Termy λ-rachunku według de Bruijna

Antoni Kościelski

Spis treści

1 Wyrażenia de Bruijna

1.1 Definicja wyrażeń de Bruijna

Znaczenie programistyczne ma sposób przedstawiania wyrażeń lambda rachunku wymyślony przez Nicolasa de Bruijna. Może zostać opisany poprzez następującą gramatykę.

1) ⟨w. de Bruijna⟩ ::= ⟨zmienna⟩ | ⟨aplikacja⟩ | ⟨abstrakcja⟩

2) ⟨wyrażenie proste⟩ ::= ⟨zmienna⟩ | (⟨aplikacja⟩) | (⟨abstrakcja⟩)

3) ⟨aplikacja⟩ ::= ⟨zmienna⟩⟨wyrażenie proste⟩ |
(⟨abstrakcja⟩)⟨wyrażenie proste⟩ | ⟨aplikacja⟩⟨wyrażenie proste⟩

4) ⟨abstrakcja⟩ ::= λ ⟨w. de Bruijna⟩

5) ⟨zmienna⟩ ::= ⟨liczba naturalna⟩

W wyrażeniach tego rodzaju po symbolu λ nie piszemy zmiennej, a na pozostałych pozycjach, w których zwykłe są zmienne, umieszczamy liczby naturalne. Liczb naturalnych jednak nie można utożsamiać ze zmiennymi: w wyrażeniach de Bruijna tylko przekazują informacje o zmiennych. Co więcej, w ustalonym wyrażeniu ta sama liczba przekazuje informacje zależne od jej umiejscowienia i w różnych miejscach może opisywać różne zmienne.

Wyrażenia de Bruijna można też uważać za drzewa. Mogą to być drzewa binarne, które mają trzy rodzaje węzłów: liście, które odpowiadają zmiennym i mają etykiety będące liczbami naturalnymi oraz wskaźniki do dwóch pustych drzew, węzły odpowiadające abstrakcjom, które zamiast lewego poddrzewa mają wskaźnik pusty, i w końcu węzły odpowiadające aplikacjom, które mają dwa wskaźniki do dwóch niepustych poddrzew.

Aby przekształcać wyrażenia w wyrażenia de Bruijna i odwrotnie potrzebny jest tzw. kontekst (słowo kontekst będzie więc miało przynajmniej dwa znaczenia). Kontekst może być rozumiany jako ciąg wszystkich zmiennych bez powtórzeń. Przyjmijmy, że jeżeli Γ jest kontekstem, to Γ_n oznacza zmienną znajdującą się w kontekście Γ na n-tym miejscu, a $\Gamma(x)$ oznacza numer zmiennej x w kontekście Γ. Dalej możemy myśleć, że posługujemy się jednym, ustalonym kontekstem Γ.

1.2 Podstawianie w wyrażeniach de Bruijna

Dla wyrażenia de Bruijna M i liczb naturalnych a i b zdefiniujemy teraz wyrażenie $M[a \leftarrow b]$. Operacja ta ma być odpowiednikiem operacji $M[\Gamma_a := \Gamma_b]$ podstawiania w zwykłym termie M za zmienną Γ_a zmiennej Γ_b.

Przyjmujemy, że

1) $a[a \leftarrow b] = b$ oraz $c[a \leftarrow b] = c$ dla liczb $c \neq a$,

2) $(MN)[a \leftarrow b] = M[a \leftarrow b]N[a \leftarrow b]$,

3) $(\lambda\, M)[a \leftarrow b] = \lambda\, M[a+1 \leftarrow b+1]$.

Definicja tej operacji pozwala częściowo odtworzyć sposób tworzenia wyrażeń de Bruijna. Przekształcając zwykły λ-term (rozumiany jako drzewo) w wyrażenie de Bruijna, konkretne wolne wystąpienie zmiennej x zastępujemy numerem zmiennej x powiększonym o liczbę operatorów λ na ścieżce od tego wystąpienia do korzenia.

1.3 Przekształcanie wyrażeń w wyrażenia de Bruijna

Aby w λ-termach zastępować zmienne liczbami naturalnymi musimy dokładnie wiedzieć, co znaczy liczba n w wyrażeniu de Bruijna. Tak więc wystąpienie liczby 0 oznacza albo zmienną związaną pierwszym (licząc od wystąpienia liczby) operatorem λ znajdującym się na ścieżce od tego wystąpienia do korzenia termu, albo zmienną Γ_0, jeżeli na tej ścieżce nie ma żadnego operatora λ. Wystąpienie liczby 1 oznacza albo zmienną związaną drugim operatorem λ znajdującym się na ścieżce od tego wystąpienia do korzenia, albo zmienną Γ_0, jeżeli na tej ścieżce jest tylko jeden operator λ, albo też zmienną Γ_1, jeżeli na rozważanej ścieżce nie ma żadnego operatora. Dla większych n sytuacja jest analogiczna.

Zdefiniujemy teraz funkcję *Usun_nazwy*, która dane λ-wyrażenie przekształca w odpowiadające mu wyrażenie de Bruijna. Algorytm definiujący tę funkcję będzie rekurencyjny i będzie korzystać z pomocniczej zmiennej h o wartościach naturalnych. Definicję funkcji *Usun_nazwy* można przedstawić w następujący sposób:

> Dane: lambda wyrażenie W (i kontekst Γ).
>
> $Usun\_nazwy(W) = Usun\_nazwy(W, 0)$, a więc wykonanie funkcji *Usun_nazwy* wymaga wywołania innej funkcji z dodatkowym parametrem h, równym 0, niżej mamy rekurencyjną definicję tej funkcji:

1) Jeżeli x jest zmienną, to $Usun\_nazwy(x, h) = \Gamma(x) + h$.

2) $Usun\_nazwy(MN, h) = Usun\_nazwy(M, h)Usun\_nazwy(N, h)$.

3) $Usun\_nazwy(\lambda x\, M, h) = \lambda\, Usun\_nazwy(M, h + 1)[\Gamma(x) + h + 1 \leftarrow 0]$.

Aby zastąpić liczbami zmienne w podtermie nie wystarczy znajomość samego podtermu; musimy znać cały term, a właściwie musimy wiedzieć, ile operatorów λ znajduje się na ścieżce prowadzącej od podtermu do korzenia termu. Liczbę tych operatorów przekazujemy funkcji w formie dodatkowego argumentu. Zauważmy też, że zastępując liczbami zmienne w niewielkim podtermie nie mamy informacji, czy występująca w nim zmienna jest w całym termie wolna, czy też jest związana. Początkowo zmienna jest więc traktowana jako wolna. Zmienne wolne zastępowane są dużymi liczbami w przeciwieństwie do zmiennych związanych. Wobec tego po stwierdzeniu, że zmienna jest jednak związana, musimy zmienić odpowiadającą jej liczbę. Robimy to wykonując stosowne podstawienie.

1.4 Przekształcanie wyrażeń de Bruijna w λ-termy

Teraz zdefiniujemy funkcję *Dodaj_nazwy*, która dane wyrażenie de Bruijna zamienia na zwykłe wyrażenie λ-rachunku. Funkcja ta będzie korzystać z pewnego parametru $c \in N$, od którego będą zależeć wybierane nazwy zmiennych związanych. W ogólnym przypadku jest potrzebna jakaś zasada wyboru nazw tych zmiennych. Będziemy zakładać, że liczba c jest większa od wszystkich liczb występujących w wyrażeniu de Bruijna danym jako argument, i jako zmiennych związanych będziemy używać tych o numerach większych od c.

Funkcję tę można przedstawić w następujący sposób:

Dane: wyrażenie de Bruijna W i kontekst Γ.

$Dodaj\_nazwy(W) = Dodaj\_nazwy(W, 0)$.

1) Jeżeli x jest liczbą, to $Dodaj\_nazwy(x, h) = \Gamma_{x-h}$.

2) $Dodaj\_nazwy(MN, h) = Dodaj\_nazwy(M, h)Dodaj\_nazwy(N, h)$.

3) $Dodaj\_nazwy(\lambda\, M, h) = \lambda\, \Gamma_c Dodaj\_nazwy(M[0 \leftarrow c + h + 1], h + 1)$ i dodatkowo wykorzystanie nazwy (zmiennej) o numerze c powinno spowodować zwiększenie c o 1.

2 α-konwersja a wyrażenia de Bruijna

2.1 Zmienne w wyrażeniach de Bruijna

Rolę zmiennych w termach de Bruijna pełnią liczby naturalne. Będziemy analizować wystąpienia zmiennych w takich wyrażeniach.

Jeżeli wyrażenie de Bruijna uważamy za drzewo, to wystąpieniem zmiennej w tym wyrażeniu będziemy nazywać dowolny liść tego drzewa. Jeżeli liściem tym (w tym liściu) jest liczba x, to będziemy mówić, że jest to wystąpienie liczby x, a nawet to wystąpienie – nie do końca poprawnie – będziemy utożsamiać z liczbą x.

Jeżeli wyrażenie de Bruijna uważamy za ciąg znaków i liczb, to wystąpienie zmiennej x to pozycja w tym ciągu, na której znajduje się liczba x.

Dla każdego wystąpienia zmiennej x w wyrażeniu de Bruijna M definiujemy indukcyjnie zagnieżdżenie $z_M(x)$ tego wystąpienia. Jeżeli M jest zmienną (czyli zmienną x), to $z_M(x) = 0$. Jeżeli $M = M_1M_2$ i wystąpienie x znajduje się w termie M_i, to $z_M(x) = z_{M_i}(x)$. W końcu, jeżeli $M = \lambda M'$, to $z_M(x) = z_{M'}(x) + 1$.

Wystąpienia w termie M zmiennej x nazywamy związanym, jeżeli $x < z_M(x)$. Pozostałe wystąpienia zmiennych nazywamy wolnymi.

Lemat 2.1 *Wykonywanie podstawienia $M[a \leftarrow b]$ polega na zamianie wystąpień liczb x takich, że $x = a + z_M(x)$ liczbami $b + z_M(x)$.* □

Lemat 2.2 *Jeżeli w wyrażeniu de Bruijna M występuje liczba x taka, że $z_M(x) \leqslant x < z_M(x) + h$, to podczas wykonywania procedury $Dodaj\_nazwy(M, h)$ występuje błąd polegający na próbie ustalenia nazwy Γ_c dla pewnego $c < 0$. W przeciwnym razie, jeżeli dla wszystkich wystąpień liczb x w termie M mamy albo $x < z_M(x)$, albo $z_M(x) + h \leqslant x$, to procedura $Dodaj\_nazwy(x, h)$ jest wykonywana poprawnie.* □

Wniosek 2.3 *Procedura $Dodaj\_nazwy(M)$, czyli $Dodaj\_nazwy(M, 0)$ jest wykonywana poprawnie dla dowolnego wyrażenia de Bruijna M.* □

2.2 Występowanie zmiennych w termach de Bruijna

Zdefiniujemy teraz pojęcie, które ma odpowiadać w przypadku termów de Bruijna warunkowi $x \in FV(M)$. Pojęcie to zostanie zdefiniowane przez indukcję ze względu na budowę termu de Bruijna M. Pamiętajmy, że rolę zmiennych w termach de Bruijna pełnią liczby naturalne.

Jeżeli term de Bruijna M jest liczbą naturalną, to liczba x występuje w M wtedy i tylko wtedy, gdy jest równa M. Jeżeli term M jest aplikacją M_1M_2, to liczba x występuje w M wtedy i tylko wtedy, gdy x występuje w M_1 lub w M_2. Jeżeli term M jest abstrakcją λN, to liczba x występuje w M wtedy i tylko wtedy, gdy liczba $x + 1$ występuje w termie N.

Lemat 2.4 *Dla dowolnego termu de Bruijna M i dla dowolnych liczb x, y i $a \neq x$, jeżeli x nie występuje w M, to nie występuje też w termie $M[y \leftarrow a]$.*

Dowód. Lemat ten dowodzimy przez indukcję ze względu na M. Sprawdzimy go jedynie w przypadku abstrakcji $M = \lambda N$.

Załóżmy, że x nie występuje w termie λN. Oznacza to, że $x + 1$ nie występuje w termie N. Term $(\lambda N)[y \leftarrow a]$ jest równy $\lambda N[y + 1 \leftarrow a + 1]$. Aby sprawdzić, że x nie występuje w $\lambda N[y + 1 \leftarrow a + 1]$, badamy, czy $x + 1$ nie występuje w $N[y + 1 \leftarrow a + 1]$. Tak jest na mocy założenia indukcyjnego dla termu N. □

Lemat 2.5 *Jeżeli $x \neq a$, to x nie występuje w termie $M[x \leftarrow a]$.*

Dowód. Przez indukcję ze względu na M dowodzimy, że teza lematu zachodzi dla wszystkich liczb x i a. □

Lemat 2.6 *Jeżeli liczba x nie występuje w termie de Bruijna M, to $M[x \leftarrow a] = M$.*

Dowód. Lemat ten ma oczywisty dowód przez indukcję ze wględu na M. □

2.3 Własności podstawiania

Lemat 2.7 *Jeeli liczby a, a', b, b' speniają warunki $a \neq a'$, $a \neq b'$ oraz $b \neq a'$, to dla wszystkich termów de Bruijna M mamy*

$$M[a \leftarrow b][a' \leftarrow b'] = M[a' \leftarrow b'][a \leftarrow b].$$

Dowód. Lemat dość oczywisty, dowodzony przez indukcję ze względu na budowę termu M, dla wszystkich możliwych parametrów. Najważniejsze, że przechodzi dla elementarnych termów, czyli liczb. „Drugie" kroki są łatwe do wykazania. □

Lemat 2.8 *Jeżeli $a \neq b$, to dla wszystkich termów de Bruijna M mamy*

$$M[a \leftarrow b][a \leftarrow b'] = M[a \leftarrow b].$$

Dowód. Taki jak poprzedni lub z lematów 2.5 i 2.6. □

Lemat 2.9 *Dla wszystkich termów de Bruijna M i liczb b nie występujących w M zachodzi równość*

$$M[a \leftarrow b][b \leftarrow c] = M[a \leftarrow c].$$

Dowód. Taki jak poprzedni. □

2.4 Własności usuwania

Lemat 2.10 *Jeżeli zmienna x nie jest wolna w (zwykłym) termie M, to $\Gamma(x) + h$ nie występuje w termie $Usun\_nazwy(M, h)$.*

Dowód. Również ten lemat łatwo dowodzi się przez indukcję ze względu na budowę termu M. □

Lemat 2.11 *Jeżeli M jest λ-termem i podstawialna za x w M zmienna y nie należy do $FV(M)$, to dla wszystkich liczb h*

$$Usun\_nazwy(M, h)[\Gamma(x) + h \leftarrow \Gamma(y) + h)] = Usun\_nazwy(M[x := y], h).$$

Dowód. Przez indukcję ze względu na M.

Przypadek 1: $M = x$. Wtedy

$$Usun\_nazwy(x, h)[\Gamma(x) + h \leftarrow \Gamma(y) + h] = \Gamma(y) + h.$$

Podobnie przekształcamy drugą stronę wzoru do tego samego rezultatu.

Przypadek 2: zmienną M jest $z \neq x$. Wtedy także $z \neq y$ oraz

$$Usun\_nazwy(z, h)[\Gamma(x) + h \leftarrow \Gamma(y) + h] = \Gamma(z) + h,$$

gdyż dla różnych x i z mamy $\Gamma(x) \neq \Gamma(z)$ (z własności kontekstów). Tak samo przekształcamy drugą stronę wzoru.

Przypadek 3: M jest aplikacją. Teza wynika stąd, że wszystkie rozważane operacje, a więc oba podstawiania $\cdot[\cdot := \cdot]$ oraz $\cdot[\cdot \leftarrow \cdot]$, a także $Usun\_nazwy$, można przestawiać z operacją aplikacji.

Przypadek 4: M jest abstrakcją $\lambda x M'$. Wtedy

$$
\begin{aligned}
&Usun\_nazwy(\lambda x M', h)[\Gamma(x)+h \leftarrow \Gamma(y)+h] = \\
&= (\lambda Usun\_nazwy(M', h+1)[\Gamma(x)]+h+1 \leftarrow 0])[\Gamma(x)+h \leftarrow \Gamma(y)+h] = \\
&= \lambda Usun\_nazwy(M', h+1)[\Gamma(x)]+h+1 \leftarrow 0][\Gamma(x)+h+1 \leftarrow \Gamma(y)+h+1] = \\
&= \lambda Usun\_nazwy(M', h+1)[\Gamma(x)]+h+1 \leftarrow 0] = \\
&= Usun\_nazwy(\lambda x M', h) = Usun\_nazwy((\lambda x M')[x := y], h),
\end{aligned}
$$

na mocy lematu 2.8.

Przypadek 5: M jest abstrakcją $\lambda z M'$ dla $z \neq x$ oraz $x \notin FV(M')$. Wtedy

$$
\begin{aligned}
&Usun\_nazwy(\lambda z M', h)[\Gamma(x)+h \leftarrow \Gamma(y)+h] = \\
&= (\lambda Usun\_nazwy(M', h+1)[\Gamma(z)]+h+1 \leftarrow 0])[\Gamma(x)+h \leftarrow \Gamma(y)+h] = \\
&= \lambda Usun\_nazwy(M', h+1)[\Gamma(z)]+h+1 \leftarrow 0][\Gamma(x)+h+1 \leftarrow \Gamma(y)+h+1] = \\
&= \lambda Usun\_nazwy(M', h+1)[\Gamma(x)]+h+1 \leftarrow 0] = \\
&= Usun\_nazwy(\lambda z M', h) = Usun\_nazwy((\lambda z M')[x := y], h)
\end{aligned}
$$

na mocy lematów z rozdziału 2.2.

Przypadek 6: M jest abstrakcją $\lambda z M'$ dla $z \neq x$ oraz $x \in FV(M')$. Tym razem podstawialność y implikuje, że $y \neq z$. Na mocy lematu 2.7

$$
\begin{aligned}
&Usun\_nazwy(\lambda z M', h)[\Gamma(x)+h \leftarrow \Gamma(y)+h] = \\
&= (\lambda Usun\_nazwy(M', h+1)[\Gamma(z)]+h+1 \leftarrow 0])[\Gamma(x)+h \leftarrow \Gamma(y)+h] = \\
&= \lambda Usun\_nazwy(M', h+1)[\Gamma(z)]+h+1 \leftarrow 0][\Gamma(x)+h+1 \leftarrow \Gamma(y)+h+1] = \\
&= \lambda Usun\_nazwy(M', h+1)[\Gamma(x)+h+1 \leftarrow \Gamma(y)+h+1][\Gamma(z)]+h+1 \leftarrow 0] = \\
&= \lambda Usun\_nazwy(M'[x := y], h+1)[\Gamma(z)+h+1 \leftarrow 0] = \\
&= Usun\_nazwy((\lambda z M')[x := y], h). \ \square
\end{aligned}
$$

Wniosek 2.12 *Przypuśćmy, że mamy dane lambda term M i zmienną y. która nie jest wolna w M i jest podstawialna w M za zmienną x. Wtedy dla dowolnego naturalnego h zachodzi równość*

$$Usun\_nazwy(\lambda x.M, h) = Usun\_nazwy(\lambda y.M[x := y], h).$$

Dowód. Zauważmy, że

$$
\begin{aligned}
&Usun\_nazwy(\lambda y.M[x := y], h) = \\
&= \lambda Usun\_nazwy(M[x := y], h+1)[\Gamma(y)+h+1 \leftarrow 0] = \\
&= \lambda Usun\_nazwy(M, h+1)[\Gamma(x)+h+1 \leftarrow \Gamma(y)+h+1][\Gamma(y)+h+1 \leftarrow 0] = \\
&= \lambda Usun\_nazwy(M, h+1)[\Gamma(x)+h+1 \leftarrow 0] = Usun\_nazwy(\lambda y.M[x := y], h).
\end{aligned}
$$

Poszczególne równości otrzymujemy z lematów 2.11, 2.10 oraz 2.9. $\square$

2.5 Termy h-poprawne

Zaczynamy od pomocniczego pojęcia związanego z operacją *Dodaj_nazwy*. Wykonanie tej operacji może zakończyć się błędem polegającym na otrzymaniu ujemnego argumentu kontekstu. Gwarancją poprawności wykonania operacji $Dodaj\_nazwy(M, h)$ jest h poprawność termu M.

Term de Bruijna M nazywamy h-poprawnym, jeżeli jest on liczbą i $M \geqslant h$, albo jest on aplikacją $M_1 M_2$ i oba jej człony M_1 i M_2 są h poprawne, albo też jest on abstrakcją $\lambda M'$ i term $M'[0 \leftarrow c]$ jest $h+1$-poprawny dla $c \geqslant h+1$ i większego od innych liczb w termie M' (np. dla najmniejszej liczby c o podanych własnościach).

Lemat 2.13 *Jeżeli M jest h-poprawny, to także h-poprawnym jest dowolny term $M[a \leftarrow b+h]$.*

Dowód. Dowodzimy to przez indukcję ze względu na M i dowód jest prosty. □

Z powyższych lematów wynika

Wniosek 2.14 *Jeżeli M jest h-poprawny, to operacja $Dodaj\_nazwy(M, h)$ jest wykonywana poprawnie (nie powoduje błędu).* □

2.6 Usuwanie i dodawanie razem

Lemat 2.15 *Dla dowolnego h i dowolnego termu h-poprawnego termu de Bruijna M mamy*

$$Usun\_nazwy(Dodaj\_nazwy(M, h), h) = M.$$

W szczególności, operacja Usun_nazwy przyjmuje jako wartości wszystkie termy de Bruijna.

Dowód. Przez indukcję ze względu na M.

Jeżeli M jest liczbą, to

$$Usun\_nazwy(Dodaj\_nazwy(M, h), h) = Usun\_nazwy(\Gamma_{M-h}, h) = (M-h)+h = M.$$

Jeżeli M jest aplikacją, to teza lematu wynika w oczywisty sposób z założeń indukcyjnych.

Przypuśćmy, że M jest abstrakcją $\lambda M'$. Wtedy

$$\begin{aligned}
&Usun_nazwy(Dodaj_nazwy(M, h), h) = \\
&= Usun_nazwy(\lambda \Gamma_c Dodaj_nazwy(M'[0 \leftarrow c+h+1], h+1), h) = \\
&= \lambda Usun_nazwy(Dodaj_nazwy(M'[0 \leftarrow c+h+1], h+1), h+1) \\
&\qquad\qquad [\Gamma(\Gamma_c)+h+1 \leftarrow 0] = \\
&= \lambda M'[0 \leftarrow c+h+1][\Gamma(\Gamma_c)+h+1 \leftarrow 0] = \\
&= \lambda M'[0 \leftarrow c+h+1][c+h+1 \leftarrow 0] = \lambda M' = M. \ \square
\end{aligned}$$

Lemat 2.16 *Dla dowolnego h i dowolnego termu h-poprawnego termu de Bruijna M mamy*

$$Dodaj\_nazwy(Usun\_nazwy(M, h), h) \equiv_\alpha M.$$

Dowód. Przez indukcję ze względu na budowę termu M. Jest to oczywiste dla aplikacji. Dla zmiennej M mamy

$$Dodaj\_nazwy(Usun\_nazwy(M,h),h) = Dodaj\_nazwy(\Gamma(M)+h,h) =$$
$$= \Gamma_{\Gamma(M)+h-h} = \Gamma_{\Gamma(M)} = M \equiv_\alpha M.$$

W końcu, dla abstrakcji λxM mamy

$$Dodaj\_nazwy(Usun\_nazwy(\lambda xM,h),h) =$$
$$= Dodaj\_nazwy(\lambda Usun\_nazwy(M,h+1)[\Gamma(x)+h+1 \leftarrow 0],h) =$$
$$= \lambda\Gamma_c Dodaj\_nazwy($$
$$Usun\_nazwy(M,h+1)[\Gamma(x)+h+1 \leftarrow 0][0 \leftarrow c+h+1],h+1)$$
$$= \lambda\Gamma_c Dodaj\_nazwy(Usun\_nazwy(M,h+1)[\Gamma(x)+h+1 \leftarrow c+h+1],h+1)$$
$$= \lambda\Gamma_c Dodaj\_nazwy(Usun\_nazwy(M,h+1),h+1)[x := \Gamma_c]$$
$$= \lambda\Gamma_c M[x := \Gamma_c] \equiv_\alpha M. \ \square$$

Jako wniosek z prowadzonych rozważań otrzymujemy

Twierdzenie 2.17 *Dla każdej pary wyrażeń rachunku lambda M i N, warunek $M \equiv_\alpha N$ jest równoważny równości $Usun\_nazwy(M) = Usun\_nazwy(N)$.* $\square$