MINUT 2019 (1), s. 10–16
ISSN 2719-3063

Alina MOMOT

Wydział Automatyki, Elektroniki i Informatyki, Politechnika Śląska,
ul. Akademicka 16, 44-100 Gliwice

Gramatyki formalne – kilka sposobów rozwiązania prostego zadania

Streszczenie. Jednym z kluczowych pojęć w informatyce jest język formalny, rozumiany jako zbiór skończonej długości sekwencji symboli (słów), które to symbole pochodzą z pewnego skończonego zbioru (alfabetu). Do opisu takiego języka służą gramatyki formalne, które wykorzystując rekurencję podają precyzyjne i jednoznaczne reguły tworzenia poprawnych słów w danym języku. Ponieważ jednak pomiędzy językiem a generującą go gramatyką nie ma wzajemnie jednoznacznej odpowiedniości, możliwe jest określenie różnych gramatyk definiujących ten sam język. Artykuł prezentuje rozwiązanie pewnego prostego zadania: definiuje kilka różnych równoważnych gramatyk generujących ten sam język formalny dla którego alfabetem jest zbiór $\{a, b, c\}$.

1. Wstęp teoretyczny

Pierwsze badania naukowe mające na celu próby opisu struktur języka rozpoczęły się w pierwszej połowie 20 wieku. W Stanach Zjednoczonych powstał ruch zwany amerykańską szkołą strukturalistyczną (ang. *American structuralism*) zapoczątkowany przez Leonarda Bloomfielda. W 1933 roku wydał on swój podręcznik [1], który prezentuje teorię, pozwalającą na opis języka na podstawie samej formy tekstów powstających w tym języku. Szkoła Bloomfielda postulowała, że lingwistyka ma charakter deskryptywny, w której kluczowe są procedury odkrywcze zaprezentowane w 1951 roku przez Zelliga Sabbetaia Harrisa [4].

Formalne procedury analizy danych językowych zapoczątkowane przez bloomfieldowskie językoznawstwo strukturalne zostały przedstawione w opublikowanej w 1957 roku przez Noama Chomsky'ego teorii gramatyki generatywnej [3]. W pracy tej został opracowany matematyczny formalizm opisu języków naturalnych. Bazuje on na procesie sekwencyjnego przepisywania, czyli modyfikowania pewnych ciągów symboli, za pomocą reguł przyjętych za dopuszczalne w danym systemie. Idea ta została zapoczątkowana w 1914 roku i badana przez takich matematyków jak Axel Thue, Emil Post czy też Alan Turing.

Teoria gramatyk formalnych zapoczątkowana i opracowana przez Noama Chomsky'ego znalazła istotne zastosowanie w informatyce, szczególnie w pracach badawczych dotyczących języków programowania, gdzie potrzebny jest analizator składniowy (parser) dokonujący analizy składniowej danych wejściowych w celu stwierdzenia poprawności ich struktury gramatycznej [5].

Autor korespondencyjny: A. Momot (alina.momot@polsl.pl).
Data wpłynięcia: 01.08.2019.

Poniżej zostanie przestawiona klasyczna definicja gramatyki zaproponowana przez Noama Chomsky'ego.

Definicja 1 (Gramatyka). *Gramatyką G nazywamy uporządkowaną czwórkę $< V, \Sigma, P, \sigma >$, gdzie poszczególne składniki to:*

- *V – skończony zbiór symboli terminalnych (symboli alfabetu, na którym zbudowany jest język),*
- *Σ – skończony zbiór symboli nieterminalnych (symboli pomocniczych tworzących metajęzyk),*
- *P – skończony zbiór reguł przepisywania (zwanych też produkcjami),*
- *σ – symbol początkowy (zwany też startowym lub aksjomatem gramatyki), którym to jest jeden z symboli nieterminalnych.*

Reguła przepisywania (produkcja), to para słów (*lewe→prawe*), w których mogą występować symbole terminalne i nieterminalne. Zakładamy ponadto, że alfabety terminalny i nieterminalny są rozłącznymi zbiorami, a słowo występujące po lewej stronie produkcji zawiera co najmniej jeden symbol nieterminalny. W pracy wydanej w 1956 roku [2], Chomsky zaproponował podział gramatyk na pewne cztery inkluzyjne klasy, zwany hierarchią Chomsky'ego. Każda z klas generuje pewien typ języka formalnego:

język regularny (klasa 3), w którym lewa strona produkcji zawiera pojedynczy symbol nieterminalny, a prawą stanowi słowo zawierające co najwyżej jeden symbol nieterminalny (albo na początku albo na końcu słowa),

język bezkontekstowy (klasa 2), w którym lewa strona produkcji zawiera pojedynczy symbol nieterminalny, a prawą stanowi dowolne słowo,

język kontekstowy (klasa 1), w którym produkcje są postaci $\alpha A\beta \to \alpha\gamma\beta$, gdzie α i β są dowolnymi słowami, A jest symbolem nieterminalnym, zaś γ jest niepustym słowem,

język rekurencyjnie przeliczalny (klasa 0), w którym lewą i prawą stronę produkcji stanowi dowolne słowo.

Zakładamy przy tym, że do języka należy każde słowo, które można wyprowadzić z symbolu startowego. Zaś wyprowadzeniem lub wywodem słowa nazywamy ciąg przekształceń, w którym stosując reguły przepisywania (produkcje) wychodząc od symbolu startowego otrzymujemy jako ostatni element ciągu dane słowo. Przy czym pojedyncze przekształcenie należy rozumieć jako zastąpienie w poprzednim słowie fragmentu równemu lewemu słowu jakiejś reguły przez prawe słowo tej reguły.

Do opisu gramatyk bezkontekstowych często wykorzystuje się notację zaproponowaną przez Johna Backusa jako metajęzyk do opisu składni nowego ówcześnie języka programowania IAL, znanego obecnie jako Algol58. Notacja ta spopularyzowana przez Petera Naura, który jako redaktor raportu ALGOL użył jej do opisywania pełnej składni języka ALGOL, nosi dziś nazwę BNF (ang. *Bacus-Naur Form*). Obecnie używa się wiele jej różnych wariantów znanych jako EBNF (ang. *Extended Bacus-Naur Form*) [6]. Notacja ta jest szczególnie użyteczna do opisu składni komputerowych języków programowania, zbiorów instrukcji, formatów dokumentów, czy też protokołów komunikacyjnych.

Podstawowe założenia notacji BNF to tworzenie reguł przepisywania (produkcji) postaci

$$< \alpha >::= \beta|\gamma,$$

gdzie

- symbole $<, >$ są ogranicznikami symbolu nieterminalnego,
- symbol ::= (czytany „jest zdefiniowany jako") zastępuje $\rightarrow$ w zapisie produkcji,
- symbol | wskazuje na alternatywę przy tworzeniu produkcji.

2. Zadanie

Jak wspomniano wcześniej, językiem generowanym przez gramatykę G jest zbiór wszystkich słów, które można wyprowadzić z symbolu początkowego tej gramatyki (oznaczamy go symbolem $L(G)$). Jednak o ile jedna gramatyka generuje tylko jeden język, to możliwe jest wygenerowanie tego samego języka stosując różne gramatyki. Mówimy wtedy, że gramatyki te są równoważne (językowo). Poniżej zostanie zdefiniowane, używając notacji BNF, kilka różnych równoważnych gramatyk generujących ten sam język formalny dla którego alfabetem jest zbiór $\{a, b, c\}$.

Sformułowanie problemu

Zdefiniować gramatykę G generującą język $L(G) = \{a^k b^m c^n : k, m, n = 1, 2, \ldots\}$.

Zacznijmy od analizy treści zadania. Zapis $\{a^k b^m c^n : k, m, n = 1, 2, \ldots\}$ oznacza, że

- poprawne słowa naszego języka zawierają jedynie litery a, b i c,
- liczba poszczególnych liter jest dowolna (ale przynajmniej jedna litera każdego typu wystąpi w słowie),
- kolejność liter jest istotna (najpierw występują litery a, po nich litery b, a na koniec litery c).

Zatem przykłady poprawnych słów to: *abc*, *aabc*, *abbbcc* itp. Natomiast słowa *bca*, *ccab* lub *abac* są niepoprawne (nie należą do języka). Bazując na tej analizie, zdefiniujmy pierwszą gramatykę generującą opisany wyżej język.

Przykład 1. $G_1 =< V_1, \Sigma_1, P_1, \sigma_1 >$, gdzie poszczególne składniki to:

$$
\begin{aligned}
V_1 = &\ \{a, b, c\},\\
\Sigma_1 = &\ \{< a >, < b >, < c >, < abc >\},\\
P_1 = &\ \{< a >::= a| < a > a,\\
&\ < b >::= b| < b > b,\\
&\ < c >::= c| < c > c,\\
&\ < abc >::=< a >< b >< c >\},\\
\sigma_1 = &\ < abc > .
\end{aligned}
$$

Zauważmy, że najpierw zdefiniowaliśmy symbole alfabetu, na którym zbudowany jest nasz język ($V_1 = \{a, b, c\}$), następnie określiliśmy symbole pomocnicze ($\Sigma_1 = \{< a >, < b >, < c >, < abc >\}$) oraz stwierdziliśmy, że naszym symbolem startowym jest jeden z nich, czyli $< abc >$. Teraz przyjrzyjmy się poszczególnym produkcjom.

Zapis $< a >::= a| < a > a$ oznacza, że symbol nieterminalny $< a >$ jest zdefiniowany jako pojedyncza litera a albo litera a dopisana po prawej stronie wygenerowanego wcześniej symbolu $< a >$. Z rekurencyjnej formuły zapisu tej produkcji wynika zatem, że symbolem nieterminalnym $< a >$ może być dowolnej długości ciąg liter a.

Symbole $< b >$ i $< c >$ zdefiniowane są w analogiczny sposób. Natomiast symbol startowy $< abc >$ jest konkatenacją (połączeniem) pozostałych symboli nieterminalnych w odpowiedniej kolejności.

Przeprowadźmy teraz wywód przykładowego słowa naszego języka, a mianowicie *aaabbc*:

$$< abc > \rightarrow < a >< b >< c > \rightarrow < a > a < b > bc \rightarrow < a > aabbc \rightarrow aaabbc.$$

Warto przy tym zauważyć, że przedstawiona wyżej metoda wywodu, to tzw. metoda zstępująca (ang. *top-down*). Jednak, gdybyśmy zmienili kierunek strzałek otrzymalibyśmy tzw. wywód wstępujący (ang. *bottom-up*) dzięki któremu moglibyśmy pokazać, jak słowo *aaabbc* przekształcić do postaci symbolu początkowego $< abc >$ naszej gramatyki.

Przykład 2. $G_2 =< V_2, \Sigma_2, P_2, \sigma_2 >$, gdzie poszczególne składniki to:

$V_2 = \{a, b, c\}$,
$\Sigma_2 = \{< a >, < ab >, < abc >\}$,
$P_2 = \{< a >::= a| < a > a,$
$< ab >::=< a > b| < ab > b,$
$< abc >::=< ab > c| < abc > c\}$,
$\sigma_2 = < abc >$.

Zauważmy, że w tym przykładzie nie zmienił się alfabet terminalny ($V_1 = V_2$). Również symbol startowy oznaczony jest w ten sam sposób ($< abc >$), a istotna różnica dotyczy sposobu jego produkcji. W tym przykładzie generując słowo zaczynamy od lewej strony dopisując kolejno z prawej kolejne, odpowiednie litery.

Przeprowadźmy teraz analogiczny względem poprzedniej gramatyki wywód opisywanego wcześniej słowa *aaabbc*:

$$< abc > \rightarrow < ab > c \rightarrow < ab > bc \rightarrow < a > bbc \rightarrow < a > abbc \rightarrow aaabbc.$$

Jeśli porównamy te dwie gramatyki, to zauważymy, że potrafią one wygenerować te same słowa, aczkolwiek odbywa się to w nieco inny sposób. Ponadto warto przy tym zauważyć, że gramatyka G_1 jest przykładem gramatyki bezkontekstowej (lewa strona produkcji zawiera pojedynczy symbol nieterminalny, a prawą stanowi dowolne słowo), natomiast G_2 należy do węższej klasy gramatyk regularnych, a dokładniej jest gramatyką regularną lewostronną. Zaś poniższa gramatyka G_3 stanowi przykład gramatyki regularnej prawostronnej.

Przykład 3. $G_3 =< V_3, \Sigma_3, P_3, \sigma_3 >$, gdzie poszczególne składniki to:

$V_3 = \{a, b, c\}$,
$\Sigma_3 = \{< c >, < bc >, < abc >\}$,
$P_3 = \{< c >::= c|c < c >,$
$< bc >::= b < c > |b < cb >,$
$< abc >::= a < bc > |a < abc >\}$,
$\sigma_3 = < abc >$.

Widzimy więc, że w gramatykach regularnych symbole nieterminalne, występujące po prawej stronie w definicjach produkcji, pojawiają się tylko po lewej (gramatyki regularne lewostronne) albo prawej stronie (gramatyki regularne prawostronne) względem symboli alfabetu terminalnego.

Innym (względem G_1) przykładem gramatyki bezkontekstowej jest poniższa gramatyka G_4. Zauważmy, że nie jest ona gramatyką regularną, gdyż po prawej stronie w definicji produkcji symbole nieterminalne występują zarówno po prawej jak i lewej stronie względem symboli alfabetu terminalnego.

Przykład 4. $G_4 =< V_4, \Sigma_4, P_4, \sigma_4 >$, gdzie poszczególne składniki to:

$V_4 = \{a, b, c\}$,
$\Sigma_4 = \{< b >, < abc >\}$,
$P_4 = \{< b >::= b | b < b >,$
$< abc >::= a < b > c | a < abc > | < abc > c\}$,
$\sigma_4 = < abc >$.

Analizując produkcję $< abc >::= a < b > c | a < abc > | < abc > c$ warto zauważyć, że do poprawnego określenia słów naszego języka potrzeba aż trzech definicji:

$$< abc > \rightarrow a < b > c; \quad < abc > \rightarrow a < abc >; \quad < abc > \rightarrow < abc > c.$$

Zauważmy bowiem, że produkcja postaci $< abc >::= a < b > c | a < abc > c$ generuje jedynie słowa w których liczba liter a oraz c jest jednakowa i taka gramatyka nie byłaby w stanie wygenerować słowa $aaabbc$, natomiast gramatyka G_4 jest w stanie:

$$< abc > \rightarrow a < abc > \rightarrow aa < abc > \rightarrow aaa < b > c \rightarrow aaab < b > c \rightarrow aaabbc.$$

Na koniec zauważmy jeszcze, że tworząc naszą gramatykę G generującą język $L(G) = \{a^k b^m c^n : k, m, n = 1, 2, \ldots\}$ można spojrzeć na ten problem jeszcze od nieco innej strony i zaproponować jako rozwiązanie poniższe gramatyki G_5 oraz G_6.

Przykład 5. $G_5 =< V_5, \Sigma_5, P_5, \sigma_5 >$, gdzie poszczególne składniki to:

$V_5 = \{a, b, c\}$,
$\Sigma_5 = \{< ab >, < abc >\}$,
$P_5 = \{< ab >::= ab | a < ab > | < ab > b,$
$< abc >::=< ab > c | < abc > c\}$,
$\sigma_5 = < abc >$.

Przykład 6. $G_6 =< V_6, \Sigma_6, P_6, \sigma_6 >$, gdzie poszczególne składniki to:

$V_6 = \{a, b, c\}$,
$\Sigma_6 = \{< bc >, < abc >\}$,
$P_6 = \{< bc >::= bc | b < bc > | < bc > c,$
$< abc >::= a < bc > | a < abc >\}$,
$\sigma_6 = < abc >$.

Oczywiście zarówno gramatyka G_5 jak i G_6 jest w stanie wygenerować słowo $aaabbc$. Wywód tego słowa przy zastosowaniu produkcji gramatyki G_5 jest następujący:

$$< abc > \rightarrow < ab > c \rightarrow a < ab > c \rightarrow aa < ab > c \rightarrow aa < ab > bc \rightarrow aaabbc,$$

zaś stosując produkcje gramatyki G_6, otrzymujemy:

$$< abc > \rightarrow a < abc > \rightarrow aa < abc > \rightarrow aaa < bc > \rightarrow aaab < bc > \rightarrow aaabbc.$$

Widzimy również, że gramatyki G_5 i G_6 stanowią kolejne przykłady gramatyk bezkontekstowych.

3. Wnioski końcowe

Podsumowując nasze rozważania, warto zauważyć, że możliwość zdefiniowania tak dużej liczby różnych gramatyk generujących język $L(G) = \{a^k b^m c^n : k, m, n = 1, 2, \ldots\}$ wynika z faktu, że język ten jest stosunkowo elastyczny i nie ma zbyt wielu ograniczeń na budowanie słów tworzących ten język. Jeśli wprowadzimy dodatkowe ograniczenie i założymy dodatkowo, że liczba liter a i c ma być jednakowa, narzuci nam to bardziej ukierunkowane na wymagany rezultat rozwiązanie (analizowane już uprzednio przy okazji tworzenia gramatyki G_4).

Przykład 7. $G_4^* =< V_4^*, \Sigma_4^*, P_4^*, \sigma_4^* >$, gdzie poszczególne składniki to:

$$\begin{aligned} V_4^* = &\ \{a, b, c\}, \\ \Sigma_4^* = &\ \{< b >, < abc >\}, \\ P_4^* = &\ \{< b > ::= b | b < b >, \\ &\ < abc > ::= a < b > c | a < abc > c\}, \\ \sigma_4^* = &\ < abc > . \end{aligned}$$

Przy tej okazji warto też dodać, że pomimo iż tworzące nasz język gramatyki $G_1, G_2, \ldots, G_6$ należą do różnych klas w hierarchii Chomsky'ego (wszystkie są gramatykami bezkontekstowymi, ale tylko gramatyki G_2 i G_3 są regularne) generowany przez nie język jest językiem regularnym (typu 3). Język, który generuje gramatyka G_4^* jest językiem bezkontekstowym, czyli typu 2, gdyż nie istnieje gramatyka regularna, która byłaby w stanie wygenerować język $L(G^*) = \{a^k b^m c^k : k, m = 1, 2, \ldots\}$. Zaś dodanie warunku polegającego na tym, że liczba wszystkich typów liter ma być jednakowa spowoduje, że języka $\{a^k b^k c^k : k = 1, 2, \ldots\}$ nie jest w stanie wygenerować żadna gramatyka bezkontekstowa i należy on do klasy języków kontekstowych (typu 1).

Literatura

1. L. Bloomfield, *Language*, The University of Chicago Press, Chicago London 1933.
2. N. Chomsky, *Three models for the description of language*, IRE Transactions on Information Theory 2(3) (1956), pp. 113-124.
3. N. Chomsky, *Syntactic Structures*, Mouton and Co., The Hague 1957.
4. Z.S. Harris, *Methods in Structural Linguistic*, The University of Chicago Press, Chicago 1951.
5. A. Nijholt, *Context-Free Grammars: Covers, Normal Forms, and Parsing*, Springer, Berlin Heidelberg 1980.
6. A.A. Puntambekar, *Formal Languages And Automata Theory*, Technical Publications Pune, Pune 2008.