Algorytmy usuwania lewostronnej rekursji

7 lipca 2006

1 Algorytmy usuwania lewostronnej rekursji

Dokument niniejszy prezentuje działanie algorytmów usuwających lewostronną rekursję z gramatyk bezkontekstowych. Na przykładowej gramatyce prezentowany jest najpierw podstawowy algorytm Paull'a oraz wpływ numeracji nieterminali na rozmiar wygenerowanej przy jego pomocy gramatyki.
Następnie zaprezentowane są algorytmy polegające na transformacji lewego rogu: Rosenkrantza i Lewis'a oraz Johnsona i ich działanie na wybranej gramatyce.
Rozmiar gramatyki G przyjęto obliczać wg następującego wzoru:
$|G| = \sum_{A \to \alpha \in P} |A\alpha|$, gdzie $|A\alpha|$ oznacza długość łańcucha $A\alpha$.

1.1 Algorytm Paull'a

uporządkuj nieterminale jako $\{ A_1, \ldots, A_n \}$.
for $i := 1$ to n **do**
  for $j := 1$ to $i - 1$ **do**
    for każda produkcja postaci $A_i \to A_j\alpha$ **do**
      usuń produkcję $A_i \to A_j\alpha$ z gramatyki
      for każda produkcja postaci $A_j \to \beta$ **do**
        dodaj $A_i \to \beta\alpha$ do gramatyki
      end for
    end for
  end for
  if A_i jest bezpośrednio lewostronnie rekursywny **then**
    usuń produkcje $A_i \to A_i\alpha_1 \mid \cdots \mid A_i\alpha_r$
    usuń produkcje $A_i \to \beta_1 \mid \cdots \mid \beta_s$ (żadne β_k nie zaczyna się od A_i)
    dodaj produkcje $A_i \to \beta_1 \mid \beta_1 A_i' \mid \cdots \mid \beta_s \mid \beta_s A_i'$
    dodaj produkcje $A_i' \to \alpha_1 \mid \alpha_1 A_i' \mid \cdots \mid \alpha_r \mid \alpha_r A_i'$
  end if
end for

1.2 Relacja lewego rogu

Relacja lewego rogu jest potrzebna do algorytmów transformacji lewego rogu. Nie jest konieczna przy algorytmie Paull'a ale pozwala wybrać najlepszą i najgorszą numerację nieterminali, co będzie miało wpływ na rozmiar wygenerowanej gramatyki.
Najpierw wyznacza się relację bezpośredniego lewego rogu, która wynika bezpośrednio z produkcji gramatyki, a następnie dokonuje się domknięcia przechodniego (i zwrotnego) relacji.

Bezpośredni lewy róg (*direct left corner*)

Symbol X jest bezpośrednim lewym rogiem nieterminala A, jeśli istnieje A-produkcja, mająca prawą stronę zaczynającą się od symbolu X tzn. zachodzi $A \to X\beta$.

Oznaczamy: $X \lhd A$
Dla wygody będziemy oznaczać $A \rhd X \Leftrightarrow X \lhd A$.
Nieterminal jest bezpośrednio lewostronnie rekursywny (*directly left recursive*), jeśli jest bezpośrenim lewym rogiem siebie samego.

Relacja lewego rogu (*left-corner relation*)

To zwrotne i przechodnie domknięcie relacji bezpośredniego lewego rogu.
Oznaczamy: $X \lhd^\star A$

Relacja właściwego lewego rogu (*proper-left-corner relation*)

To przechodnie domknięcie relacji bezpośredniego lewego rogu.
Oznaczamy: $X \lhd^+ A$
Nieterminal jest lewostronnie rekursywny (*left recursive*), jeśli jest właściwym lewym rogiem siebie samego.

1.3 Transformacje lewego rogu

Przekształcenie lewego rogu może być w prosty sposób zaimplementowane jako zbiór reguł Prologu.

1.3.1 Algorytm Rosenkrantza i Lewisa

Algorytm Rosenkrantza i Lewisa opiera się na (przechodniej i zwrotnej) relacji lewego rogu (*left-corner*), oznaczanej $\lhd^\star$ natomiast algorytm Johnsona na relacji właściwego lewego rogu (*proper-left-corner*), oznaczanej $\lhd^+$. W obu przypadkach transformacja prowadzi do utworzenia nowej gramatyki, przy czym nieistotna jest tu ani kolejność numeracji, ani kolejność wykonywania poszczególnych przekształceń. Generując kolejne produkcje opieramy się tylko na oryginalnej gramatyce, stąd bez względu na wyżej wymienione czynniki, uzyskana gramatyka będzie zawsze taka sama.

Dodawana produkcja	Warunek
$A \to aA - a$	$a \lhd^\star A$
$A \to A - C$	$C \to \epsilon$
$A - X \to \beta A - B$	$B \lhd^\star A,\ B \to X\beta$
$A - A \to \epsilon$	

1.3.2 Algorytm Johnsona

Dodawana produkcja	Warunek
$A \to aA - a$	$a \lhd^+ A$
$A \to A - C$	$C \to \epsilon$
$C \to \epsilon$	$C \to \epsilon$
$A - X \to \beta A - B$	$B \lhd^+ A,\ B \to X\beta$
$A - X \to \beta$	$X \lhd^+ A,\ A \to X\beta$

1.3.3 Algorytm LC_{LR}

Jest to modyfikacja algorytmu Johnsona, która ma na celu tylko usunięcie lewostronnej rekursji, a nie tak jak oryginalnie algorytmy lewego rogu - zmniejszenie drzewa rozbioru syntaktycznego. Opiera się on na założeniu, że wprowadzenie do gramatyki nieterminala $\langle AX \rangle$ ma sens tylko wtedy gdy spełnia wszystkie trzy poniższe warunki:

- symbol X jest właściwym lewym rogiem A ($X \lhd^+ A$).
- A spełnia
  - jest symbolem początkowym gramatyki lub
  - występuje po prawej stronie którejś produkcji nie w lewym rogu: $K \to \beta_1 A \beta_2 \in P$, $\beta_1 \neq \epsilon$ lub
  - występuje po prawej stronie producji nieterminala który nie jest lewostronnie rekursywny: $K \to \beta_1 A \beta_2 \in P$, $\neg(K \lhd^+ K)$
- nieterminal A jest lewostronnie rekursywny

Koniunkcję powyższych warunków oznaczmy (*), ponieważ będziemy się do niej wielokrotnie odwoływać podczas przebiegu algorytmu.
Ponadto algorytm ten traktuje nieterminale które nie są lewostronnie rekursywne tak jak terminale.

Oryginalna produkcja	Dodawana produkcja	Warunek
	$A \to X\langle AX \rangle$	$\langle AX \rangle$ spełnia warunki (*) $X \in T \vee X \in N$ i nie jest lewostronnie rekursywny
$B \to X\beta$	$\langle AX \rangle \to \beta\langle AB \rangle$	$\langle AB \rangle$ spełnia warunki (*) $B \in N$ lewostronnie rekursywny
$A \to X\beta$	$\langle AX \rangle \to \beta$	$\langle AX \rangle$ spełnia warunki (*) $X \in N \cup T$
$A \to \beta$	$A \to \beta$	$A \in N$ nie jest lewostronnie rekursywny

1.4 Dodatkowe usprawnienia algorytmów

1.4.1 Lewostronna faktoryzacja (*LF*)

Lewostronna faktoryzacja polega na eliminacji produkcji jednego nieterminala zaczynających się po prawej stronie od tego samego symbolu, aby uniknąć niejednoznaczności.

repeat
  for każdy nieterminal A **do**
    niech α będzie najdłuższą sekwencją taką, że istnieje więcej niż jedna A-produkcja postaci $A \to \alpha\beta$
    zastąp produkcje $A \to \alpha\beta_1 \mid \cdots \mid \alpha\beta_n \mid \gamma_1 \mid \cdots \mid \gamma_m$
    produkcjami $A \to \alpha A'' \mid \gamma_1 \mid \cdots \mid \gamma_m$
    $A'' \to \beta_1 \mid \cdots \mid \beta_n$
  end for
until nie ma żadnych zmian w gramatyce

1.4.2 Grupowanie produkcji bez lewostronnej rekursji (*NLRG*)

Przy usuwaniu lewostronnej rekursji, często powiela się produkcje nieterminali lewostronnie rekursywnych, warto ograniczyć ich liczbę do minimum. Oprócz wspomnianej lewostronnej faktoryzacji można wszystkie produkcje nierekursywne takiego nieterminala zastąpić jedną, używając do tego pomocniczego symbolu nieterminalnego.

for każdy lewostronnie rekursywny nieterminal A **do**
zastąp produkcje $A \to \alpha_1 \mid \cdots \mid \alpha_n$ (żadne α_k nie zaczyna się od lewostronnie rekursywnego nieterminala, $n \geq 2$)
produkcjami $A \to A'''$
$A''' \to \alpha_1 \mid \cdots \mid \alpha_n$
end for

2 Przykład

Przedmiotem naszej analizy będzie gramatyka prostych wyrażeń arytmetycznych zdefiniowana następująco:

$E \to E + T \mid T$
$T \to T \star F \mid F$
$F \to F \uparrow P \mid P$
$P \to (E) \mid a$

Widać, że występuje tu lewostronna rekursja bezpośrednia dla nieterminali E, T oraz F.
Rekursja pośrednia nie występuje.
Poniższa tabela przedstawia wynik wyznaczenia relacji lewego rogu.

nieterminal	$\triangleright$	$\triangleright^+$	$\triangleright^\star$
E	E, T	$E, T, F, P, (, a$	$E, T, F, P, (, a$
T	T, F	$T, F, P, (, a$	$T, F, P, (, a$
F	F, P	$F, P, (, a$	$F, P, (, a$
P	$(, a$	$(, a$	$P, (, a$

2.1 Algorytm Paull'a

2.1.1 Optymalna numeracja nieterminali

Aby wygenerowana gramatyka miała jak najmniejszy rozmiar, należy ponumerować nieterminale zgodnie z nierosnącą liczbą lewych rogów ($\triangleleft^*$) danego nieterminala. Jedyną taką numeracją jest:

$E(1), T(2), F(3), P(4)$

Aby lepiej było widać poszczególne produkcje gramatyki zapiszmy ją w postaci:

$E \to E + T$
$E \to T$
$T \to T \star F$
$T \to F$

$F \to F \uparrow P$
$F \to P$
$P \to (E)$
$P \to a$

Gramatyka składa się z 8 produkcji i potrzeba 24 symboli terminalnych i nieterminalnych do jej zapisania w powyższej postaci.
Działanie algorytmu przebiegać będzie następująco:

iteracja	produkcja usuwana	produkcje dodawane
$i = 1,\ j = 1$	$E \to E + T$	$E \to TE'$ $E' \to +T$ $E' \to +TE'$
$i = 2,\ j = 2$	$T \to T \star F$	$T \to FT'$ $T' \to \star F$ $T' \to \star FT'$
$i = 3,\ j = 3$	$F \to F \uparrow P$	$F \to PF'$ $F' \to \uparrow PF'$ $F' \to \uparrow P$

Ostatecznie wygenerowana gramatyka ma postać:
$E \to TE' \mid T$
$E' \to +TE' \mid +T$
$T \to FT' \mid F$
$T' \to \star FT' \mid \star F$
$F \to PF' \mid P$
$F' \to \uparrow PF' \mid \uparrow P$
$P \to (E) \mid a$

Wygenerowana gramatyka składa się z 14 produkcji i potrzeba 42 symboli terminalnych i nieterminalnych do jej zapisania w powyższej postaci.

2.1.2 Antyoptymalna numeracja nieterminali

Aby wygenerowana gramatyka miała jak największy rozmiar, należy ponumerować nieterminale zgodnie z niemalejącą liczbą lewych rogów danego nieterminala. Jedyną taką numeracją jest

$E(4), T(3), F(2), P(1)$

Działanie algorytmu przy takiej numeracji przebiegać będzie następująco:

iteracja	produkcja usuwana	produkcje dodawane
$i = 2,\ j = 1$	$F \to P$	$F \to (E)$ $F \to a$
$i = 2,\ j = 2$	$F \to F \uparrow P$	$F \to (E)F'$ $F \to aF'$ $F' \to \uparrow P$ $F' \to \uparrow PF'$
$i = 3,\ j = 2$	$T \to F$	$T \to (E)$ $T \to a$ $T \to (E)F'$ $T \to aF'$
$i = 3,\ j = 3$	$T \to T \star F$	$T \to (E)T'$ $T \to aT'$ $T \to (E)F'T'$ $T \to aF'T'$ $T' \to \star F$ $T' \to \star FT'$
$i = 4,\ j = 3$	$E \to T$	$E \to (E)$ $E \to a$ $E \to (E)F'$ $E \to aF'$ $E \to (E)T'$ $E \to aT'$ $E \to (E)F'T'$ $E \to aF'T'$
$i = 4,\ j = 4$	$E \to E + T$	$E \to (E)E'$ $E \to aE'$ $E \to (E)F'E'$ $E \to aF'E'$ $E \to (E)T'E'$ $E \to aT'E'$ $E \to (E)F'T'E'$ $E \to aF'T'E'$ $E' \to +T$ $E' \to +TE'$

Ostatecznie wygenerowana gramatyka składa się z 36 produkcji i zapisana jest przy pomocy 145 symboli terminalnych i nieterminalnych.

$P \to (E) \mid a$
$F \to (E) \mid a \mid (E)F' \mid aF'$
$F' \to \uparrow P \mid \uparrow PF'$
$T \to (E) \mid a \mid (E)F' \mid aF' \mid (E)T' \mid aT' \mid (E)F'T' \mid aF'T'$
$T' \to \star F \mid \star FT'$
$E \to (E) \mid a \mid (E)F' \mid aF' \mid (E)T' \mid aT' \mid (E)F'T' \mid aF'T' \mid (E)E' \mid aE' \mid (E)F'E' \mid aF'E' \mid$
$\quad (E)T'E' \mid aT'E' \mid (E)F'T'E' \mid aF'T'E'$
$E' \to +T \mid +TE'$

2.2 Algorytmy Rosenkrantza i Lewisa oraz Johnsona

Jednocześnie wykonane zostaną transformacje lewego rogu (*left-corner transformation*) obydwoma algorytmami.
Metoda Rosenkrantza i Lewisa opiera się na (przechodniej i zwrotnej) relacji lewego rogu (*left-corner*), oznaczanej $\lhd^\star$ natomiast algorytm Johnsona na relacji właściwego lewego rogu (*proper-left-corner*), oznaczanej $\lhd^+$.
Początkowo zbiór produkcji nowej gramatyki jest pusty. Do nowej gramatyki dołączane są produkcje zgodnie z poniższymi regułami.

Krok 1. $\forall A \in N$, $\forall a \in T$ takich, że zachodzi $a \lhd^\star (\lhd^+) A$ dołącz produkcje $A \to a\langle Aa\rangle$.

Dla gramatyki prostych wyrażeń arytmetycznych algorytm Rosenkrantza i Lewisa wygeneruje w tym kroku następujące produkcje:

$E \to (\langle E(\rangle$
$E \to a\langle Ea\rangle$
$T \to (\langle T(\rangle$
$T \to a\langle Ta\rangle$
$F \to (\langle F(\rangle$
$F \to a\langle Fa\rangle$
$P \to (\langle P(\rangle$
$P \to a\langle Pa\rangle$

Algorytm Johnsona w tym kroku da dokładnie ten sam rezultat, ponieważ relacja $\lhd^+$ nie ogranicza relacji $\lhd^\star$ względem terminali.

Krok 2. $\forall A \in N$, $\forall B \in N$ takich, że zachodzi $B \lhd^\star (\lhd^+) A$ oraz $\forall B \to X\beta \in P$ gdzie $X \in N \cup T$, $\beta \in (N \cup T)^\star$ dołącz produkcje $\langle AX\rangle \to \beta\langle AB\rangle$.

Dla rozważanej gramatyki algorytm wygeneruje w tym kroku następujące produkcje:

A	B	$B \to X\beta$	dołączamy $\langle AX\rangle \to \beta\langle AB\rangle$
E	E	$E \to E + T$	$\langle EE\rangle \to +T\langle EE\rangle$
		$E \to T$	$\langle ET\rangle \to \langle EE\rangle$
	T	$T \to T \star F$	$\langle ET\rangle \to \star F\langle ET\rangle$
		$T \to F$	$\langle EF\rangle \to \langle ET\rangle$
	F	$F \to F \uparrow P$	$\langle EF\rangle \to\uparrow P\langle EF\rangle$
		$F \to P$	$\langle EP\rangle \to \langle EF\rangle$
	P	$P \to (E)$	$\langle E(\rangle \to E)\langle EP\rangle$
		$P \to a$	$\langle Ea\rangle \to \langle EP\rangle$
T	T	$T \to T \star F$	$\langle TT\rangle \to \star F\langle TT\rangle$
		$T \to F$	$\langle TF\rangle \to \langle TT\rangle$
	F	$F \to F \uparrow P$	$\langle TF\rangle \to\uparrow P\langle TF\rangle$
		$F \to P$	$\langle TP\rangle \to \langle TF\rangle$
	P	$P \to (E)$	$\langle T(\rangle \to E)\langle TP\rangle$
		$P \to a$	$\langle Ta\rangle \to \langle TP\rangle$
F	F	$F \to F \uparrow P$	$\langle FF\rangle \to\uparrow P\langle FF\rangle$
		$F \to P$	$\langle FP\rangle \to \langle FF\rangle$
	P	$P \to (E)$	$\langle F(\rangle \to E)\langle FP\rangle$
		$P \to a$	$\langle Fa\rangle \to \langle FP\rangle$
P	P	$P \to (E)$	$\langle P(\rangle \to E)\langle PP\rangle$
		$P \to a$	$\langle Pa\rangle \to \langle PP\rangle$

Algorytm Johnsona da podobny rezultat, nie dołączy jedynie ostatnich dwóch produkcji, ponieważ nie zachodzi relacja $P \lhd^+ P$ (P nie jest właściwym lewym rogiem samego siebie).

Krok 3. Krok ten stanowi największą różnicę pomiędzy dwoma omawianymi transformacjami lewego rogu. W przypadku algorytmu Rosenkrantza i Lewisa krok ten wygląda następująco:
$\forall A \in N$ dołącz produkcję postaci $\langle AA\rangle \to \epsilon$.

Zatem dla naszej gramatyki wygeneruje produkcje:

$\langle EE\rangle \to \epsilon$
$\langle TT\rangle \to \epsilon$
$\langle FF\rangle \to \epsilon$
$\langle PP\rangle \to \epsilon$

Natomiast dla algorytmu Johnsona krok ten wygląda następująco:
$\forall A \in N$, $\forall X \in N \cup T$ takich, że $A \to X\beta \in P$, $\beta \in (N \cup T)^\star$ dołącz produkcję $\langle AX\rangle \to \beta$.

Dla naszej gramatyki wygeneruje produkcje:

$A \to X\beta$	dołączamy $\langle AX\rangle \to \beta$
$E \to E + T$	$\langle EE\rangle \to +T$
$E \to T$	$\langle ET\rangle \to \epsilon$
$T \to T \star F$	$\langle TT\rangle \to \star F$
$T \to F$	$\langle TF\rangle \to \epsilon$
$F \to F \uparrow P$	$\langle FF\rangle \to\uparrow P$
$F \to P$	$\langle FP\rangle \to \epsilon$
$P \to (E)$	$\langle P(\rangle \to E)$
$P \to a$	$\langle Pa\rangle \to \epsilon$

Rozmiar gramatyk wygenerowanych obydwoma algorytmami jest podobny.
Gramatyka po transformacji Rosenkrantza i Lewisa składa się z 32 produkcji, a do jej pełnego zapisu potrzeba 92 symboli terminalnych bądź nieterminalnych.
Gramatyka po transformacji Johnsona jest nieco większa, ponieważ składa się z 34 produkcji, a do jej pełnego zapisu potrzeba 98 symboli.

2.3 Podsumowanie

Porównanie wielkości gramatyk wygenerowanych przy pomocy poszczególnych algorytmów przedstawia poniższa tabela.

Algorytm	Ilość produkcji	Ilość symboli
Paull'a (najlepsza kolejność numeracji nieterminali)	14	41
Paull'a (najgorsza kolejność numeracji nieterminali)	36	145
Rosenkrantza i Lewisa	32	92
Johnsona	34	98

3 Przykład

Przykład niniejszy prezentuje działanie algorytmów usuwających lewostronną rekursję z gramatyk bezkontekstowych. Celem tej części jest przedstawienie przewagi algorytmów polegających na transformacji lewego rogu: Rosenkrantza i Lewis'a oraz Johnsona nad podstawowym algorytmem Paull'a.
Przedmiotem analizy będzie utworzona w sposób systematyczny gramatyka:

$A \to Ba \mid Bb \mid Bc$
$B \to Ad \mid Ae \mid Af \mid z$

Występuje tutaj tylko lewostronna rekursja pośrednia dla nieterminali A i B. Poniższa tabela przedstawia wynik wyznaczenia relacji lewego rogu.

nieterminal	$\triangleright$	$\triangleright^+$	$\triangleright^\star$
A	B	A, B, z	A, B, z
B	A, z	A, B, z	A, B, z

W tym przypadku obydwa terminale naszej gramatyki mają jednakowe zbiory symboli z którymi są w relacji lewego rogu.
Taka sytuacja zachodzi w przypadku cykli w relacji lewego rogu, a więc zawsze wówczas, kiedy występuje lewostronna rekursja pośrednia.
Każdy nieterminal należący do cyklu, jest w relacji lewego rogu z każdym elementem cyklu.
W takim przypadku, należy wśród rozpatrywanych symboli ustalić kolejność numeracji według niemalejącej liczby produkcji.

3.1 Algorytm Paull'a

3.1.1 Optymalna numeracja nieterminali

Ponieważ tak jak napisaliśmy powyżej relacja lewego rogu nie jest tu rozstrzygająca, a nieterminal B posiada więcej produkcji w oryginalnej gramatyce, najkorzystniej przyjąć że będzie miał on numer ostatni. Zatem mamy numerację:

$A(1), B(2)$

Aby lepiej było widać poszczególne produkcje naszej gramatyki zapiszmy ją w postaci:

$A \to Ba$
$A \to Bb$
$A \to Bc$
$B \to Ad$
$B \to Ae$
$B \to Af$
$B \to z$

Gramatyka składa się z 7 produkcji i potrzeba 20 symboli terminalnych i nieterminalnych do jej zapisania w powyższej postaci.
Działanie algorytmu przebiegać będzie następująco:

iteracja	produkcja usuwana	produkcje dodawane
$i = 2,\ j = 1$	$B \to Ad$	$B \to Bad$
		$B \to Bbd$
		$B \to Bcd$
	$B \to Ae$	$B \to Bae$
		$B \to Bbe$
		$B \to Bce$
	$B \to Af$	$B \to Baf$
		$B \to Bbf$
		$B \to Bcf$
$i = 2,\ j = 2$	$B \to Bad$	$B \to zB'$
		$B' \to ad$
		$B' \to adB'$
	$B \to Bbd$	$B' \to bd$
		$B' \to bdB'$
	$B \to Bcd$	$B' \to cd$
		$B' \to cdB'$
	$B \to Bae$	$B' \to ae$
		$B' \to aeB'$
	$B \to Bbe$	$B' \to be$
		$B' \to beB'$
	$B \to Bce$	$B' \to ce$
		$B' \to ceB'$
	$B \to Baf$	$B' \to af$
		$B' \to afB'$
	$B \to Bbf$	$B' \to bf$
		$B' \to bfB'$
	$B \to Bcf$	$B' \to cf$
		$B' \to cfB'$

Ostatecznie wygenerowana gramatyka ma następującą postać:

$A \to Ba \mid Bb \mid Bc$
$B \to z \mid zB'$
$B' \to ad \mid adB' \mid bd \mid bdB' \mid cd \mid cdB' \mid ae \mid aeB' \mid be \mid beB' \mid ce \mid ceB' \mid af \mid afB' \mid bf \mid bfB' \mid$
$cf \mid cfB'$

Wygenerowana gramatyka składa się z 23 produkcji i potrzeba jej w pełnym zapisie 77 symboli terminalnych i nieterminalnych.

3.1.2 Antyoptymalna numeracja nieterminali

Drugą i jednocześnie najgorszą kolejnością numeracji nieterminali jest:

$A(2), B(1)$

Działanie algorytmu przy takiej numeracji przebiegać będzie następująco:

iteracja	produkcja usuwana	produkcje dodawane
$i = 2,\ j = 1$	$A \to Ba$	$A \to Ada$ $A \to Aea$ $A \to Afa$ $A \to za$
	$A \to Bb$	$A \to Adb$ $A \to Aeb$ $A \to Afb$ $A \to zb$
	$A \to Bc$	$A \to Adc$ $A \to Aec$ $A \to Afc$ $A \to zc$
$i = 2,\ j = 2$	$A \to Ada$	$A \to zaA'$ $A' \to da$ $A' \to daA'$
	$A \to Aea$	$A' \to ea$ $A' \to eaA'$
	$A \to Afa$	$A' \to fa$ $A' \to faA'$
	$A \to Adb$	$A \to zbA'$ $A' \to db$ $A' \to dbA'$
	$A \to Aeb$	$A' \to eb$ $A' \to ebA'$
	$A \to Afb$	$A' \to fb$ $A' \to fbA'$
	$A \to Adc$	$A \to zcA'$ $A' \to dc$ $A' \to dcA'$
	$A \to Aec$	$A' \to ec$ $A' \to ecA'$
	$A \to Afc$	$A' \to fc$ $A' \to fcA'$

Ostatecznie wygenerowana gramatyka składa się z 28 produkcji i zapisana jest przy pomocy 95 symboli terminalnych i nieterminalnych.
Oto jej skrócony zapis:

$B \to Ad \mid Ae \mid Af \mid z$
$A \to za \mid zaA' \mid zb \mid zbA' \mid zc \mid zcA'$
$A' \to da \mid daA' \mid ea \mid eaA' \mid fa \mid faA' \mid db \mid dbA' \mid eb \mid ebA' \mid fb \mid fbA' \mid dc \mid dcA' \mid ec \mid ecA' \mid fc \mid fcA'$

3.2 Algorytmy Rosenkrantza i Lewisa oraz Johnsona

Krok 1. $\forall A \in N$, $\forall a \in T$ takich, że zachodzi $a \lhd^\star (\lhd^+) A$ dołącz produkcje $A \to a\langle Aa \rangle$.

Dla naszej gramatyki algorytm Rosenkrantza i Lewisa, jak również algorytm Johnsona wygenerują w tym kroku następujące produkcje:

$A \to z\langle Az\rangle$
$B \to z\langle Bz\rangle$

Krok 2. $\forall A \in N, \forall B \in N$ takich, że zachodzi $B \lhd^\star (\lhd^+)A$ oraz $\forall B \to X\beta \in P$ gdzie $X \in N \cup T$, $\beta \in (N \cup T)^\star$ dołącz produkcje $\langle AX\rangle \to \beta\langle AB\rangle$.

Dla naszej gramatyki algorytmy Rosenkrantza i Lewisa, jak również Johnsona wygenerują w tym kroku te same produkcje.
Zachodzi to tylko dla gramatyk, w których relacje $\lhd^\star$ oraz $\lhd^+$ pokrywają się.

A	B	$B \to X\beta$	dołączamy $\langle AX\rangle \to \beta\langle AB\rangle$
A	A	$A \to Ba$	$\langle AB\rangle \to a\langle AA\rangle$
		$A \to Bb$	$\langle AB\rangle \to b\langle AA\rangle$
		$A \to Bc$	$\langle AB\rangle \to c\langle AA\rangle$
	B	$B \to Ad$	$\langle AA\rangle \to d\langle AB\rangle$
		$B \to Ae$	$\langle AA\rangle \to e\langle AB\rangle$
		$B \to Af$	$\langle AA\rangle \to f\langle AB\rangle$
		$B \to z$	$\langle Az\rangle \to \langle AB\rangle$
B	A	$A \to Ba$	$\langle BB\rangle \to a\langle BA\rangle$
		$A \to Bb$	$\langle BB\rangle \to b\langle BA\rangle$
		$A \to Bc$	$\langle BB\rangle \to c\langle BA\rangle$
	B	$B \to Ad$	$\langle BA\rangle \to d\langle BB\rangle$
		$B \to Ae$	$\langle BA\rangle \to e\langle BB\rangle$
		$B \to Af$	$\langle BA\rangle \to f\langle BB\rangle$
		$B \to z$	$\langle Bz\rangle \to \langle BB\rangle$

Krok 3. Krok ten stanowi w przypadku omawianej gramatyki jedyną różnicę pomiędzy dwoma stosowanymi transformacjami lewego rogu.
W przypadku algorytmu Rosenkrantza i Lewisa krok ten wygląda następująco:
$\forall A \in N$ dołącz produkcję postaci $\langle AA\rangle \to \epsilon$.
Dla rozważanej gramatyki wygenerowane zostaną następujące produkcje:

$\langle AA\rangle \to \epsilon$
$\langle BB\rangle \to \epsilon$

Natomiast algorytm Johnsona, w którym krok ten wygląda następująco:
$\forall A \in N, \forall X \in N \cup T$ takich, że $A \to X\beta \in P$, $\beta \in (N \cup T)^\star$ dołącz produkcję $\langle AX\rangle \to \beta$
dla naszej gramatyki wygeneruje następujące produkcje:

$\langle AB\rangle \to a$
$\langle AB\rangle \to b$
$\langle AB\rangle \to c$
$\langle BA\rangle \to d$
$\langle BA\rangle \to e$
$\langle BA\rangle \to f$
$\langle Bz\rangle \to \epsilon$

Rozmiar gramatyk wygenerowanych obydwoma algorytmami jest podobny. Gramatyka po transformacji Rosenkrantza i Lewisa składa się z 18 produkcji, a do jej pełnego zapisu potrzeba 50

symboli terminalnych i nieteminalnych. Gramatyka po transformacji Johnsona jest nieco większa, ponieważ składa się z 23 produkcji, a do jej pełnego zapisu potrzeba 60 symboli.

Po usunięciu symboli nieosiągalnych gramatyka wygenerowana algorytmem Rosenkrantza i Lewisa wygląda następująco:

$A \to z\langle Az\rangle$
$\langle Az\rangle \to \langle AB\rangle$
$\langle AB\rangle \to a\langle AA\rangle \mid b\langle AA\rangle \mid c\langle AA\rangle$
$\langle AA\rangle \to d\langle AB\rangle \mid e\langle AB\rangle \mid f\langle AB\rangle \mid \epsilon$

Natomiast po usunięciu symboli nieosiągalnych gramatyka wygenerowana algorytmem Johnsona wygląda następująco:

$A \to z\langle Az\rangle$
$\langle Az\rangle \to \langle AB\rangle$
$\langle AB\rangle \to a\langle AA\rangle \mid b\langle AA\rangle \mid c\langle AA\rangle$
$\langle AA\rangle \to d\langle AB\rangle \mid e\langle AB\rangle \mid f\langle AB\rangle \mid a \mid b \mid c$

3.3 Podsumowanie

Porównanie wielkości gramatyk wygenerowanych przy pomocy poszczególnych algorytmów i ich wariantów przedstawia poniższa tabela.

Algorytm	Ilość produkcji(Ilość symboli)	Po usunięciu symboli nieosiągalnych
Paull'a (lepsza kolejność numeracji nieterminali)	23(77)	
Paull'a (gorsza kolejność numeracji nieterminali)	28(95)	
Rosenkrantza i Lewisa	18(50)	9(25)
Johnsona	23(60)	11(29)

Widać przewagę algorytmów opartych o transformację lewego rogu, choć nie jest to przewaga znaczna. Dla większych gramatyk o podobnej budowie ta przewaga jest wyraźniejsza. Na przykład gramatyka typu:

$A_1 \to a \mid A_n\beta_{1,1} \mid A_n\beta_{1,2} \mid \cdots \mid A_n\beta_{1,k}$
$A_i \to A_{i-1}\beta_{i,1} \mid A_{i-1}\beta_{i,2} \mid \cdots \mid A_{i-1}\beta_{i,k} \quad i \in \{\, 2, 3, \ldots, n \,\}$

ma rozmiar $k * n + 1$ produkcji w oryginalnej postaci. Wszystkie nieterminale tej gramatyki tworzą cykl rekursji pośredniej.

Algorytm Paull'a dla tej gramatyki będzie dodatkowo generować około $2 * k^n + 1$ produkcji, usuwając przy tym tylko k. Dla naszej gramatyki (przy $k = 3, n = 2$) daje łącznie $7+19-3 = 23$ produkcje w nowej gramatyce. Natomiast dla $k = 4, n = 4$ otrzymujemy olbrzymią liczbę $17 + 513 - 4 = 526$ produkcji. Jest to liczba podana dla optymalnej numeracji terminali.

Dla tej samej gramatyki algorytm Rosenrantza i Lewisa da kolejno: n produkcji w kroku 1, $(k * n + 1) * n$ produkcji w kroku 2 oraz n produkcji w kroku 3. Dla naszej gramatyki (przy $k = 3, n = 2$) daje łącznie $2 + 14 + 2 = 18$ produkcje w nowej gramatyce. Natomiast dla $k = 4, n = 4$ otrzymujemy liczbę $4 + (4 * 4 + 1) * 4 + 4 = 76$ produkcji.

Algorytm Johnsona wygeneruje kolejno: n produkcji w kroku 1, $(k*n+1)*n$ produkcji w kroku 2 oraz $k * n + 1$ produkcji w kroku 3. Dla naszej gramatyki (przy $k = 3, n = 2$) daje łącznie

$2 + 14 + 7 = 23$ produkcje w nowej gramatyce. Natomiast dla $k = 4, n = 4$ otrzymujemy liczbę $4 + (4 * 4 + 1) * 4 + 17 = 89$ produkcji.
Wyliczenia te przedstawia zbiorczo poniższa tabela. Kolumna dla algorytmu Paull'a zawiera rozmiary wygenerowanych przez algorytm gramatyk dla najlepszej możliwej numeracji terminali.

k	n	oryginalna	Paull	R. & L.	Johnson
3	2	7	23	18	23
4	4	17	526	76	89

4 Przykład

Przykład niniejszy prezentuje działanie algorytmów usuwających lewostronną rekursję z gramatyk bezkontekstowych. Przedstawione są również algorytmy hybrydowe. Do podstawowych algorytmów Paull'a i lewego rogu można dodać lewostronną faktoryzację (LF) lub przekształcenie $NLRG$ (*non-left-recursion grouping*). Ponadto istnieje odmiana algorytmu Johnsona przeznaczona jedynie do usuwania lewostronnej rekursji.
Następnie usuwane są symbole nieosiągalne i ϵ-produkcje z gramatyk wygenerowanych transformacjami lewego rogu. Algorytm Paull'a nie generuje dodatkowych symboli nieosiągalnych ani ϵ-produkcji, podobnie przekształcenia LF oraz $NLRG$.

Przedmiotem naszej analizy będzie utworzona w sposób systematyczny gramatyka:

$A \to Bz \mid By \mid a \mid b$
$B \to Aw \mid Au \mid c \mid d$

Występuje tutaj tylko lewostronna rekursja pośrednia dla nieterminali A i B.
Poniższa tabela przedstawia wynik wyznaczenia relacji lewego rogu

nieterminal	$\triangleright$	$\triangleright^+$
A	B, a, b	A, B, a, b, c, d
B	A, c, d	A, B, a, b, c, d

Rozważane będą także algorytmy usuwania lewostronnej rekursji dla których wejściem nie będzie oryginalna gramatyka G, lecz gramatyka poddana lewostronnej faktoryzacji lub lewostronnej faktoryzacji i przekształceniu $NLRG$.

Lewostronna faktoryzacja (*LF*)

Aby dokonać lewostronnej faktoryzacji produkcji $A \to Bz \mid By$ wprowadza się dodatkowy nieterminal (oznaczmy go A'' dla odróżnienia od uzyskiwanego w algorytmie Paull'a A'), który będzie reprezentował wszystkie możliwe kontynuacje tej produkcji, w następujący sposób:

$A \to BA''$
$A'' \to z \mid y$

Tej samej zmiany musimy dokonać dla każdej występującej w gramatyce tego typu niejednoznaczności, a zatem w naszym przypadku jeszcze dla nieterminala B i prawych stron zaczynających się od A. Otrzymujemy:

$B \to AB''$
$B'' \to w \mid u$

Ostatecznie gramatyka po przekształceniu LF składa się z 10 produkcji, 22 symboli i wygląda następująco:

$A \to BA'' \mid a \mid b$
$A'' \to z \mid y$
$B \to AB'' \mid c \mid d$
$B'' \to w \mid u$

Grupowanie produkcji bez lewostronnej rekursji (*NLRG*)

W naszym przypadku przekształcenie będzie przebiegało następująco:

produkcje usuwane	produkcje dodawane
$A \to a \mid b$	$A \to A'''$ $A''' \to a \mid b$
$B \to c \mid d$	$B \to B'''$ $B''' \to c \mid d$

Po przeprowadzeniu przekształceń LF oraz $NLRG$ otrzymamy następującą gramatykę o rozmiarze 12 produkcji(26 symboli):

$A \to BA'' \mid A'''$
$A'' \to z \mid y$
$A''' \to a \mid b$
$B \to AB'' \mid B'''$
$B'' \to w \mid u$
$B''' \to c \mid d$

4.1 Algorytm Paull'a

Po przeprowadzeniu przekształcenia gramatyki metodą Paull'a otrzymaliśmy następującą gramatykę:

$A \to Bz \mid By \mid a \mid b$
$B \to c \mid d \mid aw \mid bw \mid au \mid bu \mid awB' \mid bwB' \mid auB' \mid buB' \mid cB' \mid dB'$
$B' \to zw \mid zwB' \mid yw \mid ywB' \mid zu \mid zuB' \mid yu \mid yuB'$

Składa się ona z 24 produkcji i 77 symboli terminalnych i nieterminalnych przy pełnym zapisie.

4.2 Algorytm Rosenkrantz'a i Lewis'a

Gramatyka wygenerowana przez algorytm zawiera wiele symboli nieosiągalnych i ϵ-produkcji, składa się z 26 produkcji i 68 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle$
$B \to a\langle Ba\rangle \mid b\langle Bb\rangle \mid c\langle Bc\rangle \mid d\langle Bd\rangle$
$\langle AB\rangle \to z\langle AA\rangle \mid y\langle AA\rangle$

$\langle Aa\rangle \to \langle AA\rangle$
$\langle Ab\rangle \to \langle AA\rangle$
$\langle AA\rangle \to w\langle AB\rangle \mid u\langle AB\rangle \mid \epsilon$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$
$\langle BB\rangle \to z\langle BA\rangle \mid y\langle BA\rangle \mid \epsilon$
$\langle Ba\rangle \to \langle BA\rangle$
$\langle Bb\rangle \to \langle BA\rangle$
$\langle BA\rangle \to w\langle BB\rangle \mid u\langle BB\rangle$
$\langle Bc\rangle \to \langle BB\rangle$
$\langle Bd\rangle \to \langle BB\rangle$

Po usunięciu symboli nieosiągalnych otrzymujemy dokładnie o połowę mniejszą gramatykę, składającą się z 13 produkcji i 34 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle$
$\langle AB\rangle \to z\langle AA\rangle \mid y\langle AA\rangle$
$\langle Aa\rangle \to \langle AA\rangle$
$\langle Ab\rangle \to \langle AA\rangle$
$\langle AA\rangle \to w\langle AB\rangle \mid u\langle AB\rangle \mid \epsilon$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$

Po usunięciu ϵ-produkcji otrzymujemy nieco większą gramatykę, składającą się z 16 produkcji i 40 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle \mid a \mid b$
$\langle AB\rangle \to z\langle AA\rangle \mid y\langle AA\rangle \mid z \mid y$
$\langle Aa\rangle \to \langle AA\rangle$
$\langle Ab\rangle \to \langle AA\rangle$
$\langle AA\rangle \to w\langle AB\rangle \mid u\langle AB\rangle$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$

4.3 Algorytm Johnsona

Gramatyka wygenerowana przez algorytm zawiera wiele symboli nieosiągalnych i ϵ-produkcji, składa się z 32 produkcji i 82 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle$
$B \to a\langle Ba\rangle \mid b\langle Bb\rangle \mid c\langle Bc\rangle \mid d\langle Bd\rangle$
$\langle AB\rangle \to z\langle AA\rangle \mid y\langle AA\rangle \mid z \mid y$
$\langle Aa\rangle \to \langle AA\rangle \mid \epsilon$
$\langle Ab\rangle \to \langle AA\rangle \mid \epsilon$
$\langle AA\rangle \to w\langle AB\rangle \mid u\langle AB\rangle$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$

$\langle BB\rangle \to z\langle BA\rangle \mid y\langle BA\rangle$
$\langle Ba\rangle \to \langle BA\rangle$
$\langle Bb\rangle \to \langle BA\rangle$
$\langle BA\rangle \to w\langle BB\rangle \mid u\langle BB\rangle \mid w \mid u$
$\langle Bc\rangle \to \langle BB\rangle \mid \epsilon$
$\langle Bd\rangle \to \langle BB\rangle \mid \epsilon$

Po usunięciu symboli nieosiągalnych otrzymujemy zdecydowanie mniejszą gramatykę, składającą się z 13 produkcji i 34 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle$
$\langle AB\rangle \to z\langle AA\rangle \mid y\langle AA\rangle \mid z \mid y$
$\langle Aa\rangle \to \langle AA\rangle \mid \epsilon$
$\langle Ab\rangle \to \langle AA\rangle \mid \epsilon$
$\langle AA\rangle \to w\langle AB\rangle \mid u\langle AB\rangle$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$

Po usunięcie ϵ-produkcji otrzymujemy nową gramatykę, ale o tym samym rozmiarze, bo składającą się z 16 produkcji i 40 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle \mid a \mid b$
$\langle AB\rangle \to z\langle AA\rangle \mid y\langle AA\rangle \mid z \mid y$
$\langle Aa\rangle \to \langle AA\rangle$
$\langle Ab\rangle \to \langle AA\rangle$
$\langle AA\rangle \to w\langle AB\rangle \mid u\langle AB\rangle$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$

4.4 Algorytm LC_{LR}

nieterminal	czy spełnia warunki (*)
A	$\surd$
B	$-$

Algorytm przebiega w czterech krokach.

Krok 1.
$X \in T \cup N$ nie jest lewostronnie rekursywny,
$\langle AX\rangle$ spełnia warunki (*),
do nowej gramatyki dołącz produkcję $A \to X\langle AX\rangle$

Dla rozważanej gramatyki algorytm wygeneruje w tym kroku następujące produkcje:

$A \to a\langle Aa\rangle$
$A \to b\langle Ab\rangle$
$A \to c\langle Ac\rangle$
$A \to d\langle Ad\rangle$

Zauważmy, że oryginalny algorytm Johnsona dodałby jeszcze produkcje związane z nieterminalem B.

Krok 2.
$B \in N$ lewostronnie rekursywny,
$\langle AB \rangle$ spełnia warunki (*),
$B \to X\beta$ jest produkcją oryginalnej gramatyki,
dołącz produkcję $\langle AX \rangle \to \beta\langle AB \rangle$

Dla naszej gramatyki algorytm przebiegać będzie następująco:

A	B	$B \to X\beta$	dołączamy $\langle AX \rangle \to \beta\langle AB \rangle$
A	A	$A \to Bz$	$\langle AB \rangle \to z\langle AA \rangle$
		$A \to By$	$\langle AB \rangle \to y\langle AA \rangle$
		$A \to a$	$\langle Aa \rangle \to \langle AA \rangle$
		$A \to b$	$\langle Ab \rangle \to \langle AA \rangle$
	B	$B \to Aw$	$\langle AA \rangle \to w\langle AB \rangle$
		$B \to Au$	$\langle AA \rangle \to u\langle AB \rangle$
		$B \to c$	$\langle Ac \rangle \to \langle AB \rangle$
		$B \to d$	$\langle Ad \rangle \to \langle AB \rangle$

Krok 3.
$A \in N$ spełnia warunki (*),
$X \in N \cup T$,
$A \to X\beta$ jest produkcją oryginalnej gramatyki,
dołącz produkcję $\langle AX \rangle \to \beta$

Zatem w tym kroku zostaną dodane do gramatyki następujące produkcje:

$\langle AB \rangle \to z$
$\langle AB \rangle \to y$
$\langle Aa \rangle \to \epsilon$
$\langle Ab \rangle \to \epsilon$

Krok 4.
$A \in N$ nie jest lewostronnie rekursywny,
$A \to \beta$ jest produkcją oryginalnej gramatyki,
przepisz tą produkcję do nowej gramatyki.

W przypadku naszej gramatyki oba terminale, zarówno A jak i B są lewostronnie rekursywne, a więc w tym kroku algorytm nie doda żadnych produkcji.

Otrzymana gramatyka ma rozmiar 16 produkcji (40 symboli) i jest mniejsza niż ta uzyskana algorytmem Paull'a:

$A \to a\langle Aa \rangle \mid b\langle Ab \rangle \mid c\langle Ac \rangle \mid d\langle Ad \rangle$
$\langle AB \rangle \to z\langle AA \rangle \mid y\langle AA \rangle \mid z \mid y$
$\langle Aa \rangle \to \langle AA \rangle \mid \epsilon$

$\langle Ab\rangle \to \langle AA\rangle \mid \epsilon$
$\langle AA\rangle \to w\langle AB\rangle \mid u\langle AB\rangle$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$

Algorytm LC_{LR} ma tą własność, że nie generuje żadnych dodatkowych symboli nieosiągalnych. Zauważmy ciekawą rzecz, że identyczną gramatykę otrzymaliśmy powyżej po usunięciu symboli nieosiągalnych po zastosowaniu algorytmu Johnsona. Z powyższego powodu nie będziemy powtarzać usunięcia ϵ-produkcji z tej gramatyki. Chociaż algorytm LC_{LR} jest pewną wariacją algorytmu Johnsona, zaistniała sytuacja nie jest regułą, o czym przekonamy się na kolejnych przykładach.

4.5 LF + algorytm Rosenkrantz'a i Lewis'a

Stosując algorytm Rosenkrantz'a i Lewis'a po uprzednim zastosowaniu LF otrzymujemy następującą gramatykę o 32 produkcjach o 80 symbolach. Zawiera ona wiele symboli nieużytecznych i ϵ-produkcji.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle$
$B \to a\langle Ba\rangle \mid b\langle Bb\rangle \mid c\langle Bc\rangle \mid d\langle Bd\rangle$
$A'' \to z\langle A''z\rangle \mid y\langle A''y\rangle$
$B'' \to w\langle B''w\rangle \mid u\langle B''u\rangle$
$\langle AB\rangle \to A''\langle AA\rangle$
$\langle Aa\rangle \to \langle AA\rangle$
$\langle Ab\rangle \to \langle AA\rangle$
$\langle AA\rangle \to B''\langle AB\rangle \mid \epsilon$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$
$\langle BB\rangle \to A''\langle BA\rangle$
$\langle Ba\rangle \to \langle BA\rangle$
$\langle Bb\rangle \to \langle BA\rangle$
$\langle BA\rangle \to B''\langle BB\rangle \mid \epsilon$
$\langle Bc\rangle \to \langle BB\rangle$
$\langle Bd\rangle \to \langle BB\rangle$
$\langle A''z\rangle \to \langle A''A''\rangle$
$\langle A''y\rangle \to \langle A''A''\rangle$
$\langle B''w\rangle \to \langle B''B''\rangle$
$\langle B''u\rangle \to \langle B''B''\rangle$
$\langle A''A''\rangle \to \epsilon$
$\langle B''B''\rangle \to \epsilon$

Po usunięciu symboli nieosiągalnych otrzymujemy nową gramatykę składającą się z 21 produkcji i 52 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle$
$A'' \to z\langle A''z\rangle \mid y\langle A''y\rangle$
$B'' \to w\langle B''w\rangle \mid u\langle B''u\rangle$

$\langle AB\rangle \to A''\langle AA\rangle$
$\langle Aa\rangle \to \langle AA\rangle$
$\langle Ab\rangle \to \langle AA\rangle$
$\langle AA\rangle \to B''\langle AB\rangle \mid \epsilon$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$
$\langle A''z\rangle \to \langle A''A''\rangle$
$\langle A''y\rangle \to \langle A''A''\rangle$
$\langle B''w\rangle \to \langle B''B''\rangle$
$\langle B''u\rangle \to \langle B''B''\rangle$
$\langle A''A''\rangle \to \epsilon$
$\langle B''B''\rangle \to \epsilon$

Po usunięciu ϵ-produkcji otrzymujemy gramatykę składającą się z 17 produkcji i 40 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle \mid a \mid b$
$A'' \to z \mid y$
$B'' \to w \mid u$
$\langle AB\rangle \to A''\langle AA\rangle \mid A''$
$\langle Aa\rangle \to \langle AA\rangle$
$\langle Ab\rangle \to \langle AA\rangle$
$\langle AA\rangle \to B''\langle AB\rangle$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$

4.6 LF + NLRG + algorytm Rosenkrantz'a i Lewis'a

Po zastosowaniu kolejno algorytmów *LF, NLRG* oraz Rosenkrantz'a i Lewisa, otrzymaliśmy poniższą gramatykę, składającą się z 46 produkcji i 113 symboli terminalnych i nieterminalnych w pełnym zapisie.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle$
$B \to a\langle Ba\rangle \mid b\langle Bb\rangle \mid c\langle Bc\rangle \mid d\langle Bd\rangle$
$A'' \to z\langle A''z\rangle \mid y\langle A''y\rangle$
$B'' \to w\langle B''w\rangle \mid u\langle B''u\rangle$
$A''' \to a\langle A'''a\rangle \mid b\langle A'''b\rangle$
$B''' \to c\langle B'''c\rangle \mid d\langle B'''d\rangle$
$\langle AB\rangle \to A''\langle AA\rangle$
$\langle AA'''\rangle \to \langle AA\rangle$
$\langle AA\rangle \to B''\langle AB\rangle \mid \epsilon$
$\langle AB'''\rangle \to AB\rangle$
$\langle Aa\rangle \to \langle AA'''\rangle$
$\langle Ab\rangle \to \langle AA'''\rangle$
$\langle Ac\rangle \to \langle AB'''\rangle$
$\langle Ad\rangle \to \langle AB'''\rangle$
$\langle A''z\rangle \to \langle A''A''\rangle$
$\langle A''y\rangle \to \langle A''A''\rangle$

$\langle A'''a\rangle \to \langle A'''A'''\rangle$
$\langle A'''b\rangle \to \langle A'''A'''\rangle$
$\langle BB\rangle \to A''\langle BA\rangle \mid \epsilon$
$\langle BA'''\rangle \to \langle BA\rangle$
$\langle BA\rangle \to B''\langle BB\rangle$
$\langle BB'''\rangle \to \langle BB\rangle$
$\langle Ba\rangle \to \langle BA'''\rangle$
$\langle Bb\rangle \to \langle BA'''\rangle$
$\langle Bc\rangle \to \langle BB'''\rangle$
$\langle Bd\rangle \to \langle BB'''\rangle$
$\langle B''w\rangle \to \langle B''B''\rangle$
$\langle B''u\rangle \to \langle B''B''\rangle$
$\langle B'''c\rangle \to \langle B'''B'''\rangle$
$\langle B'''d\rangle \to \langle B'''B'''\rangle$
$\langle A''A''\rangle \to \epsilon$
$\langle B''B''\rangle \to \epsilon$
$\langle A'''A'''\rangle \to \epsilon$
$\langle B'''B'''\rangle \to \epsilon$

Po usunięciu symboli nieosiągalnych otrzymujemy o połowę mniejszą gramatykę, składającą się z 23 produkcji i 56 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle$
$A'' \to z\langle A''z\rangle \mid y\langle A''y\rangle$
$B'' \to w\langle B''w\rangle \mid u\langle B''u\rangle$
$\langle AB\rangle \to A''\langle AA\rangle$
$\langle AA'''\rangle \to \langle AA\rangle$
$\langle AA\rangle \to B''\langle AB\rangle \mid \epsilon$
$\langle AB'''\rangle \to \langle AB\rangle$
$\langle Aa\rangle \to \langle AA'''\rangle$
$\langle Ab\rangle \to \langle AA'''\rangle$
$\langle Ac\rangle \to \langle AB'''\rangle$
$\langle Ad\rangle \to \langle AB'''\rangle$
$\langle A''z\rangle \to \langle A''A''\rangle$
$\langle A''y\rangle \to \langle A''A''\rangle$
$\langle B''w\rangle \to \langle B''B''\rangle$
$\langle B''u\rangle \to \langle B''B''\rangle$
$\langle A''A''\rangle \to \epsilon$
$\langle B''B''\rangle \to \epsilon$

Po usunięciu ϵ-produkcji otrzymujemy jeszcze mniejszą gramatykę składającą się z 19 produkcji i 44 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle \mid a \mid b$
$A'' \to z \mid y$
$B'' \to w \mid u$
$\langle AB\rangle \to A''\langle AA\rangle \mid A''$

$\langle AA''' \rangle \to \langle AA \rangle$
$\langle AA \rangle \to B''\langle AB \rangle$
$\langle AB''' \rangle \to \langle AB \rangle$
$\langle Aa \rangle \to \langle AA''' \rangle$
$\langle Ab \rangle \to \langle AA''' \rangle$
$\langle Ac \rangle \to \langle AB''' \rangle$
$\langle Ad \rangle \to \langle AB''' \rangle$

4.7 LF + algorytm Johnsona

Stosując algorytm Johnsona po uprzednim zastosowaniu LF otrzymujemy następującą gramatykę o 34 produkcjach o 84 symbolach:

$A \to a\langle Aa \rangle \mid b\langle Ab \rangle \mid c\langle Ac \rangle \mid d\langle Ad \rangle$
$B \to a\langle Ba \rangle \mid b\langle Bb \rangle \mid c\langle Bc \rangle \mid d\langle Bd \rangle$
$A'' \to z\langle A''z \rangle \mid y\langle A''y \rangle$
$B'' \to w\langle B''w \rangle \mid u\langle B''u \rangle$
$\langle AB \rangle \to A''\langle AA \rangle \mid A''$
$\langle Aa \rangle \to \langle AA \rangle \mid \epsilon$
$\langle Ab \rangle \to \langle AA \rangle \mid \epsilon$
$\langle AA \rangle \to B''\langle AB \rangle$
$\langle Ac \rangle \to \langle AB \rangle$
$\langle Ad \rangle \to \langle AB \rangle$
$\langle BB \rangle \to A''\langle BA \rangle$
$\langle Ba \rangle \to \langle BA \rangle$
$\langle Bb \rangle \to \langle BA \rangle$
$\langle BA \rangle \to B''\langle BB \rangle \mid B''$
$\langle Bc \rangle \to \langle BB \rangle \mid \epsilon$
$\langle Bd \rangle \to \langle BB \rangle \mid \epsilon$
$\langle A''z \rangle \to \epsilon$
$\langle A''y \rangle \to \epsilon$
$\langle B''w \rangle \to \epsilon$
$\langle B''u \rangle \to \epsilon$

Po usunięciu symboli nieosiągalnych otrzymujemy mniejszą gramatykę, składającą się z 21 produkcji i 51 symboli.

$A \to a\langle Aa \rangle \mid b\langle Ab \rangle \mid c\langle Ac \rangle \mid d\langle Ad \rangle$
$A'' \to z\langle A''z \rangle \mid y\langle A''y \rangle$
$B'' \to w\langle B''w \rangle \mid u\langle B''u \rangle$
$\langle AB \rangle \to A''\langle AA \rangle \mid A''$
$\langle Aa \rangle \to \langle AA \rangle \mid \epsilon$
$\langle Ab \rangle \to \langle AA \rangle \mid \epsilon$
$\langle AA \rangle \to B''\langle AB \rangle$
$\langle Ac \rangle \to \langle AB \rangle$
$\langle Ad \rangle \to \langle AB \rangle$
$\langle A''z \rangle \to \epsilon$

$\langle A''y\rangle \to \epsilon$
$\langle B''w\rangle \to \epsilon$
$\langle B''u\rangle \to \epsilon$

Po usunięciu ϵ-produkcji otrzymujemy jeszcze mniejszą gramatykę składającą się z 17 produkcji i 40 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle \mid a \mid b$
$A'' \to z \mid y$
$B'' \to w \mid u$
$\langle AB\rangle \to A''\langle AA\rangle \mid A''$
$\langle Aa\rangle \to \langle AA\rangle$
$\langle Ab\rangle \to \langle AA\rangle$
$\langle AA\rangle \to B''\langle AB\rangle$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$

Warto zauważyć, że jest to identyczna gramatyka, jak uzyskana w wyniku zastosowania algorytmu Rosenkrantz'a i Lewis'a.

4.8 LF + NLRG + algorytm Johnsona

Otrzymana w wyniku połączenia przekształceń *LF, NLRG* i algorytmu Johnsona gramatyka miała rozmiar 44 produkcji i 108 symboli:

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle$
$B \to a\langle Ba\rangle \mid b\langle Bb\rangle \mid c\langle Bc\rangle \mid d\langle Bd\rangle$
$A'' \to z\langle A''z\rangle \mid y\langle A''y\rangle$
$B'' \to w\langle B''w\rangle \mid u\langle B''u\rangle$
$A''' \to a\langle A'''a\rangle \mid b\langle A'''b\rangle$
$B''' \to c\langle B'''c\rangle \mid d\langle B'''d\rangle$
$\langle AB\rangle \to A''\langle AA\rangle \mid A''$
$\langle AA'''\rangle \to \langle AA\rangle \mid \epsilon$
$\langle AA\rangle \to B''\langle AB\rangle$
$\langle AB'''\rangle \to \langle AB\rangle$
$\langle Aa\rangle \to \langle AA'''\rangle$
$\langle Ab\rangle \to \langle AA'''\rangle$
$\langle Ac\rangle \to \langle AB'''\rangle$
$\langle Ad\rangle \to \langle AB'''\rangle$
$\langle BB\rangle \to A''\langle BA\rangle$
$\langle BA'''\rangle \to \langle BA\rangle$
$\langle BA\rangle \to B''\langle BB\rangle \mid B''$
$\langle BB'''\rangle \to \langle BB\rangle \mid \epsilon$
$\langle Ba\rangle \to \langle BA'''\rangle$
$\langle Bb\rangle \to \langle BA'''\rangle$
$\langle Bc\rangle \to \langle BB'''\rangle$
$\langle Bd\rangle \to \langle BB'''\rangle$

$\langle A''z\rangle \to \epsilon$
$\langle A''y\rangle \to \epsilon$
$\langle B''w\rangle \to \epsilon$
$\langle B''u\rangle \to \epsilon$
$\langle A'''a\rangle \to \epsilon$
$\langle A'''b\rangle \to \epsilon$
$\langle B'''c\rangle \to \epsilon$
$\langle B'''d\rangle \to \epsilon$

Po usunięciu symboli nieosiągalnych otrzymujemy i w tym przypadku dwukrotnie zmniejszenie gramatyki do 22 produkcji i 54 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle$
$A'' \to z\langle A''z\rangle \mid y\langle A''y\rangle$
$B'' \to w\langle B''w\rangle \mid u\langle B''u\rangle$
$\langle AB\rangle \to A''\langle AA\rangle \mid A''$
$\langle AA'''\rangle \to \langle AA\rangle \mid \epsilon$
$\langle AA\rangle \to B''\langle AB\rangle$
$\langle AB'''\rangle \to \langle AB\rangle$
$\langle Aa\rangle \to \langle AA'''\rangle$
$\langle Ab\rangle \to \langle AA'''\rangle$
$\langle Ac\rangle \to \langle AB'''\rangle$
$\langle Ad\rangle \to \langle AB'''\rangle$
$\langle A''z\rangle \to \epsilon$
$\langle A''y\rangle \to \epsilon$
$\langle B''w\rangle \to \epsilon$
$\langle B''u\rangle \to \epsilon$

Po usunięciu ϵ-produkcji otrzymujemy jeszcze mniejszą gramatykę składającą się z 19 produkcji i 44 symboli.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle \mid a \mid b$
$A'' \to z \mid y$
$B'' \to w \mid u$
$\langle AB\rangle \to A''\langle AA\rangle \mid A''$
$\langle AA'''\rangle \to \langle AA\rangle$
$\langle AA\rangle \to B''\langle AB\rangle$
$\langle AB'''\rangle \to \langle AB\rangle$
$\langle Aa\rangle \to \langle AA'''\rangle$
$\langle Ab\rangle \to \langle AA'''\rangle$
$\langle Ac\rangle \to \langle AB'''\rangle$
$\langle Ad\rangle \to \langle AB'''\rangle$

Otrzymaliśmy identyczną gramatykę jak po usunięciu symboli nieosiągalnych i ϵ-produkcji z gramatyki wygenerowanej przez połączenie przekształceń LF + $NLRG$ + algorytm Rosenkrantz'a i Lewis'a.

4.9 LF + algorytm LC_{LR}

Otrzymana w wyniku połączenia przekształceń *LF*, *NLRG* i algorytmu Johnsona gramatyka miała rozmiar 17 produkcji i potrzebowała do pełnego zapisu 40 symboli terminalnych i nieterminalnych. Lewostronna faktoryzacja nieznacznie powiększyła rozmiar generowanej gramatyki w przypadku algorytmu LC_{LR}. Zapis gramatyki przedstawiony jest poniżej.

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle$
$\langle AB\rangle \to A''\langle AA\rangle \mid A''$
$\langle Aa\rangle \to \langle AA\rangle \mid \epsilon$
$\langle Ab\rangle \to \langle AA\rangle \mid \epsilon$
$\langle AA\rangle \to B''\langle AB\rangle$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$
$A'' \to z \mid y$
$B'' \to w \mid u$

Gramatyka nie posiada symboli nieosiągalnych. Po usunięciu ϵ-produkcji otrzymujemy trochę inną gramatykę o tym samym rozmiarze 17 (40).

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle \mid a \mid b$
$A'' \to z \mid y$
$B'' \to w \mid u$
$\langle AB\rangle \to A''\langle AA\rangle \mid A''$
$\langle Aa\rangle \to \langle AA\rangle$
$\langle Ab\rangle \to \langle AA\rangle$
$\langle AA\rangle \to B''\langle AB\rangle$
$\langle Ac\rangle \to \langle AB\rangle$
$\langle Ad\rangle \to \langle AB\rangle$

Jest to identyczna gramatyka, jak uzyskana po algorytmie Rosenkrantz'a i Lewis'a i lewostronnej faktoryzacji oraz Johnsona i lewostronnej faktoryzacji. W tym przypadku wszystkie algorytmy lewego rogu prowadzą w różny sposób do tego samego rezultatu.

4.10 LF + NLRG + algorytm LC_{LR}

Gramatyka wygenerowana przez algorytm ma rozmiar 20 produkcji (48 symboli) i wygląda następująco:

$A \to a\langle Aa\rangle \mid b\langle Ab\rangle \mid c\langle Ac\rangle \mid d\langle Ad\rangle \mid A'''\langle AA'''\rangle \mid B'''\langle AB'''\rangle$
$\langle AB\rangle \to A''\langle AA\rangle \mid A''$
$\langle AA'''\rangle \to \langle AA\rangle \mid \epsilon$
$\langle AA\rangle \to B''\langle AB\rangle$
$\langle AB'''\rangle \to \langle AB\rangle$
$A'' \to z \mid y$
$B'' \to w \mid u$
$A''' \to a \mid b$

$B''' \to c \mid d$

Po usunięciu symboli nieużytecznych otrzymujemy gramatykę zawierającą 16 produkcji i 36 symboli:

$A \to A'''\langle AA'''\rangle \mid B'''\langle AB'''\rangle$
$\langle AB\rangle \to A''\langle AA\rangle \mid A''$
$\langle AA'''\rangle \to \langle AA\rangle \mid \epsilon$
$\langle AA\rangle \to B''\langle AB\rangle$
$\langle AB'''\rangle \to \langle AB\rangle$
$A'' \to z \mid y$
$B'' \to w \mid u$
$A''' \to a \mid b$
$B''' \to c \mid d$

Gramatyka nie posiada symboli nieosiągalnych. Po usunięciu ϵ-produkcji otrzymujemy trochę inną gramatykę o tym samym rozmiarze 16 (36).

$A \to A'''\langle AA'''\rangle \mid B'''\langle AB'''\rangle \mid A'''$
$\langle AB\rangle \to A''\langle AA\rangle \mid A''$
$\langle AA'''\rangle \to \langle AA\rangle$
$\langle AA\rangle \to B''\langle AB\rangle$
$\langle AB'''\rangle \to \langle AB\rangle$
$A'' \to z \mid y$
$B'' \to w \mid u$
$A''' \to a \mid b$
$B''' \to c \mid d$

Dodanie do LF przekształcenia $NLRG$ sprawia, że tylko podstawowe dwie wersje algorytmu lewego rogu po usunięciu symboli nieosiągalnych i ϵ-produkcji prowadzą do tego samego rezultatu.

4.11 Podsumowanie

Podsumowanie rozmiarów uzyskanych gramatyk przedstawia zbiorczo poniższa tabela. Dla skrócenia zapisu oznaczono $NLRG$ jako G, usuwanie symboli nieosiągalnych jako N, zaś usuwanie ϵ-produkcji jako ϵ.
Dla odróżnienia gramatyk identycznych od gramatyk o tym samym rozmiarze, zastosowano oznaczenia:
"-" w wierszu, gdy dane przekształcenie nic nie wnosi,
"$-||-$" w kolumnie, gdy różne algorytmy lewego rogu dały te same rezultaty.

przed po	- -	- N	- N+ϵ	LF -	LF N	LF N+ϵ	LF+G -	LF+G N	LF+G N+ϵ
-	8(20)	-	-	10(22)	-	-	12(26)	-	-
Paull	24(77)	-	-	17(46)	-	-	16(40)	-	-
R. & L.	26 (68)	13 (34)	16 (40)	32 (80)	21 (52)	17 (40)	46 (113)	23 (56)	19 (44)
Johnson	32 (82)	16 (40)	$-\|\|-$	34 (84)	21 (51)	$-\|\|-$	44 (108)	22 (54)	$-\|\|-$
LC_{LR}	16 (40)	- $-\|\|-$	$-\|\|-$	17 (40)	-	$-\|\|-$	20 (48)	16 (36)	16 (36)

Widać, że żadne przekształcenie nie jest uniwersalne. *LF* daje bardzo dużą poprawę dla algorytmu Paull'a zastosowanego do naszej gramatyki, zaś *NLRG* już tylko nieznaczną. Najlepszy rezultat dał algorytm LC_{LR}.
Nieprawdziwa okazała się hipoteza, jakoby algorytm Johnsona po usunięciu symboli nieosiągalnych miał dawać te same rezultaty co LC_{LR}, już po zastosowaniu *LF* nasza gramatyka przestała wykazywać tę własność.
Początkowo wydawało się, że wszystkie algorytmy lewego rogu po usunięciu symboli nieosiągalnych i ϵ-produkcji sprowadzają się do tego samego. Jednak po zastosowaniu *NLRG* otrzymaliśmy gramatykę, która zaprzecza tej hipotezie, ponieważ LC_{LR} dał inny rezultat. Natomiast nadal pozostaje przypuszczenie, że algorytmy Rosenkrantz'a i Lewis'a oraz Johnsona sprowadzają się do tego samego.

Literatura

[1] Johnson M., *Finite-state Approximation of Constraint-based Grammars using Left-corner Grammar Transforms*, Cognitive and Linguistic Sciences, Brown University

[2] Moore R. C., *Removing Left Recursion from Context-Free Grammars*, Microsoft Research, One Microsoft Way.

[3] Moore R. C., *Improved Lef-Corner Chart Parsing for Large Context-Free Grammars*, Microsoft Research, One Microsoft Way.

[4] Rosenkrantz D. J. and Lewis P. M., *Deterministic Left-Corner Parsing*, General Electric Research and Development Center, New York.

[5] Johnson M., *Left-Corner Transforms and Finite-state Approximations*, Rank Xerox Research Centre, Grenoble.