ if } i \leq m$$

- Konteksty:

$$E ::= [\,] \mid E[[\,]\,c]$$

Symulacja ewaluacji w obu semantykach

Theorem

Niech c_1 i c_2 będą domknięciami z języka $\lambda\rho$. Wówczas

- *Jeśli $c_1 \xrightarrow{\rho} c_2$, to $c_1 \xrightarrow{\widehat{\rho}}{}^* c_2$.*
- *Jeśli $c_1 \xrightarrow{\widehat{\rho}}{}^* c_2$, to $c_1 \xrightarrow{\rho}{}^* c_2$.*

Symulacja reguły (Eval)

$$(t_1\ t_2)[s] \xrightarrow{\widehat{\rho}} (t_1[s])\ (t_2[s]) \xrightarrow{\widehat{\rho}}{}^* ((\lambda t_1')[s'])\ (t_2[s]) \xrightarrow{\widehat{\rho}} t_1'[(t_2[s]) \cdot s']$$

Związek λρ i maszyn abstrakcyjnych

- Startując z rachunku domknięć z odpowiednią strategią, stosując konkretyzację semantyki redukcyjnej ("refocusing") otrzymujemy maszynę abstrakcyjną symulującą strategię ewaluacji rachunku źródłowego
- Konfiguracje otrzymanej maszyny składają się z: kodu (term), środowiska (podstawienie) i stosu (kontekst)
- Poprawność maszyny wynika z poprawności kolejnych kroków metody wyprowadzenia

Plan

Nielokalne reguły redukcji

- Standardowa semantyka redukcyjna:
  - Lokalne pojęcie redukcji (kontrakcje)

$$r \to r'$$

  - redukcja jednokrokowa

$$E[r] \to_1 E[r']$$

- Semantyka redukcyjna nielokalna (wrażliwa na kontekst):
  - Nielokalne pojęcie redukcji

$$E[r] \to r' \neq E[r'']$$

  - Redukcja jednokrokowa nie jest kompatybilnym domknięciem pojęcia redukcji
  - Cały term wpływa na postać reguły redukcji

Przykład rozszerzeń nielokalnych

- Operatory kontroli:

$$\text{Abort:} \quad E[\mathcal{A}\ t] \longrightarrow t$$

$$\text{Call/cc:} \quad E[\mathcal{C}\lambda k.t] \longrightarrow E[t\ [E^*/k]]$$

Leniwy rachunek lambda

- Leniwa ewaluacja wymaga wprowadzenia do języka reprezentacji "pamięci"
- Maszyna abstrakcyjna modelująca leniwą ewaluację:

$$\langle i,\, s,\, C,\, \sigma\rangle \Rightarrow_{\mathrm{l}} \langle C,\, (\lambda t',\, s'),\, \sigma\rangle \qquad \text{gdy } \sigma(s(i)) = (\lambda t',\, s')$$

$$\langle i,\, s,\, C,\, \sigma\rangle \Rightarrow_{\mathrm{l}} \langle t',\, s',\, C[\mathrm{upd}(\ell, [\,])],\, \sigma\rangle \qquad \text{gdy } \sigma(s(i)) = (t',\, s')$$

$$\langle \lambda t,\, s,\, C,\, \sigma\rangle \Rightarrow_{\mathrm{l}} \langle C,\, (\lambda t,\, s),\, \sigma\rangle$$

$$\langle t_1\, t_2,\, s,\, C,\, \sigma\rangle \Rightarrow_{\mathrm{l}} \langle t_1,\, s,\, C[[\,]\,\ell],\, \sigma[\ell \mapsto (t_2,\, s)]\rangle \qquad \ell \text{ świeże}$$

$$\langle [\,],\, v,\, \sigma\rangle \Rightarrow_{\mathrm{l}} (v,\, \sigma)$$

$$\langle C[[\,]\,\ell],\, (\lambda t,\, s),\, \sigma\rangle \Rightarrow_{\mathrm{l}} \langle t,\, \ell \cdot s,\, C,\, \sigma\rangle$$

$$\langle C[\mathrm{upd}(\ell, [\,])],\, v,\, \sigma\rangle \Rightarrow_{\mathrm{l}} \langle C,\, v,\, \sigma[\ell \mapsto v]\rangle$$

Leniwy rachunek lambda, c.d.

- Reguły redukcji:

$$
\begin{array}{lll}
(\mathrm{Var}_1) & \langle i[\ell_1 \cdots \ell_j],\ C[\sigma]\rangle \to_{\mathrm{l}} \langle v,\ C[\sigma]\rangle & \text{gdy } \sigma(\ell_i) = v \\
(\mathrm{Var}_2) & \langle i[\ell_1 \cdots \ell_j],\ C[\sigma]\rangle \to_{\mathrm{l}} \langle \mathtt{upd}(\ell_i, c),\ C[\sigma]\rangle & \text{gdy } \sigma(\ell_i) = c \\
(\mathrm{Beta}) & \langle ((\lambda t)[s])\,\ell,\ C[\sigma]\rangle \to_{\mathrm{l}} \langle t[\ell \cdot s],\ C[\sigma]\rangle & \\
(\mathrm{App}) & \langle (t_1\ t_2)[s],\ C[\sigma]\rangle \to_{\mathrm{l}} \langle (t_1[s])\,\ell,\ C[\sigma[\ell \mapsto t_2[s]]]\rangle & \text{gdzie } \ell \text{ “świeże”} \\
(\mathrm{Upd}) & \langle \mathtt{upd}(\ell, v),\ C[\sigma]\rangle \to_{\mathrm{l}} \langle v,\ C[\sigma[\ell \mapsto v]]\rangle &
\end{array}
$$

Inne rozszerzenia

- Nielokalne redukcje i rozszerzenia składni stosuje się do modelowania efektów takich, jak:
  - przypisanie
  - operatory kontroli
  - wejście/wyjście
  - wyjątki

Plan

Rachunek lambda

- Podręczniki
  1. H. Barendregt “The Lambda Calculus: Its Syntax and Semantics” (1984)
  2. Ch. Hankin “Lambda Calculi: A guide for computer scientists” (1994)
  3. P. Urzyczyn “Beztypowy rachunek lambda” (http://www.mimuw.edu.pl/~urzy/Lambda/beztypow.pdf)
- Artykuły
  1. H. Barendregt “The impact of the lambda calculus in logic and computer science” (1997)
  2. A. Church “The calculi of lambda conversion” (1941)
  3. T. Æ. Mogensen “Efficient self-interpretation in lambda calculus” (1992)

Języki programowania

- Podręczniki
  1. M. Felleisen, M. Flatt “Programming languages and lambda calculi”
  2. G. Winskel “The formal semantics of programming languages”
  3. J. C. Reynolds “Theories of programming languages”
  4. H. Abelson, G. J. Sussman, J. Sussman “Structure and interpretation of computer programs”
- Artykuły
  1. P. J. Landin “The mechanical evaluation of expressions” (1964)
  2. J. C. Reynolds “Definitional interpreters for higher-order programming languages” (1972)
  3. G. D. Plotkin “Call-by-name, call-by-value and the lambda calculus.” (1975)
  4. P.-L. Curien “An abstract framework for environment machines” (1990)
  5. D. A. Turner “A new implementation technique for applicative languages” (1979)
- W internecie: blog o językach programowania:
  `http://lambda-the-ultimate.org/`