# Matematyczna wieża Babel.
# 3. Gramatyki – o językach bezkontekstowych materiały do ćwiczeń

Projekt „Matematyka dla ciekawych świata”
spisał: Michał Korch

21 marca 2019

## 1 Gramatyki!

Gramatyka to taki przepis na tworzenie słów z języka. Składa się z reguł, które pozwalają przekształcać poszczególne symbole (robocze) na ciągi symboli (roboczych i liter z alfabetu). Po tym, jak po ciągu przekształceń, zaczynając od roboczego symbolu startowego, przerobimy symbole robocze w ten sposób, że słowo składa się tylko z liter z alfabetu mamy to słowo, które chcemy. Nazwę i ideę można ogarnąć patrząc na gramatykę jakiegoś prawdziwego języka. Na przykład, języka angielskiego. Niech zatem naszym alfabetem będą angielskie słowa. Nasz język to będą poprawne zdania po angielsku, a więc ciągi liter z naszego alfabetu (których rolę pełnią angielskie słowa: a, an, the, black, good, happy, computer, she, it, has, likes, takes, talks, for, to, on, and, but, because). Symbolami roboczymi niech będą nazwy różnych obiektów gramatycznych – dla odróżnienia od liter będziemy je pisać pochyloną czcionką. Symbolem startowym niech będzie *Zdanie*. Uproszczona gramatyka zdań języka angielskiego wygląda tak:

$$\textit{Zdanie} \to \textit{Podmiot}\,\textit{Czasownik}\,\textit{Dopelnienie} \mid \textit{Zdanie}\,\textit{Spojnik}\,\textit{Zdanie}$$

$$\textit{Podmiot} \to \textit{Rodzajnik}\,\textit{Przymiotnik}\,\textit{Rzeczownik} \mid \textit{Rodzajnik}\,\textit{Rzeczownik}$$

$$\textit{Rodzajnik} \to \text{-} \mid \text{a} \mid \text{an} \mid \text{the}$$

$$\textit{Przymiotnik} \to \text{black} \mid \text{good} \mid \text{happy}$$

$$\textit{Rzeczownik} \to \text{girl} \mid \text{cat} \mid \text{computer} \mid \text{she} \mid \text{it}$$

$$\textit{Czasownik} \to \text{has} \mid \text{likes} \mid \text{takes} \mid \text{talks}$$

$$\textit{Dopelnienie} \to \textit{Rodzajnik}\,\textit{Rzeczownik} \mid \textit{Przyimek}\,\textit{Rodzajnik}\,\textit{Rzeczownik}$$

$$\textit{Dopelnienie} \to \textit{Przyslowek} \mid \textit{Dopelnienie}\,\textit{Dopelnienie} \mid \text{-}$$

$$\textit{Przyimek} \to \text{for} \mid \text{to} \mid \text{on}$$

$$\textit{Przyslowek} \to \text{fast} \mid \text{often}$$

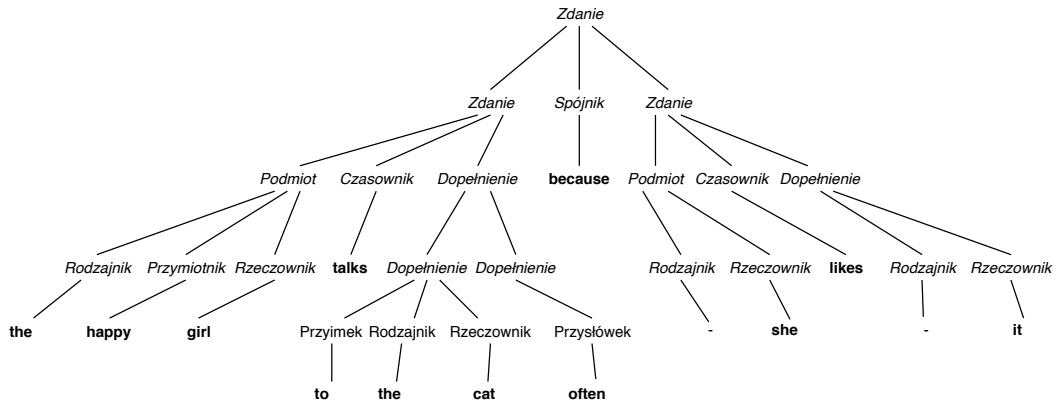
$$\textit{Spojnik} \to \text{and} \mid \text{but} \mid \text{because}$$

Jak widzicie każda reguła mówi na co można zamienić dany symbol roboczy i podaje kilka opcji oddzielonych od siebie pionowymi kreskami. Jest też jeszcze dodatkowy symbol – mówiący, że dany symbol roboczy można po prostu wykreślić zamiast zamieniać go na coś.

Zatem słowo (zdanie) „the happy girl talks to the cat often because she likes it" możemy wygenerować następująco, korzystając z tych reguł:

$$Zadnie \rightarrow$$
$$Zdanie\ Spojnik\ Zdanie \rightarrow$$
$$Podmiot\ Czasownik\ Dopelnienie \text{ because, } Podmiot\ Czasownik\ Dopelnienie \rightarrow$$
$$Rodzajnik\ Przymiotnik\ Rzeczownik \text{ talks } Dopelnienie\ Dopelnienie \text{ because, } Rodzajnik\ Rzeczownik \text{ likes } Rodzajnik\ Rzeczownik \rightarrow$$
$$\text{the happy girl talks } Przyimek\ Rodzajnik\ Rzeczownik\ Przyslowek \text{ because, she likes it} \rightarrow$$
$$\text{the happy girl talks to the cat often because she likes it}$$

Wszystkie te przejścia wygodniej jest przedstawić przy pomocy następującego drzewiastego schematu (zwanego drzewem wyprowadzenia):




## Zadanie 1

Narysuj drzewo wyprowadzenia dla następujących słów (zdań) w powyższej gramatyce

- the cat likes the girl
- she has a computer and it talkes
- the computer talks to a cat

## Zadanie 2

Zaproponujcie dwa dodatkowe zdania po angielsku (słowa w naszym języku), które należą do tego języka (które można wyprowadzić przy pomocy tej gramatyki). Narysuj dla nich drzewa wyprowadzeń. Następnie zaproponuj sensowne zdanie po angielsku, którego nie obejmuje nasza gramatyka.

## Zadanie 3

Poprawcie zdefiniowaną wyżej gramatykę w taki sposób, żeby można było używać przymiotników nie tylko w podmiocie, ale i w dopełnieniu. Możecie „przy okazji" wprowadzić też inne zmiany w tej gramatyce, które wydają Wam się przydatne (np. takie, żeby można było w niej wyprowadzić ostatnie zdanie, które przygotowaliście w poprzednim zadaniu).

Generalnie będziemy rozważać gramatyki bezkontekstowe, w których alfabet składa się z „prawdziwych" liter, np $\{a, b\}$. W takich gramatykach można będzie więc wyprowadzić słowa (czyli ciągi liter). Powiemy, że język jest definiowany przez pewną gramatykę bezkontekstową, jeśli jest zbiorem wszystkich słów, które da się z niej wyprowadzić.

### Zadanie 4

Rozważcie następującą gramatykę nad alfabetem $\{a, b\}$ z symbolem początkowym $P$

$$P \to \mathrm{aa}P\mathrm{aa} \,|\, \mathrm{bb}P\mathrm{bb} \,|\, S$$

$$S \to - \,|\, \mathrm{aa} \,|\, \mathrm{bb}$$

Znajdź 3 słowa, które można wyprowadzić przy pomocy tej gramatyki. Następnie, opisz język zdefiniowany tą gramatyką.

### Zadanie 5

Zaproponujcie gramatykę, która definiuje język złożony ze słów, w których najpierw są litery $a$, a potem litery $b$, ale liczby liter $a$ i $b$ są różne.
*Wskazówka: Niech pierwsza reguła to*

$$P \to W \,|\, M$$

*Następnie niech $W$ generuje słowa, których najpierw są litery $a$, a potem litery $b$, ale liter $a$ jest więcej niż $b$, a $M$ generuje słowa, których najpierw są litery $a$, a potem litery $b$, ale liter $a$ jest mniej niż $b$.*

Odnotujmy jeszcze fakt, że każdy język regularny jest bezkontekstowy. Rzeczywiście każde wyrażenie regularne można bez trudu przerobić na gramatykę bezkontekstową.

### Zadanie 6

Znajdźcie gramatykę bezkontekstową równoważną wyrażeniu regularnemu $(a.a.b + b.a)^*.a.a^*$.
*Wskazówka: Zacznijcie od wyrażenia $a.a.b+b.a$*
*Wskazówka: Stwórzcie nowy symbol początkowy, żeby dostać $(a.a.b + b.a)^*$*
*Wskazówka: Zajmijcie się teraz częścią $a.a^*$*
*Wskazówka: Połącz ostatnie dwa kroki w jedną całość dodając nowy symbol początkowy*

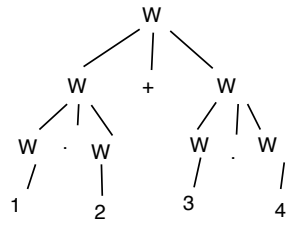
## 2 Zastosowanie gramatyk bezkontekstowych

Gramatyki komputerowe odgrywają zasadnicze znaczenie w procesie rozumienia wyrażeń przez komputer. Jak widzisz na przykładzie języka angielskiego, jeśli dostaniemy zdanie „the happy girl talks to the cat often because she likes it" to znając gramatykę bezkontekstową możemy znaleźć jego drzewo wyprowadzenia. A znając drzewo wyprowadzenia wiemy już dużo więcej – co jest tutaj podmiotem, co jest orzeczeniem itd.

W szczególności, w ten właśnie sposób działają języki programowania. Interpreter rozpoznaje poszczególne elementy napisanego kodu programu tworząc w pamięci drzewo jego wyprowadzenia. Najprostszym przypadkiem mogą być tutaj wyrażenia arytmetyczne (dla uproszczenia załóżmy, że liczby w tych wyrażeniach są jednocyfrowe i jest tylko dodawanie i mnożenie, bez nawiasów). Moglibyśmy takie wyrażenie zdefiniować gramatyką:

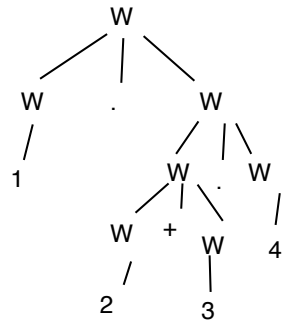
$$W \to W \cdot W \,|\, W + W \,|\, 0 \,|\, 1 \,|\, 2 \,|\, 3 \,|\, 4 \,|\, 5 \,|\, 6 \,|\, 7 \,|\, 8 \,|\, 9$$

Wtedy możemy „zrozumieć" wyrażenie $1 \cdot 2 + 3 \cdot 4$ jako




Co brzmi świetnie, bo umożliwia w ogóle poprawne obliczenie tego wyrażenia $1 \cdot 2 + 3 \cdot 4 = (1 \cdot 2) + (3 \cdot 4) = 2 + 12 = 14$.

Ale, ale, moglibyśmy też nasze wyrażenie wyprowadzić jako




Ale, gdybyśmy tego wyprowadzenia chcieli użyć do konstrukcji gramatyki, skończyłoby się to źle: $1 \cdot 2 + 3 \cdot 4 \neq 1 \cdot ((2 + 3) \cdot 4) = 1 \cdot (5 \cdot 4) = 20$.

Ale da się to naprawić.

### Zadanie 7

Naprawcie tą gramatykę tak, żeby każda metoda wyprowadzenia była zgodna z kolejnością wykonywania działań arytmetycznych i w związku z tym mogła zostać użyta do kalkulacji wyniku.

## 3 Nie każdy język jest bezkontekstowy!

Zauważmy jednak, że nie każdy język jest bezkontekstowy. Przykładem takiego języka jest język słów nad alfabetem $\{a, b, c\}$ takich, że najpierw są litery $a$, potem $b$, potem $c$, ale liter $a$ jest tyle samo co liter $b$, których jest tyle samo, co liter $c$. Aby udowodnić, że tak jest, trzeba posłużyć się dowodem nie wprost. Załóżmy zatem, że istnieje gramatyka, która generuje ten język. Powiedzmy, że gramatyka ta ma $N$ symboli roboczych, a najdłuższa reguła przekształca symbol w $M$ symboli (inaczej mówiąc patrząc na drzewo wyprowadzenia, największe rozgałęzienie to rozgałęzienie na $M$ gałęzi).

Ale w takim razie weźmy słowo $w$, które ma najpierw $M^{N+1}$ liter $a$, potem $M^{N+1}$ liter $b$, potem $M^{N+1}$ liter $c$. I wyobraźmy sobie drzewo wyprowadzenia tego słowa w gramatyce, której istnienie założyliśmy. Skoro nasze drzewo może się rozgałęziać maksymalnie na $M$, musi mieć wysokość większą niż $N$ (bowiem drzewo wysokości $N$ rozgałęziające się w każdym miejscu na $M$ wygeneruje maksymalnie $M^{N+1}$ liter, a my musimy wygenerować $3M^{N+1}$ liter). Ale w takim razie pewna ścieżka (gałąź) tego drzewa musi mieć długość co najmniej $N + 1$. Ale skoro

symboli roboczych użytych w tej gramatyce jest tylko $N$, to któryś symbol musi się na tej ścieżce powtórzyć dwa razy. Mógłbym powiedzieć zatem, że nasze słowo było generowane następująco:

$$P \to \alpha\overline{X}\beta \to \alpha\overline{\gamma X\delta}\beta \to a\ldots ab\ldots bc\ldots c,$$

gdzie $\alpha, \beta, \gamma, \delta$ są też pewnymi ciągami symboli. Zatem moje słowo $w$ składa mogę podzielić na pięć kolejnych części: $w_\alpha, w_\gamma, w_X, w_\delta, w_\beta$, które zostały odpowiednio wygenerowane z ciągów symboli $\alpha, \gamma, X, \delta$ i $\beta$.

Ale, zauważ, że to oznacza, że z symbolu $X$ mogę wygenerować ciąg symboli $\gamma X\delta$. Zatem poniższe wyprowadzenie też jest poprawne:

$$P \to \alpha\overline{X}\beta \to \alpha\overline{\gamma\overline{X}\delta}\beta \to \alpha\overline{\gamma\overline{\gamma X\delta}\delta}\beta \to w_\alpha w_\gamma w_\gamma w_X w_\delta w_\delta w_\beta.$$

Ale zauważ, że niezależnie od tego, jak dokładnie słowo $a\ldots ab\ldots bc\ldots c$ jest podzielone na części $w_\alpha, w_\gamma, w_X, w_\delta, w_\beta$, to nie jest możliwe, że słowo $w_\alpha w_\gamma w_\gamma w_X w_\delta w_\delta w_\beta$ jest postaci $a\ldots ab\ldots bc\ldots c$ o tej samej liczbie liter $a$, $b$ i $c$.

### Zadanie 8

Rozważcie przypadki, żeby zrozumieć dlaczego!

Wobec tego nasza gramatyka pozwala wyprowadzić słowo, które nie jest w języku, który ta gramatyka miała definiować. Mamy sprzeczność, zatem taka gramatyka nie istnieje!

### Zadanie 9

Stosując podobne rozumowanie, udowodnijcie, że język słów nad alfabetem $\{a, b\}$, które mają postać $ww$ konkatenacji (złożenia) dwóch identycznych słów nie jest bezkontekstowy.
*Wskazówka: Zauważcie, że w dowodzie można przyjąć, że część $w_X$ nie może być dłuższa niż $M^{N+1}$.*

## 4 Automaty ze stosem

Wróćmy zatem do koncepcji automatów! Załóżmy, że nasze automaty będą niedeterministyczne. Co więcej dodajmy im nową rzecz. Stos. Stos jest takim miejscem, na który (w ramach przejścia) nasz automat może wrzucać rzeczy (jakieś symbole). Może też zdejmować rzeczy ze stosu (z wierzchu, czyli te najświeższe). Może też widzieć co jest na górze stosu i na tej podstawie decydować o przejściu. Zatem tym razem nasze przejścia będziemy opisywać nie tylko pojedynczą (czytaną przez automat literą), ale tak:

$$a, c \to dd$$

co oznacza, że takie przejście automat może wykonać czytając literę $a$ i widząc literę $c$ na szczycie stosu. W wyniku przejścia na stosie zamiast litery $c$ pojawiają się kolejno dwie litery $d$. Będziemy również dopuszczać przejścia, w których automat nic nie czyta ze słowa, którym się zajmuje (a tylko wykonuje operacje na stosie)

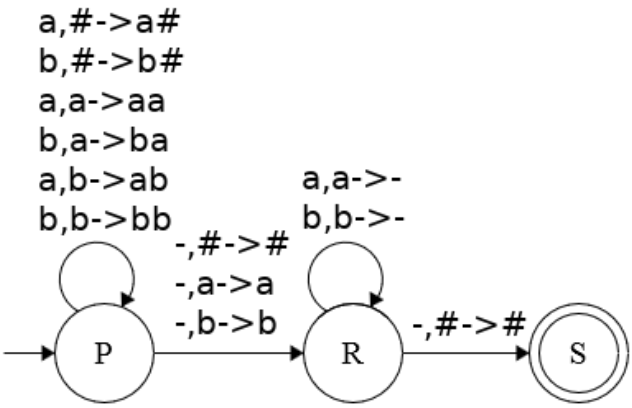
$$-, c \to dd$$

Może być tak, że automat tylko zdejmuje coś ze stosu, a nic tam nie odkłada, co będzie oznaczane, np.

$$a, c \to -$$

(automat czytając $a$ i widząc na stosie $c$, usuwa to $c$ nie zostawiając nic nowego.) Będzie też specjalny symbol #, który zawsze będzie na dnie stosu (i jest jedynym symbolem na stosie na początku działania automatu).

Na przykład to jest automat ze stosem.




### Zadanie 10

Powyższy automat może zaakceptować na przykład słowo *abbaabba*. Jak on to zrobi? Prześledźcie jakie stany przejdzie i co dokładnie będzie na stosie w kolejnych krokach.

### Zadanie 11

Jaki język rozpoznaje rozważany automat? Dlaczego?

### Zadanie 12

Wzbogaćcie powyższy automat w taki sposób, żeby akceptował tylko takie słowa (spośród akceptowanych przez oryginalny automat), które mają liczbę liter $a$ podzielną przez 4.

Okazuje się, że język jest bezkontekstowy wtedy i tylko wtedy, gdy istnieje automat ze stosem który go rozpoznaje. Inaczej mówiąc dla każdej gramatyki istnieje równoważny automat ze stosem. Ale o tym już dowiesz się na wykładzie!

## 5 Zadania dodatkowe

### Zadanie 13

Zastanówcie się nad tym, czego nauczyłeś się w kwestii programowania w Pythonie. Czy moglibyście użyć gramatyki bezkontekstowej do zapisania składni tego języka? Wybierzcie jakiś prosty typ instrukcji w Pythonie, np. test logiczny (to co może się np. pojawić po słowie if) i spiszcie gramatykę, która pozwala wygenerować taki test logiczny. Możemy ograniczyć się tylko do takich testów logicznych, w których podstawowym „budulcem" są liczby (ale nie ma operacji arytmetycznych).

## Zadanie 14

Rzeczywiście, gramatyk bezkontekstowych (w połączeniu z wyrażeniami regularnymi) używa się do opisywania składni języków programowania. Gramatyka składni Pythona jest dostępna pod adresem `https://docs.python.org/3/reference/grammar.html`. Szybko zauważycie, że gramatyka ta jest zapisana nieco inaczej niż zapisujemy ją tu. Spróbujcie zrozumieć, co oznaczają poszczególne oznaczenia.

## Zadanie 15

Przejrzyjcie tę gramatykę Pythona w poszukiwaniu konstrukcji programistycznych, których używaliśmy na ostatnich zajęciach z programowania. Czy zauważyłeś przy okazji coś nowego?

## Zadanie 16

Skonstruujcie gramatykę, która definiuje język złożony ze słów nad alfabetem $\{a, b\}$, który zawiera wszystkie słowa nieparzystej długości oraz te słowa parzystej długości, które są da się przedstawić jako konkatenacja $uv$ (sklejenie) dwóch słów $u$ i $v$, które są nieparzystej długości i mają różne środkowe litery.
*Wskazówka: Niech pierwsza reguła to*

$$P \to A \,|\, B \,|\, AB \,|\, BA$$

*a reguła na $A$ niech generuje dowolne słowo o nieparzystej długości i środkowej literze $A$, zaś reguła na $B$ dowolne słowo o nieparzystej długości i środkowej literze $B$*

## Zadanie 17

Skonstruujcie gramatykę, która definiuje język złożony ze słów nad alfabetem $\{a, b\}$, który zawiera wszystkie słowa, które nie są konkatenacją (sklejeniem) dwóch identycznych słów (nie są postaci $ww$, gdzie $w$ to pewne słowo).
*Wskazówka: Przyjrzyjcie się porządnie słowom generowanym przez gramatykę z poprzedniego zadania. Cała trudność polega na udowodnieniu obserwacji dotyczącej tych słów.*

## Zadanie 18

Skonstruujcie automat ze stosem, który rozpoznaje dokładnie te słowa, które zdefiniowane są przez gramatykę

$$S \to aSb \,|\, A$$

$$A \to bSa \,|\, S \,|\, -$$

*Wskazówka: Użycie symboli roboczych jako symboli stosu. Na początku niech automat wrzuci sobie na stos symbol początkowy. W dowolnym momencie automat może zamienić symbol na stosie według jednej z reguł.*

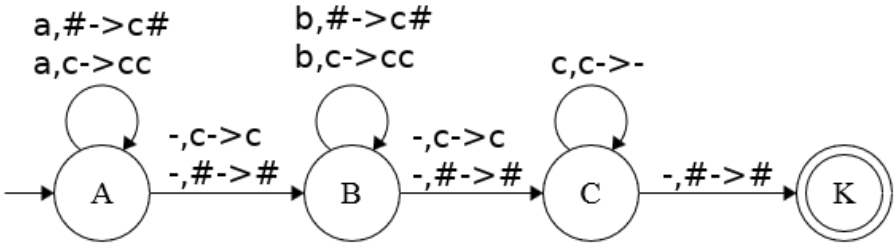
# 6 Zadania domowe

## Zadanie 19 (1 punkt)

Popraw gramatykę przedstawioną na samym początku ćwiczeń, tak aby uniemożliwiała ustawienie rodzajnika (a, the, an) przed zaimkiem (she, it). Przedstaw przykład zdania z języka zdefiniowanego przez tę nową gramatykę łącznie z jego drzewem wyprowadzenia.

## Zadanie 20 (2 punkty)

Prześledź działanie poniższego automatu ze stosem na przykładzie słowa *aaabbccccc*.




Czy ten automat może zaakceptować to słowo? A jeśli tak, jakie są poszczególne stany i zawartość stosu? Opisz język, który jest rozpoznawany przez ten automat.

## Zadanie 21 (3 punkty)

Dopełnieniem pewnego języka $L$ nad danym alfabetem to język złożony ze wszystkich słów nad tym alfabetem, które nie są w $L$. Udowodnij, że nie zawsze dopełnienie języka bezkontekstowego jest językiem bezkontekstowym.

*Wskazówka: Znajdź gramatykę definiującą język, który składa się ze słów, w których kolejno występują najpierw litery a, potem litery b, a następnie litery c, z tym że liter a jest inna liczba niż liter b lub liter b jest inna liczba niż liter a (przyjrzyj się rozwiązaniu zadania 5). Gdy już przygotujesz taką gramatykę, dorzuć do niej reguły generujące słowa, które nie są postaci, w której kolejno występują najpierw litery a, potem litery b, a następnie litery c.*