Uniwersytet Zielonogórski
Instytut Sterowania i Systemów Informatycznych

Teoretyczne Podstawy Informatyki

Lista 5 – Gramatyki bezkontekstowe i automaty ze stosem

1 Wprowadzenie

1.1 Gramatyka bezkontekstowa

Gramatykę bezkontekstową nazywamy skończony zbiór zmiennych nazywanych symbolami niekońcowymi bądź symbolami pomocniczymi lub kategoriami syntaktycznymi. To ostatnie określenie jest stosowane głównie w przypadku, jeśli gramatyka bezkontekstowa opisuje język naturalny. Tego typu gramatyki mają ogromne znaczenie praktyczne w informatyce. Z ich pomocą można opisywać w sposób formalny i ścisły języki programowania. Znakomitymi przykładami są wyrażenia arytmetyczne, bądź bloki wyznaczane przez np.: słowa kluczowe begin oraz end. Co ważne do opisu tego typu wyrażeń nie można stosować wyrażeń regularnych. Jest tak dlatego iż języki opisane przez gramatykę bezkontekstową są opisywane przez wzajemną rekursję z zastosowaniem elementów pierwotnych które nazwane zostały symbolami końcowymi.

Przykładem opisu bezkontekstowego jest poniższy zestaw produkcji (reguł pomiędzy zmiennymi i symbolami terminalnymi) który opisuje pewien model zdania:

$$\begin{aligned}
< \text{zdanie} > &\rightarrow < \text{fraza rzeczownikowa} >< \text{fraza czasownikowa} > \\
< \text{fraza czasownikowa} > &\rightarrow < \text{przymiotnik} >< \text{fraza rzeczownikowa} > \\
< \text{fraza rzeczownikowa} > &\rightarrow < \text{rzeczownik} > \\
< \text{rzeczownik} > &\rightarrow \text{chłopiec} \\
< \text{przymiotnik} > &\rightarrow \text{mały}
\end{aligned}$$

Symbolami końcowymi są dwa słowa „chłopiec" oraz „mały". Natomiast zmienne ujęte w nawiasy ostre to kategorie semantyczne, inna nazwa to symbole nieterminalne.

Innym przykładem jest opis wyrażeń arytmetycznych zbudowanych z podstawowych operatorów arytmetycznych oraz nawiasów zwykłych. Gramatyka bezkontekstowa może przyjąć następującą postać:

$$\begin{aligned}
< \text{expr} > &\rightarrow < \text{expr} > + < \text{expr} > \\
< \text{expr} > &\rightarrow < \text{expr} > - < \text{expr} > \\
< \text{expr} > &\rightarrow < \text{expr} > * < \text{expr} > \\
< \text{expr} > &\rightarrow < \text{expr} > / < \text{expr} > \\
< \text{expr} > &\rightarrow (< \text{expr} >) \\
< \text{expr} > &\rightarrow \textbf{num}
\end{aligned}$$

Zmienna $< expr >$ reprezentuje dowolne wyrażenie złożone ze znaków terminalnych $+, -, *, /, (,)$ oraz liczb (załóżmy, że całkowitych) które są reprezentowane przez zmienną **num**. Naturalnie każda liczba jest symbolem terminalnym choć postać liczby opisujemy za pomocą gramatyki regularnej (wyrażeń regularnych).

Dla przykładu rozważmy wyrażenie $(2+5)*5$. Jego zapis gramatyczny jest następujący: (**num** + **num**) * **num**, zostaje ono wyprowadzone w następujący sposób:

$$\begin{aligned}
< \text{expr} > &\rightarrow < \text{expr} > * < \text{expr} > \\
&\rightarrow (< \text{expr} >) * < \text{expr} > \\
&\rightarrow (< \text{expr} >) * \textbf{num} \\
&\rightarrow (< \text{expr} > + < \text{expr} >) * \textbf{num} \\
&\rightarrow (< \text{expr} > + \textbf{num}) * \textbf{num} \\
&\rightarrow (\textbf{num} + \textbf{num}) * \textbf{num}
\end{aligned}$$

1.2 Formalna definicja i postać normalna Chomsky'ego

Formalnie gramatykę G zapiszemy jako następującą czwórkę: $G = (V, T, P, S)$. Gdzie V i T są odpowiednio skończonymi zbiorami zmiennych i symboli końcowych. Dodatkowo zakładamy, że V i T są rozłączne. P jest skończonym zbiorem produkcji. Produkcje mają postać $A \rightarrow \alpha$, gdzie A jest zmienną, a α to łańcuch symboli z $(V \cup T)^*$. Zakładamy, także że S jest zmienną specjalną, zwaną symbolem początkowym.

Twierdzenie 1 *(Postać normalna Chomsky'ego)*
Dowolny język bezkontekstowy nie zawierający λ jest generowany przez gramatykę, której wszystkie produkcje są postaci $A \to BC$ lub $A \to a$; A, B i C są tu zmiennymi, natomiast a jest symbolem końcowym.

Przykład 1 *Mamy gramatykę $\{\{S, A, B\}, \{a, b\}, P, S\}$, która posiada następujące produkcje:*

$$\begin{array}{lcllll} S & \to & bA & | & aB & \\ A & \to & aAA & | & aS & | \quad a \\ B & \to & aBB & | & bS & | \quad b \end{array}$$

Produkcje postaci $A \to a$ oraz $B \to b$ są już we właściwej postaci. Produkcję $S \to aA$ zastąpimy produkcjami $S \to C_bA$ oraz $C_b \to b$. Analogicznie dla $A \to aS$ wprowadzamy dwie nowe produkcje $S \to C_aS$ oraz $C_a \to a$. Postępując podobnie dla pozostałych reguł ostatecznie otrzymamy następującą gramatykę w postaci normalnej Chomsky'ego:

$$\begin{array}{lcllll} S & \to & C_bA & | & C_aB & \\ A & \to & C_aS & | & C_bD_1 & | \quad a \\ B & \to & C_bS & | & C_aD_2 & | \quad b \\ D_1 & \to & AA & & & \\ D_2 & \to & BB & & & \\ C_a & \to & a & & & \\ C_b & \to & b & & & \end{array}$$

1.2.1 Znajdowanie symboli pustych

Jeśli $G = (V, T, P, S)$ jest gramatyką bezkontekstową zawierającą symbole puste λ, to wszystkie symbole zerowalne, czyli takie, które wyniku zastosowania reguł gramatycznych doprawdzają do symbolu λ, można wyznaczyć za pomocą następującego algorytmu indukcyjnego:
Podstawa: Jeśli $A \to \lambda$ przynależy do G, to A jest zerowalna.
Krok indukcyjny: Jeśli, w produkcji $B \to C_1C_2C_3 \ldots C_k$, każde C_i jest zerowalne to także B jest zerowalne.

Stosowanie tego prostego algorytmu powoduje wykrywanie, które zmienne są zerowalne. Jeśli pojawiają się takie zmienne to rozbijają one daną produkcję na części, w zależności od ilości zmiennych zerowalnych.

1.3 Notacja BNF

Podstawowy zapis gramatyki w notacji BNF jest następujący:

$$< \text{symbol} > ::= < \text{inne wyrażenia} >$$

Symbole ujęte w nawiasy ostre reprezentują symbole nieterminalne, czyli zmienne leksykograficzne. Wyrażenie po znaku przypisania ::= może zawierać sekwencję innych symboli zawierających zarówno symbole nieterminalne oraz terminalne. Do każdej zmiennej można przypisać kilka wyrażeń rozdzielonych za pomocą pionowej kreski |. Symbol ten reprezentuje możliwość wyboru innej reguły gramatycznej przypisanej do danej zmiennej. Również, co jest bardzo istotne symbole terminalne występują tylko po prawej stronie znaku przypisania.

Świetnym przykładem zastosowania notacji BNF jest np.: poniższy opis gramatyki adresu domu/mieszkania jaki jest stosowany w Stanach Zjednoczonych:

$$\begin{array}{lcl} <\text{postal} - \text{addr}> & ::= & < \text{name} - \text{part} >< \text{street} - \text{addr} >< \text{zip} - \text{part} > \\ <\text{personal} - \text{part}> & ::= & < \text{first} - \text{name} > | < \text{initial} > \text{"."} \\ <\text{name} - \text{part}> & ::= & < \text{personal} - \text{part} >< \text{last} - \text{name} > [< \text{jr} - \text{part} >] < \text{EOL} > | \\ & & < \text{personal} - \text{part} >< \text{name} - \text{part} > \\ <\text{street} - \text{addr}> & ::= & [< \text{apt} >] < \text{house} - \text{num} >< \text{street} - \text{name} >< EOL > \\ <\text{zip} - \text{part}> & ::= & < \text{town} - \text{name} > \text{","} < \text{state} - \text{code} >< \text{ZIP} - \text{code} >< \text{EOL} > \end{array}$$

Gramatykę tego rodzaju odczytujemy w następujący sposób: dowolny adres postal − addr składa się z dwóch części nazwy, ulicy, kodu pocztowego. Następnie należy zanalizować zmienną name − part której elementami jest część personalna, nazwisko, w nawiasach kwadratowych umieszczona została dodatkowy przydomek. Natomiast symbol $< \text{EOL} >$ oznacza znak końca linii. Elementy ujęte w cudzysłowy także są traktowane jako symbole terminalne.

Zasady notacji BNF może zapisać za pomocą samej notacji BNF, co przedstawia się następująco:

$$
\begin{array}{lll}
<\text{syntax}> & ::= & <\text{rule}> [<\text{syntax}>] \\
<\text{rule}> & ::= & <\text{whitespace}> \text{" < "} <\text{rule}-\text{name}> \text{" > "} <\text{whitespace}> \text{" ::= "} \\
 & & <\text{expr}><\text{whitespace}><\text{line}-\text{end}> \\
<\text{expr}> & ::= & <\text{whitespace}><\text{or}-\text{expr}> \\
<\text{or}-\text{expr}> & ::= & <\text{whitespace}><\text{list}-\text{expr}> [\text{"|"} <\text{or}-\text{expr}>] \\
<\text{list}-\text{expr}> & ::= & <\text{whitespace}> (\text{" < "} <\text{rule}-\text{name}> \text{" > "} \\
 & & | <\text{QUOTE}><\text{text}><\text{QUOTE}> \\
 & & |\text{"("} <\text{expr}> \text{")"}|\text{"["} <\text{expr}> \text{"]"})[<\text{list}-\text{expr}>] \\
<\text{whitespace}> & ::= & [\text{""} <\text{whitespace}>] \\
<\text{line}-\text{end}> & ::= & [<\text{whitespace}>] <\text{EOL}> [<\text{line}-\text{end}>]
\end{array}
$$

Istnieją także inne warianty zapisu notacji BNF, gdzie stosowane są oznaczenia krotności $*$ oraz $+$. Jednakże, zaprezentowana powyżej notacja jest najbardziej spotykana i bezpośrednio odnosi się do gramatyki bezkontekstowej.

1.4 Drzewa wyprowadzeń

Drzewa wyprowadzeń to inna reprezentacja wyrażeń opisywanych przez gramatykę. Drzewa są stosowane w kompilatorach języków programowania do tworzenia reprezentacji programu źródłowego. Struktura drzewa pomaga w analizie wyrażenia, ponieważ wskazuje zależności pomiędzy poszczególnymi wyrażeniami. Ułatwia także translację programu, ponieważ łatwo zbudować analizator wyrażeń zapisanych w postaci drzew za pomocą funkcji rekurencyjnych.

Ogólnie dla danej gramatyki G może istnieć wiele drzew wyprowadzeń. Tego typu gramatyki naturalnie nie mogą reprezentować programów komputerowych. Drzewa wyprowadzeń mogą zatem służyć do porównania dwóch potencjalnie różnych programów komputerowych.

Dla ustalonej gramatyki $G = (V, T, P, S)$ drzewo wyprowadzenia lub drzewo rozkładu G, to drzewa spełniające następujące warunki:

1. każdy wierzchołek wewnętrzny jest etykietowany zmienną z V,
2. każdy liść etykietowany jest zmienną, symbolem końcowym lub ε. W przypadku etykietowania przez symbol ε, musi on być jedynym dzieckiem swojego rodzica.
3. Jeśli wierzchołek wewnętrzny etykietowany jest przez zmianną ze zbioru A, a wierzchołki potomne są etykietowany, od lewej do prawej symbolami

   $$X_1, X_2, \ldots, X_k$$

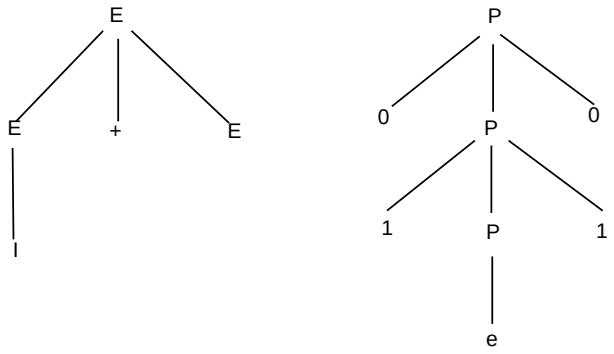
   to $A \to X_1 X_2 \ldots X_k$ jest produkcją z P. Zauważamy, że jedyny przypadek, gdy jedno z X_i może być równe ε, to taki, gdy jest ono etykietą jedynego dziecka, a $A \to \varepsilon$ jest produkcją G.

Przykładowe dwa drzewa prezentuje rysunek 1. Lewe drzewa przedstawiaja wyrażenie oparte o prostą gramatykę dla wyrażeń arytmetycznych. Korzeń jest etykietowany zmienną E. Zastosowana produkcja to $E \to E + E$, ponieważ troje dzieci korzenia mają, od lewej do prawej, etykiety E, $+$, E. Przy czym, przy najbardziej lewym zewnętrznym liściu pojawia się identyfikator I. Drugi przykład drzewa, to wyprowadzenie dla gramatyki generującej palindromy parzyste.

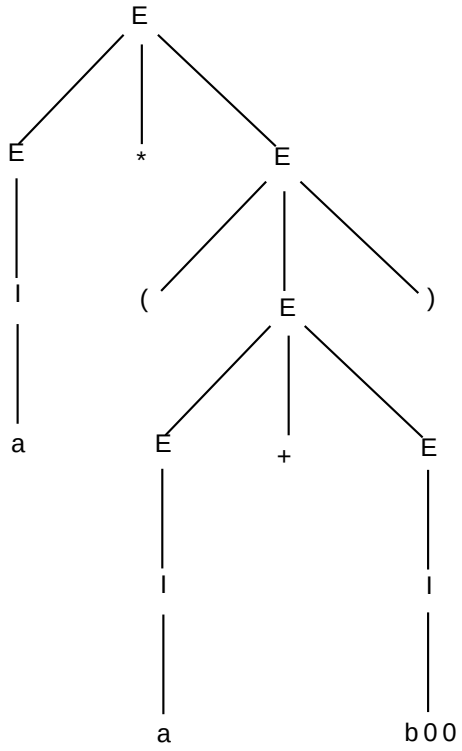
Rysunek 2 pokazuje bardziej skomplikowane drzewo oparte o gramatykę o następującej postaci:

$$
\begin{array}{lll}
E & \to & I \\
E & \to & E + E \\
E & \to & E * E \\
E & \to & (E) \\
I & \to & a \\
I & \to & b \\
I & \to & Ia \\
I & \to & Ib \\
I & \to & I0 \\
I & \to & I1
\end{array}
$$

Korzeniem w tym drzewie jest produkcja E. W przykładzie zostało wprowadzone pewne uproszczenie, bowiem wyrażenie $b00$ nie zostało rozwinięte za pomocą produkcji I z gramatyki podanej powyżej.



Rysunek 1: Dwa drzewa wyprowadzeń dla wyrażenia $I + E$ dla gramatyki wyrażeń arytmetycznych oraz 0110 dla gramatyki generującej palindromy parzyste



Rysunek 2: Drzewa wyprowadzenia oparte o gramatykę wyrażeń arytmetycznych dla wyrażenia $a * (a + b00)$

2 Automaty ze stosem

Automat ze stosem to niedeterministyczny automat z przejściami typu ε oraz dodatkowym stosem na którym można przechowywać symbole, określane także jak symbole stosowe. Stos pozwala na przechowywanie dowolnej ilości informacji, jednakże dostęp do informacji znajdującej się na stosie jest typowy dla tej struktury danych. Ze stosu można pobierać informacje tylko ze szczytu stosu. Automaty stosowe są ściśle związane z gramatyką bezkontekstową bowiem rozpoznają tego typu języki. Choć istnieją języki które nie można rozpoznać za pomocą automatu stosowego choć można opracować program dla komputera uniwersalnego, który rozpozna taki język.

Przejścia automat stosowego dość podobne do innych automatów jednak ze względu na obecność stosu, czynności jakie wykonuje automat są nieco inne:

- automat może pobrać z wejścia symbol, który zostanie użyty podczas przejścia, jeśli zamiast wejścia zastosowany zostanie symbol ε, to nie następuje pobranie żadnego symbolu wejściowego,
- automat przechodzi do nowego stanu (może to być tak sam stan jak poprzedni),

- automat może zastąpić symbol na wierzchołku stosu, jeśli symbolem jest ε, to usuwany jest element ze szczytu stosu. Automat może zmienić symbol znajdujący się na wierzchołku stosu, bądź też pozostawić go w tym samym miejscu. Symbol znajdujący się na wierzchołku może zostać zastąpiony większą ilością symboli. Nowe symbole mogą też zostać dopisane do stosu. W tym przypadku naturalnie ostatni dodany symbol będzie stanowić nowy wierzchołek.

Zaprojektujemy w sposób nieformalny automat do rozpoznawania palindromów o parzystej długości nad alfabetem $\{0, 1\}$, definicja języka jest następująca:

$$L_{palin_p} = \{ww^R \mid w \in (0+1)^*\}$$

a gramatyka bezkontekstowa, która opisuje ten język jest następująca:

$$\begin{array}{lcl} P & \rightarrow & \varepsilon \\ P & \rightarrow & 0P0 \\ P & \rightarrow & 1P1 \end{array}$$

Nasz nieformalny opis automatu przedstawia się następująco:

- Funkcjonowanie automat rozpoczyna w stanie q_0. Stan q_0 oznacza, że jeszcze nie widziano środka całego łańcucha — to oznacza, iż nie widzieliśmy końca łańcucha w, po którym ma nastąpić jego odwrócenie. W stanie q_0, czytamy także kolejne symbole i są one odkładane na stos.

- Jednakże w dowolnej chwili pracy automatu można „zgadywać", że widziano środek całego łańcucha, tzn. koniec w. W tym momencie w będzie znajdować się na stosie, z prawym końcem w na wierzchołku, a lewym końcem u początku stosu. Ten fakt może zostać oznaczony przez spontaniczne przejście do stanu q_1. Ponieważ automat jest niedeterministyczny, więc wybierane są dwa stany. Automat „domyśla się" że widział już koniec w, ale jednocześnie pozostaje w stanie q_0 nadal czytając symbole wejściowe i składując je na stosie.

- Znajdując się w stanie q_1, należy porównywać kolejne symbole wejściowe z symbolem znajdującym się na wierzchołku stosu. Jeśli pasują one do siebie, pobieramy kolejny symbol wejściowy, i po usunięciu elementy z wierzchołka następuje porównanie następnego symbolu. W przypadku gdy symbole nie pasują to oznacza iż po słowie w nie nastąpiło słowo w^R. Gałąź automatu zamiera, choć inne gałęzie niedeterministycznego automatu mogą nadal istnieć i doprowadzić do akceptacji.

- Jeśli stos jest pustym, to istotnie widziano jakieś słowo w, po którym nastąpiło w^R. Pozwala to na akceptację przeczytanych dotąd symboli.

2.1 Definicja formalna

Formalna notacja dla automatów ze stosem obejmuje siedem składowych:

$$P = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\}$$

Poszczególne elementy mają następujące znaczenie:

Q: Skończony zbiór stanów, podobnych do stanów automatu skończonego.

Σ: Skończony zbiór symboli wejściowych, także analogiczny do odpowiedniej składowej automatu skończonego.

Γ: Skończony alfabet stosowy. Składowa ta, nie ma odpowiednika w automacie skończonym, jest zbiór symboli, które można umieszczać na stos bądź usuwać ze stosu.

δ: Funkcja przejścia. Analogicznie jak dla automatu skończonego, δ odpowiada za zachowanie automatu. Funkcja ta bierze jako argument trójkę (q, a, X), gdzie:

(a) q jest stanem z Q,

(b) a jest symbolem wejściowym z Σ lub ε, łańcuchem pustym, który (zgodnie z założeniem) nie jest symbolem wejściowym,

(c) X jest symbolem stosowym, czyli elementem Γ.

Wejściem dla funkcji δ jest skończony zbiór par (p, γ), gdzie p jest nowym stanem, a γ jest łańcuchem symboli stosowych, który zastępuje X na wierzchołku stosu. Na przykład, jeśli $\gamma = \varepsilon$, to element znajdujący na szczycie stosu jest usuwany, jeśli $\gamma = X$, to stos pozostaje bez zmian, a jeśli $\gamma = YZ$, to X jest zastępowane przez Z, Y jest umieszczane na szczycie stosu.

q_0: Stan początkowy. Automat ze stosem przez rozpoczęciem pracy znajduje się w tym stanie.

Z_0: Symbol początkowy. Stos zawiera jeden symbol tego typu przed rozpoczęciem pracy.

F: Zbiór stanów akceptujących.

2.1.1 Automat ze stosem dla parzystych palindromów

Formalna definicja automatu podobnie jak w innych automatach skończonych, wymaga dokładnej specyfikacji funkcji δ. Jako przykład posłuży automat zdolny do rozpoznawania palindromów parzystych opisanych przez następujący język:

$$L_{palin_p} = \{ww^R \mid w \in (0+1)^*\}$$

W definicji automatu istotny jest symbol Z_0 pełni bardzo ważna rolę. Ponieważ jest umieszczony na samym początku, to stanowi „swoiste" dno stosu. Jeśli to wykonaniu wszystkich operacji wierzchołek stosu będzie zawierał ten symbol oznaczać to będzie, że możemy przejść do stanu akceptującego, uprzedzając dokładny opis będzie to stan q_2.

Formalnie automat jest opisano jako następująca krotka:

$$P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$$

Najistotniejsza jest naturalnie postać funkcji przejścia, w przypadku tego automatu, δ jest określona za pomocą następujących reguł:

1. $\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$ oraz $\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$. Jedna z podanych reguł jest stosowana na początku, gdy automat znajduje się w stanie q_0 i widziany jest symbol Z_0 znajdujący się na wierzchołku stosu. Odczytywane jest pierwsze wejście i odkładane jest na stos, symbol Z_0 zgodnie z wcześniejszą umową oznacza spód stosu.

2. $\delta(q_0, 0, 0) = \{(q_0, 00)\}$, $\delta(q_0, 0, 1) = \{(q_0, 01)\}$, $\delta(q_0, 1, 0) = \{(q_0, 10)\}$ oraz $\delta(q_0, 1, 1) = \{(q_0, 11)\}$. Wymienione cztery przejścia pozwalają pozostawać w stanie q_0 i czytać wejście, odkładając każdy element na stos a pozostałe symbole pozostają na stosie bez zmian.

3. $\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\}$, $\delta(q_0, \varepsilon, 0) = \{(q_1, 0)\}$ oraz $\delta(q_0, \varepsilon, 1) = \{(q_1, 1)\}$. Wymienione reguły pozwalają aby automat wykonał spontaniczne przejście z q_0 do stanu q_1, a symbol znajdujący się na wierzchołku stosu pozostaje nienaruszony.

4. $\delta(q_0, 0, 0) = \{(q_0, \varepsilon)\}$ oraz $\delta(q_0, 1, 1) = \{(q_0, \varepsilon)\}$. Będąc w stanie q_1 możliwe jest porównywanie symboli z wierzchołka stosu i zdejmować symbole, jeśli pasują one do symboli wejściowych.

5. $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$. Jest to ostatnia reguła, która oznacza, iż jeśli automat znajdzie się w stanie q_1 i zostanie odkryty symbol spodu Z_0, to oznacza iż słowo wejściowe było w postaci ww^R. Przechodząc do stanu q_2 akceptujemy wejście.

2.2 Przekształcanie gramatyki na automat ze stosem

Przedstawiona formalna definicja automatu ze stosem zakłada iż akceptacja słowa wejściowego nastąpi jeśli automat znajdzie się w jednym ze swoich stanów końcowym. Innym sposobem opisania sytuacji, gdy słowo wejściowe ma zostać zaakceptowane jest doprowadzenie do sytuacji, gdy stos automatu jest pusty bądź znajduje się na nim tylko symbol Z_0.

Zmiana gramatyki na automat stosowy jest dość prosta jeśli uzyskany automat będzie akceptował przez pusty stos oraz sama gramatyka będzie w postaci lewostronnej np.: $S \to Sa$. Dla danej gramatyki bezkontekstowej

$G = (V, T, Q, S)$ automat ze stosem P akceptujący język gramatyki G jest budowany za pomocą następujących reguł:

$$P = (\{q\}, T, V \cup T, \delta, q, S)$$

gdzie funkcja δ jest określona w następujący sposób:

(I) Dla każdej zmiennej A mamy

$$\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \to \beta \text{ jest produkcją } G\}$$

(II) Dla każdego symbolu końcowego a

$$\delta(q, a, a) = \{(q, \varepsilon)\}$$

Sposób użycia tych reguł zostanie pokazany na przykładzie konwersji następującej gramatyki:

$$\begin{array}{lcl} I & \to & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\ E & \to & I \mid E * E \mid E + E \mid (E) \end{array}$$

Zbiorem symboli końcowych dla automatu stosowego jest zbiór $\{a, b, 0, 1, (,), +, *\}$. Wymienione elementy oraz symbol I, E tworzą także alfabet stosowy. Funkcja przejścia δ jest określona przez następujące reguły:

1. $\delta(q, \varepsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)\}$.
2. $\delta(q, \varepsilon, E) = \{(q, I), (q, E + E), (q, E * E), (q, (E))\}$
3. $\delta(q, a, a) = \{(q, \varepsilon)\}$ $\delta(q, b, b) = \{(q, \varepsilon)\}$ $\delta(q, 0, 0) = \{(q, \varepsilon)\}$
   $\delta(q, 1, 1) = \{(q, \varepsilon)\}$ $\delta(q, (, () = \{(q, \varepsilon)\}$ $\delta(q,),)) = \{(q, \varepsilon)\}$
   $\delta(q, +, +) = \{(q, \varepsilon)\}$ $\delta(q, *, *) = \{(q, \varepsilon)\}$

Reguły 1 i 2 funkcji δ są generowana przez regułę (I), jednakże osiem dodatkowych przejść 3 jest produkowane przez regułę (II).

3 Zadania

1. Jakie wyrazy są generowane przez następującą gramatykę bezkontekstową: $G = (V, T, P, S)$, gdzie $V = \{S\}$, $T = \{a, b\}$ oraz $P = \{S \to aSb, S \to ab\}$.
2. Podać gramatyki bezkontekstowe generujące następujące zbiory:
   (a) zbiór palindromów nad alfabetem $\{a, b\}$
   (b) zbiór wszystkich łańcuchów zrównoważonych nawiasów
   (c) zbiór wszystkich łańcuchów nad alfabetem $\{a, b\}$ zawierających dokładnie dwa razy więcej symboli a niż symboli b
3. Podać, jakie języki są generowane przez poniższe gramatyki (słowa opisać w sposób nie formalny, podać przykłady):[1]

   (a)
   $$\begin{array}{lcl} S & \to & aS \mid bS \mid aA \\ A & \to & aA \mid bA \mid aB \\ B & \to & aB \mid bB \mid \lambda \end{array}$$

   (b)
   $$\begin{array}{lcl} S & \to & aS \mid bS \mid aA \\ A & \to & aB \\ B & \to & aB \mid bB \mid \lambda \end{array}$$

[1] Wielkie litery łacińskie oznaczają symbole nieterminalne, małe litery łacińskie, cyfry oraz znaki specjalne, jak np. nawiasy okrągłe lub kwadratowe, oznaczają symbole terminalne, symbolem początkowym jest nieterminal S, symbol λ reprezentuje symbol pusty.

(c)
$$\begin{array}{lcl} S & \to & aS \mid bS \mid aA \\ A & \to & aB \mid bB \\ B & \to & \lambda \end{array}$$

4. Podać, jakie języki są generowane przez poniższe gramatyki (słowa opisać w sposób formalny):

(a)
$$\begin{array}{lcl} S & \to & SAB \mid AB \\ A & \to & Aa \mid a \\ B & \to & Bb \mid b \end{array}$$

(b)
$$\begin{array}{lcl} S & \to & ABC \\ A & \to & aAb \mid ab \\ B & \to & bBa \mid ba \\ C & \to & aCb \mid ab \end{array}$$

(c)
$$\begin{array}{lcl} S & \to & AC \\ A & \to & aAb \mid aBb \\ B & \to & bBa \mid ba \\ C & \to & aCb \mid ab \end{array}$$

(d)
$$\begin{array}{lcl} S & \to & aAb \mid aSa \\ A & \to & bAcc \mid ab \end{array}$$

(e)
$$\begin{array}{lcl} S & \to & aSb \mid aAb \\ A & \to & bAa \mid bBa \\ B & \to & aBb \mid ab \end{array}$$

5. Podać przykłady słów generowanych przez następujące gramatyki:

(a)
$$\begin{array}{lcl} S & \to & A \mid B \mid C \\ A & \to & ab \mid aaaA \\ B & \to & bbc \mid bbbB \\ C & \to & ccca \mid cccC \end{array}$$

(b) $S \to a \mid ab \mid abc \mid abcS$

(c)
$$\begin{array}{lcl} S & \to & SS \mid [S] \mid A \\ A & \to & AA \mid (A) \mid \lambda \end{array}$$

6. Podać gramatyki które będą generować poniższe słowa:

(a) $\{ ab, bbc, ccca, aaaab, bbbbbc, cccccca, aaaaaaab, \dots \}$

(b) $\{ a, b, ab, ba, aba, bab, abab, baba, ababa, babab, \dots \}$

(c) $\{ abb, abbaab, abbaababb, abbaababbaab, \dots \}$

(d) $\{ba^m ba^{m+n}c \mid n \geqslant 1, m \geqslant 1\}$

(e) $\{a^i b^j \mid i \geqslant 0, j \geqslant 0, i \geqslant j\}$

(f) $\{(aabb)^i \mid i \geqslant 0\}$

(g) $\{a^{3i} \mid i \geqslant 1\}$

7. Podać gramatyki które będą generować poniższe słowa:

(a) $\{a, b\}^*$

(b) $\{\lambda\}$

(c) $\{\lambda, a, b, ab, ba\}$

(d) język nad alfabetem $\{0, 1\}$ będący zbiorem wszystkich łańcuchów zawierających co najmniej dwa zera

(e) język nad alfabetem $\{0, 1\}$ będący zbiorem wszystkich łańcuchów zawierających dwa kolejne zera

(f) język nad alfabetem $\{0, 1\}$ będący zbiorem wszystkich łańcuchów, w których przedostatni symbol jest jedynką

(g) język nad alfabetem $\{0, 1\}$ będący zbiorem wszystkich łańcuchów, w których drugi licząc od początku symbol jest jedynką

(h) $\{\ a^n b^m a^m b^n a^k b^k \mid n \geqslant 1, m \geqslant 1, k \geqslant 1\ \}$

(i) język nad alfabetem $\{a, b\}$ będący zbiorem wszystkich łańcuchów rozpoczynających się od symbolu a i kończących się symbolem b

8. Wyznaczyć drzewa wyprowadzeń dla następujących wyrażeń arytmetycznych:

(a) $10 - d + (a \cdot 56) - a \cdot (10 \cdot c)$

(b) $\cos(a - \sin(\frac{\pi}{3a}))$

(c) $a + b^2 + c^3 + d$

(d) $\frac{|1-\cos(2(3-\log 3))|}{3}$

9. Podać gramatyką dla prostych wyrażeń arytmetycznych i podać dwa różne drzewa wyprowadzeń dla wyrażenia $42 + 5 \cdot 2$.

(a) W jaki sposób należy zapisać gramatykę aby uniknąć niejednoznaczność?

(b) Podać gramatykę i drzewo wyprowadzenia dla wymienionego wyrażenia.

10. Dla gramatyki instrukcji warunkowej **if** o następującej postaci

stmt → **if** expr **then** stmt
if expr **then** stmt **else** stmt
other

(a) Narysować drzewo wyprowadzenia dla instrukcji: **if** E_1 **then if** E_2 **then** S_1 **else** S_2.

(b) Jak należy uściślić gramatykę aby gramatyka była jednoznaczna?

11. Podać opisy w notacji BNF:

(a) czterech podstawowych wyrażeń arytmetycznych dla języka Pascal łącznie z operacją przypisania,

(b) języka maszyny licznikowej,

(c) języka maszyny o dostępie swobodnym (RAM).

12. Podać opisy w notacji BNF:

(a) podać opis dla instrukcji warunkowej dla języka Pascal,

(b) podać opis dla instrukcji pętli while dla języka Pascal,

(c) podać opis dla instrukcji pętli repeat until dla języka Pascal,

(d) podać opis dla instrukcji wielokrotnego wyboru.

13. Podać gramatykę zdań w języku polskim zbudowanych z rzeczownika, czasownika, przymiotnika, przysłówka.

14. Podać opisy wykorzystując wyrażenia regularne następujących słów:

(a) $a, aa, aaa, aaaa, \ldots$

(b) $ab, aba, ababa, abababa, \ldots$

(c) $az, abz, abcz, abbz, abbccz, abbccdz, a \ldots z$

15. Przekształcić gramatyki bezkontekstowe do postaci normalnej Chomsky'ego.

(a)
$$\begin{array}{lcl} S & \to & aAb \mid aSa \\ A & \to & bAc \mid ab \end{array}$$

(b)
$$\begin{array}{lcl} S & \to & AB \\ A & \to & aAA \mid \lambda \\ B & \to & bBB \mid \lambda \end{array}$$

(c)
$$\begin{array}{lcl} S & \to & aaSb \mid bAa \\ A & \to & aAbb \mid a \end{array}$$

(d)
$$\begin{array}{lcl} S & \to & bA \mid aB \\ A & \to & aAA \mid aS \mid a \\ B & \to & aBB \mid bS \mid b \end{array}$$

16. Przekształcić gramatyki na odpowiedni automat ze stosem:

(a)
$$\begin{array}{lcl} S & \to & 0S1 \mid A \\ A & \to & 1A0 \mid S \mid \varepsilon \end{array}$$

(b)
$$\begin{array}{lcl} S & \to & aAA \\ A & \to & aS \mid bS \mid a \end{array}$$

17. Zbudować automat ze stosem akceptujący następujące języki:

(a) $\{a^{3n+1}b^{5n+1} \mid n \geqslant 0\}$

(b) $\{a^{3n+2}b^{7n+2} \mid n \geqslant 0\}$

(c) $\{a^{3n+2}b^{4n+3} \mid n \geqslant 0\}$

(d) $\{a^{3n+3}b^{5n+3} \mid n \geqslant 0\}$

4 Dalsze informacje

Poniższe pozycje odnoszą się do wszystkich list z ćwiczeniami z przedmiotu teoretyczne podstawy informatyki.

Literatura

[1] David Harel: Rzecz o istocie informatyki Algorytmika, Edycja polska Wydanie drugie Wydawnictwa Naukowo-Techniczne 2000

[2] Tomasz Bilski, Krzysztof Chmiel, Janusz Stokłosa: Zbiór zadań ze złożoności obliczeniowej algorytmów. Politechnika Poznańska 1992

[3] Janusz Stokłosa: Zadania ze złożoności obliczeniowej algorytmów, Politechnika Poznańska 1989

[4] L. Banachowski, Antoni Kreczmar: Elementy analizy algorytmów, Wydawnictwa Naukowo-Techniczne 1982

[5] John E.Hopcroft, Jeffrey D.Ullman: Wprowadzenie do teorii automatów, języków i obliczeń. Wydawnictwo Naukowe PWN 2003

[6] Mordechai Ben-Ari: Logika matematyczna w informatyce, Wydawnictwa Naukowo-Techniczne 2005.

[7] Christos H.Papadimitriou: Złożoność obliczeniowa, Wydawnictwa Naukowo-Techniczne 2002.

[8] R.L. Graham, D.E. Knuth, O.Patashnik: Matematyka konkretna,Wydawnictwo Naukowe PWN 2002.

[9] Kenneth A.Ross, Charles R.B.Wright: Matematyka dyskretna, Wydawnictwo Naukowe PWN 2000.

[10] Piotr Wróblewski,: Algorytmy struktury danych i techniki programowania, Helion 1997.

[11] Materiały ze strony dr inż. Janusza Majewskiego dotyczące przedmiotu „Automaty i języki formalne”.