Programowanie 2009

Lista zadań nr 3

Na zajęcia 17–18 marca 2009

Grupy zasadnicze

Zadanie 1 (1 pkt). Napisz gramatyki bezkontekstowe opisujące następujące języki nad alfabetem $\Sigma = \{0, 1\}$:

1. $\{0^n 1^m 0^n \mid n, m \in \mathbb{N}\}$;
2. $\{0^n 1^n 0^m \mid n, m \in \mathbb{N}\}$;
3. $\{0^n 1^m 0^k \mid n, m, k \in \mathbb{N}, n \leq m\}$;
4. $\{(01)^n 0^{2n} \mid n \in \mathbb{N}\}$;
5. zbiór ciągów zerojedynkowych, w których liczba zer i jedynek jest taka sama;
6. zbiór ciągów zerojedynkowych zawierających dwukrotnie więcej zer niż jedynek.

Zadanie 2 (1 pkt). Gramatyka bezkontekstowa jest *prawostronnie liniowa*, jeżeli wszystkie jej produkcje są postaci $A \to wB$ lub $A \to w$, gdzie $w \in \Sigma^*$ oraz $A, B \in V$.

Napisz gramatyki prawostronnie liniowe opisujące następujące języki nad alfabetem $\Sigma = \{0, 1\}$:

1. $\{0^{2n} \mid n \in \mathbb{N}\}$;
2. $\{0^n 1^m \mid n, m \in \mathbb{N}\}$;
3. zbiór ciągów zerojedynkowych, które nie zawierają trzech kolejnych jedynek;
4. zbiór ciągów zerojedynkowych, w których liczba zer jest parzysta, a jedynek — dowolna;
5. zbiór ciągów zerojedynkowych, w których liczba zer jest parzysta, a jedynek — nieparzysta;
6. zbiór ciągów zerojedynkowych, w których różnica liczby zer i jedynek jest parzysta.

Czy można napisać gramatyki bezkontekstowe posiadające prostsze zbiory produkcji i opisujące powyższe języki?

Zadanie 3 (1 pkt). Zaprojektuj algorytm, który dla podanej gramatyki bezkontekstowej rozstrzyga, czy język przez nią generowany jest niepusty.

Programming 2009

Problem set no. 3

Due March 17, 2009

Basic groups

Problem 1 (1 p). Define context-free grammars describing the following languages over the alphabet $\Sigma = \{0, 1\}$:

1. $\{0^n 1^m 0^n \mid n, m \in \mathbb{N}\}$;
2. $\{0^n 1^n 0^m \mid n, m \in \mathbb{N}\}$;
3. $\{0^n 1^m 0^k \mid n, m, k \in \mathbb{N}, n \leq m\}$;
4. $\{(01)^n 0^{2n} \mid n \in \mathbb{N}\}$;
5. the set of zero-one sequences containing the same number of zeros and ones;
6. the set of zero-one sequences containing twice as much zeros as ones.

Problem 2 (1 p). A context-free grammar is *right-linear*, if all of its productions are of the form $A \to wB$ or $A \to w$, where $w \in \Sigma^*$ and $A, B \in V$.

Define right-linear grammars describing the following languages over the alphabet $\Sigma = \{0, 1\}$:

1. $\{0^{2n} \mid n \in \mathbb{N}\}$;
2. $\{0^n 1^m \mid n, m \in \mathbb{N}\}$;
3. the set of zero-one sequences which do not contain three consecutive ones;
4. the set of zero-one sequences containing an even number of zeros and any number of ones;
5. the set of zero-one sequences containing an even number of zeros and an odd number of ones;
6. the set of such zero-one sequences, that the difference of the numbers of zeros and ones is even.

Is it possible to define context-free grammars describing the aforementioned languages and having simpler sets of productions?

Problem 3 (1 p). Devise an algorithm that decides if a given context-free grammar generates a nonempty language.

Zadanie 4 (1 pkt). Poniższa gramatyka w notacji BNF opisuje fragment składni język C:

Problem 4 (1 p). The BNF grammar below describes a fragment of the C programming language syntax:

⟨declaration⟩ ::= ⟨type⟩⟨declarator⟩ `;`
⟨type⟩ ::= `int`
| `char`
⟨declarator⟩ ::= `*` ⟨declarator⟩
| ⟨declarator⟩ `[` ⟨number⟩ `]`
| ⟨declarator⟩ `(` ⟨type⟩ `)`
| `(` ⟨declarator⟩ `)`
| ⟨name⟩

Wykaż że powyższa gramatyka nie jest jednoznaczna (znajdź napis dla którego istnieją dwa różne drzewa wyprowadzenia; narysuj te drzewa). Konstrukcje `[` ⟨number⟩ `]` oraz `(` ⟨type⟩ `)` można traktować jak operatory postfiksowe zaaplikowane do ⟨declarator⟩. Załóżmy że `*` jest operatorem o niższym priorytecie. Napisz gramatykę jednoznaczną która opisuje ten sam język co powyższa, ale w której sposób rozbioru wyrażeń jest zgodny z określonymi wyżej priorytetami. Załóżmy że w oryginalnej gramatyce uczynimy `*` operatorem postfiksowym. Czy i dlaczego tak zmieniona gramatyka jest jednoznaczna?

Prove that this grammar is ambiguous (find a string which has two different derivation trees; draw these trees). Constructs `[` ⟨number⟩ `]` and `(` ⟨type⟩ `)` can be considered as postfix operators applied to ⟨declarator⟩. Suppose that `*` is an operator of a lower precedence. Write an unambiguous grammar describing the same language as the grammar above which conforms to the stated precedences. Suppose that in the original grammar we change `*` to be postfix. If and why the modified grammar is unambiguous?

Zadanie 5 (1 pkt). Wszystkie operatory w Pascalu są podzielone na cztery grupy pod względem priorytetu (od najmniejszego do największego):

Problem 5 (1 p). Pascal operators are divided into four groups of priorities (from the lowest to the highest):

```
< <= = <> >= > in
      + - or
  / div mod and
       not
```

Operatory binarne łączą w lewo. Dodatkowo unarny operator `-` ma taki sam priorytet, jak binarne operatory addytywne `+`, `-` i `or` i może pojawić się przed liczbą i nawiasem otwierającym, ale nie po operatorze binarnym, np. `-a` oraz `-(1+2)` są poprawne, `a+-b` zaś — nie. Gramatyka bezkontekstowa Pascala nie rozróżnia wyrażeń różnych typów, lecz traktuje wszystkie operatory jednakowo (wyrażenia niepoprawne w sensie typów są odrzucane w późniejszych fazach kompilacji). Napisz gramatykę BNF opisującą wyrażenia w Pascalu.

Dla każdego z poniższych wyrażeń w Pascalu narysuj drzewo wyprowadzenia tego wyrażenia z powyższej gramatyki oraz abstrakcyjne drzewo rozbioru tego wyrażenia.

Binary operators associate to the left. Additionally the unary operator `-` has the same priority as the additive binary operators `+`, `-` and `or`, and can occur in front of a number or the opening parenthesis, but not after a binary operator, *e.g.*, `-a` and `-(1+2)` are legal expressions, but `a+-b` is not. Pascal context-free grammar makes no distinction according to types of expressions but treats all operators in the same way (expressions invalid with respect to types are rejected in later compilation phases). Write a BNF grammar describing Pascal expressions.

For every expression below draw its derivation tree and abstract syntax tree.

1. `i >= 0`
2. `(i >= 0) and not p`
3. `i >= 0 and not p`
4. `(i>= 0) and (x <> y)`

Zadanie 6 (1 pkt). Napisz w notacji BNF jednoznaczną gramatykę opisującą wyrażenia złożone z literałów całkowitoliczbowych, identyfikatorów, nawiasów i następujących operatorów binarnych:

Problem 6 (1 p). Write a BNF grammar describing expressions consisting of integer literals, identifiers, parentheses and the following binary operators:

operator	associativity	priority
`^`	right	4
`*`	left	3
`+`	left	2
`<`	not associative	1
`=`	not associative	1

Zadanie 7 (1 pkt). *Literał zmiennopozycyjny* w Pascalu to napis nad alfabetem

Problem 7 (1 p). *A floating-point literal* in Pascal is a word over the alphabet

$$\Sigma = \{\texttt{0}, \texttt{1}, \texttt{2}, \texttt{3}, \texttt{4}, \texttt{5}, \texttt{6}, \texttt{7}, \texttt{8}, \texttt{9}, \texttt{+}, \texttt{-}, \texttt{e}, \texttt{.}\}$$

następującej postaci: na początku występuje opcjonalnie znak `+` lub `-`, następnie niepusty ciąg cyfr `0`...`9`, potem opcjonalnie część ułamkowa, tj. kropka i niepusty ciąg cyfr i na końcu opcjonalnie wykładnik, tj. znak `e`, opcjonalnie znak `+` lub `-` i niepusty ciąg cyfr. Co najmniej jeden napis spośród części ułamkowej i wykładnika musi wystąpić (inaczej literał będzie tzw. literałem całkowitoliczbowym). Zdefiniuj gramatykę BNF opisującą język literałów zmiennopozycyjnych w Pascalu.

having the following form: there is optionally `+` or `-` sign at the beginning, then a nonempty sequence of digits `0`...`9`, then optionally the fractional part, *i.e.*, a period and a nonempty sequence of digits, then optionally the exponent, *i.e.*, the symbol `e`, optionally `+` or `-` sign and a nonempty sequence of digits. At least one of the fractional part and the exponent should occur (otherwise it would be an integer literal). Define a BNF grammar describing the language of floating-point literals in Pascal.

Zadanie 8 (1 pkt). Napisz w Prologu gramatykę DCG odpowiadającą gramatyce BNF z poprzedniego zadania. Wykorzystaj ją do zdefiniowania predykatu, który zamienia literały zmiennopozycyjne na liczby zmiennopozycyjne.

Problem 8 (1 p). Define in Prolog a DCG grammar corresponding to the BNF grammar from the previous problem. Use it to define a predicate which converts floating-point literals to floating-point numbers.

Grupy rozszerzone

Extended groups

Zadanie 1 (1 pkt). Słowo x nad alfabetem $\{(,)\}$ jest *ciągiem poprawnie rozstawionych nawiasów*, jeśli $\text{balance}(x) = 0$ i $\text{balance}(y) \geq 0$, dla każdego prefiksu y słowa x, gdzie $\text{balance}(\epsilon) = 0$, $\text{balance}(z() = \text{balance}(z) + 1$ oraz $\text{balance}(z)) = \text{balance}(z) - 1$. Dla przykładu `(((()))())` jest ciągiem poprawnie rozstawionych nawiasów, zaś `(()))(()` nim nie jest. Udowodnij, że gramatyka $G = \langle\{(,)\}, \{S\}, S, \{S \to (S), S \to SS, S \to \epsilon\}\rangle$ generuje zbiór wszystkich ciągów poprawnie rozstawionych nawiasów.

Problem 1 (1 p). A word x over the alphabet $\{(,)\}$ is *a sequence of properly distributed parentheses* if $\text{balance}(x) = 0$ and $\text{balance}(y) \geq 0$, for any prefix y of x, where $\text{balance}(\epsilon) = 0$, $\text{balance}(z() = \text{balance}(z) + 1$, and $\text{balance}(z)) = \text{balance}(z) - 1$. For example `(((()))())` is a sequence of properly distributed parentheses, but `(()))(()` is not. Prove that the grammar $G = \langle\{(,)\}, \{S\}, S, \{S \to (S), S \to SS, S \to \epsilon\}\rangle$ generates the set of all properly distributed sequences of parentheses.

Zadanie 2 (1 pkt). Jaki język generuje gramatyka $G = \langle\Sigma, V, S, P\rangle$, gdzie $\Sigma = \{0, 1\}$, $V = \{S\}$, zaś $P = \{S \to 0S1, S \to 0S, S \to \epsilon\}$? Wykaż, że jest ona niejednoznaczna. Zdefiniuj jednoznaczną gramatykę opisującą ten sam język. Udowodnij, że Twoja gramatyka jest jednoznaczna i że generuje ten sam język.

Problem 2 (1 p). What language is generated by the grammar $G = \langle\Sigma, V, S, P\rangle$, where $\Sigma = \{0, 1\}$, $V = \{S\}$, and $P = \{S \to 0S1, S \to 0S, S \to \epsilon\}$? Prove that this grammar is ambiguous. Devise an unambiguous grammar that generates the same language. Prove that your grammar is unambiguous and that it generates the same language.

Zadanie 3 (1 pkt). Niech $|w|_0$ i $|w|_1$ oznaczają liczbę wystąpień odpowiednio zer i jedynek w słowie $w \in \{0,1\}^*$. Napisz gramatykę generującą język

Problem 3 (1 p). Let $|w|_0$ and $|w|_1$ stand for the numbers of occurrences of, respectively, zeros and ones in a word w. Write a grammar generating the language

$$L \quad = \quad \{w \in \{0,1\}^* \ : \ |w|_0 = 2|w|_1 \ \wedge \ \forall v \leq w . |v|_0 \leq 2|v|_1\},$$

gdzie $v \leq w$ oznacza, że v jest prefiksem słowa w. Proszę napisać gramatykę na tyle prostą, by autor rozwiązania mógł w rozsądnym czasie przekonać widownię, że jego gramatyka generuje język, o który nam chodzi. *Wskazówka:* należy wzorować się na zadaniu 1, przy czy teraz definiujemy balance$(\epsilon) = 0$, balance$(z0) =$ balance$(z) - 1$ i balance$(z1) =$ balance$(z) + 2$.

where $v \leq w$ means that v is a prefix of w. Your grammar has to be simple enough so that you can convince the audience in a reasonable time that it generates the language we want. *Hint:* follow the line of Problem 1 now defining balance$(\epsilon) = 0$, balance$(z0) =$ balance$(z) - 1$ and balance$(z1) =$ balance$(z) + 2$.

Zadanie 4 (1 pkt). Rozważmy gramatykę $G = \langle \Sigma, V, X_0, P \rangle$, gdzie

Problem 4 (1 p). Consider the grammar $G = \langle \Sigma, V, X_0, P \rangle$, where

$$\begin{aligned} \Sigma &= \{0,1\}, \\ V &= \{X_0, X_1, X_2\}, \\ P &= \{\ X_0 \to X_0 0,\ X_0 \to X_1 1,\ X_0 \to 0, X_1 \to X_2 0,\ X_1 \to X_0 1,\ X_1 \to 1, X_2 \to X_1 0,\ X_2 \to X_2 1\ \}. \end{aligned}$$

Ciągi zero-jedynkowe traktujemy jako zapisy liczb naturalnych w postaci binarnej, tj. dla $w = w_1 \ldots w_n \in \{0,1\}^*$ przyjmujemy

We treat zero-one sequences as binary representations of natural numbers, *i.e.*, for $w = w_1 \ldots w_n \in \{0,1\}^*$ we take

$$\mathrm{val}(w) \quad = \quad \sum_{i=1}^{n} w_i 2^{n-i}.$$

Udowodnij, że dowolne słowo $w \in \{0,1\}^*$ należy do $\mathfrak{L}(G)$ wtedy i tylko wtedy, gdy val(w) jest podzielne przez 3. *Wskazówka:* pokaż przez indukcję względem długości słowa, że dla dowolnego słowa $w \in \{0,1\}^*$ oraz $i \in \{0,1,2\}$ zachodzi $w \in \mathfrak{L}(G_i)$ wtedy i tylko wtedy, gdy val$(w) \equiv i \bmod 3$, gdzie $G_i = \langle \Sigma, V, X_i, P \rangle$ dla $i = 0,1,2$.

Prove that a word $w \in \{0,1\}^*$ belongs to $\mathfrak{L}(G)$ if and only if val(w) is divisible by 3. *Hint:* prove by induction on the length of a word, that for any word $w \in \{0,1\}^*$ and $i \in \{0,1,2\}$ we have $w \in \mathfrak{L}(G_i)$ if and only if val$(w) \equiv i \bmod 3$, where $G_i = \langle \Sigma, V, X_i, P \rangle$ for $i = 0,1,2$.

Zadanie 5 (1 pkt). Pokaż, że istnieje gramatyka *nieskończenie niejednoznaczna*, tj. taka, że istnieje słowo posiadające nieskończenie wiele drzew wyprowadzenia z tej gramatyki.

Pokaż więcej, że istnieje taka gramatyka opisująca nieskończony język, że dowolne słowo albo nie ma wcale, albo ma nieskończenie wiele drzew wyprowadzenia z tej gramatyki.

Problem 5 (1 p). Show that there exists an *infinitely ambiguous* grammar, *i.e.*, such a grammar, that there exists a word having an infinite number of derivation trees from this grammar.

Prove even more, that there exists such a grammar describing an infinite language, that any word either has no or has infinite number of derivation trees from this grammar.

Zadanie 6 (1 pkt). Niech

Problem 6 (1 p). Let

$$L_1 L_2 \quad = \quad \{uv \mid u \in L_1, v \in L_2\},$$

dla dowolnych języków L_1 i L_2. Udowodnij, że jedynym językiem $L \subseteq \{0,1\}^*$ spełniającym równość

for any languages L_1 i L_2. Prove that the only language $L \subseteq \{0,1\}^*$ satisfying the equality

$$L \quad = \quad \{\epsilon\} \cup \{0\} L \{1\}$$

jest język

is the language

$$L \quad = \quad \{0^n 1^n \mid n \in \mathbb{N}\}.$$

Zadanie 7 (1 pkt). Niech $L_1, L_2 \subseteq \{0,1\}^*$ i niech $\epsilon \notin L_1$. Znajdź język $L \subseteq \{0,1\}^*$ spełniający równość $L = L_1 L \cup L_2$. Wykaż, że jest to jedyne rozwiązanie. Pokaż, że jeżeli $\epsilon \in L_1$, to równość $L = L_1 L \cup L_2$ ma więcej niż jedno rozwiązanie. Co jest najmniejszym rozwiązaniem w tym przypadku? Co jest największym?

Problem 7 (1 p). Let $L_1, L_2 \subseteq \{0,1\}^*$ and $\epsilon \notin L_1$. Find a language $L \subseteq \{0,1\}^*$ satisfying the equality $L = L_1 L \cup L_2$. Prove it is unique. Show that if $\epsilon \in L_1$, then the equality $L = L_1 L \cup L_2$ has more than one solution. What is the smallest solution in this case? What is the biggest one?

Zadanie 8 (1 pkt). Gramatyka jest w *postaci Chomsky'ego*, jeżeli wszystkie jej produkcje są postaci $X \to YZ$ lub $X \to a$, gdzie X, Y i Z są symbolami nieterminalnymi, zaś a jest symbolem terminalnym. Zaprojektuj algorytm, który dla zadanej gramatyki w postaci Chomsky'ego sprawdza, czy podane słowo $a_1 \dots a_n$ należy do języka opisanego tą gramatyką. Wykorzystaj technikę *programowania dynamicznego*. Algorytm powinien wyznaczyć dla każdego podsłowa $a_i \dots a_j$ $(1 \le i \le j \le n)$ zbiór symboli nieterminalnych, z których daje się to słowo wyprowadzić. Jest to łatwe dla podsłów długości 1. Dalej należy postępować indukcyjnie. Słowo należy do języka generowanego przez gramatykę, jeśli symbol startowy gramatyki należy do zbioru symboli, z których dane słowo daje się wyprowadzić.

Problem 8 (1 p). A grammar is in *Chomsky normal form* if all of its productions are of the form $X \to YZ$ or $X \to a$ where X, Y and Z are nonterminal symbols, and a is a terminal symbol. Devise an algorithm which for a given grammar in Chomsky normal form checks if a given word $a_i \dots a_j$ belongs to the language generated by this grammar. Use the *dynamic programming* method. For any given subword $a_i \dots a_j$ $(1 \le i \le j \le n)$ the algorithm should find the set of nonterminals from which this subword can be derived. It is straightforward for subwords of length 1. Then proceed by induction. A word belongs to the language generated by the grammar if the starting symbol of the grammar belongs to the set of symbols from which this word can be derived.

Grupa zaawansowana

Advanced group

Zadanie 1 (2 pkt). Oto fragment gramatyki opisującej deklaracje klas w pewnym języku programowania:

Problem 1 (2 p). Here is a grammar describing class declarations in some programming language:

⟨class declaration⟩	::=	⟨modifiers⟩ ⟨class declarator⟩
⟨modifiers⟩	::=	⟨modifiers⟩ ⟨modifier⟩
	|	⟨empty⟩
⟨modifier⟩	::=	⟨access modifier⟩
	|	⟨inheritance modifier⟩
	|	⟨storage modifier⟩
⟨access modifier⟩	::=	`public` | `private`
⟨inheritance modifier⟩	::=	`abstract` | `final`
⟨storage modifier⟩	::=	`persistent` | `ephemeral`

Właściwy deklarator może być poprzedzony ciągiem przymiotników. Semantyka statyczna języka zawiera dodatkowo regułę mówiącą, że przymiotniki mogą pojawić się w deklaracji klasy w dowolnej kolejności, lecz nie mogą jednocześnie wystąpić dwa przymiotniki należące do tej samej grupy. Powyższa gramatyka nie uwzględnia tego ograniczenia. Uzasadnij, że decyzja twórców języka o przeniesieniu tego ograniczenia z opisu składni do semantyki statycznej jest słuszna, gdyż znacznie upraszcza opis gramatyki języka. Aby to zrozumieć, rozważmy następujący problem. Niech $\{a_i, b_i\}$ dla $i = 1, \dots, n$ będą parami różnych liter i niech $\Sigma = \bigcup_{i=1}^{n} \{a_i, b_i\}$. Udowodnij, że dowolna gramatyka bezkontekstowa generująca język

A proper declarator may be preceded by a sequence of adjectives. Additionally the static semantics of the language contains a rule which states that adjectives can occur in a declaration in any order, but adjectives belonging to the same group cannot occur simultaneously. The grammar above does not impose this restriction. Give an argument, that moving this restriction from syntax description to static semantics is reasonable, since it greatly simplifies the description of the language. To understand this, consider the following problem. Let $\{a_i, b_i\}$ for $i = 1, \dots, n$ be pairwise disjoint letters and let $\Sigma = \bigcup_{i=1}^{n} \{a_i, b_i\}$. Prove that any context-free grammar that generates the language

$$L \quad = \quad \{u \in \Sigma^* \mid |u|_{a_i} + |u|_{b_i} \le 1\}$$

ma co najmniej $2^{\Theta(n)}$ produkcji.

Zadanie 2 (2 pkt). Niech $\Sigma = \{a, b\}$,

$$\begin{aligned} \text{balance}(\epsilon) &= 0 \\ \text{balance}(wa) &= \text{balance}(w) + 1 \\ \text{balance}(wb) &= \text{balance}(w) - 1 \end{aligned}$$

dla $w \in \Sigma^*$ i

$$\begin{aligned} L &= \{w \in \Sigma^* \mid \forall v \leq w.\text{balance}(v) \geq 0\} \\ G_1 &= \langle \Sigma, \{S\}, S, \{S \to \epsilon, S \to aSbS, S \to aS\}\rangle \\ G_2 &= \langle \Sigma, \{N\}, N, \{N \to \epsilon, N \to aNbN\}\rangle \\ G_3 &= \langle \Sigma, \{N, T\}, T, \{N \to \epsilon, N \to aNbN, T \to \epsilon, T \to aNbT, T \to aT\}\rangle \end{aligned}$$

gdzie $v \leq w$ oznacza, że słowo v jest prefiksem słowa w. Udowodnij, że

1. $\mathfrak{L}(G_1) = L$,
2. $\mathfrak{L}(G_3) = L$,
3. $\mathfrak{L}(G_2) = L \cap \{w \in \Sigma^* \mid \text{balance}(w) = 0\}$.
4. G_1 nie jest jednoznaczna,
5. G_2 jest jednoznaczna,
6. G_3 jest jednoznaczna (*wskazówka*: rozważ dwa drzewa wyprowadzenia pewnego słowa; pokaż że ich korzenie muszą być takie same; dalej przez indukcję względem struktury drzew pokaż, że drzewa te są równe).

Z powyższych rozważań wynika następujący wniosek: $\mathfrak{L}(G_1) = \mathfrak{L}(G_3)$ i G_3 jest jednoznaczna.

Opcjonalna fraza `else` w gramatyce

$$\begin{aligned} \langle \textit{instrukcja} \rangle \quad ::= \quad & \texttt{if}\ \langle \textit{wyrażenie logiczne} \rangle\ \texttt{then}\ \langle \textit{instrukcja} \rangle\ \texttt{else}\ \langle \textit{instrukcja} \rangle \\ \mid \quad & \texttt{if}\ \langle \textit{wyrażenie logiczne} \rangle\ \texttt{then}\ \langle \textit{instrukcja} \rangle \\ \mid \quad & \langle \textit{instrukcja prosta} \rangle \end{aligned}$$

powoduje, że gramatyka ta jest niejednoznaczna. Korzystając z powyższego wniosku napisz jednoznaczną gramatykę opisującą ten sam język, w której fraza `else` jest wiązana z najbliższym niezwiązanym `then`.

Zadanie 3 (2 pkt). Rozważmy gramatykę bezkontekstową $G = \langle \Sigma, V, S, P\rangle$. Mówimy, że gramatyka jest *lewostronnie rekurencyjna*, jeżeli dla pewnego symbolu $A \in V$ oraz pewnego słowa $u \in (\Sigma \cup V)^*$ jest $A \overset{+}{\Rightarrow}_G Au$. Pokaż, że dla dowolnego języka bezkontekstowego istnieje gramatyka, która go generuje i która nie jest lewostronnie rekurencyjna. *Wskazówka:* pokaż, że z każdej takiej gramatyki G, że dla dowolnego symbolu $A \in V$ jest $(A \to \epsilon) \notin P$ oraz nieprawda, że $A \overset{+}{\Rightarrow}_G A$, można usunąć produkcje, które prowadzą do rekursji lewostronnej.

has at least $2^{\Theta(n)}$ productions.

Problem 2 (2 p). Let $\Sigma = \{a, b\}$,

$$\begin{aligned} \text{balance}(\epsilon) &= 0 \\ \text{balance}(wa) &= \text{balance}(w) + 1 \\ \text{balance}(wb) &= \text{balance}(w) - 1 \end{aligned}$$

for $w \in \Sigma^*$ and

$$\begin{aligned} L &= \{w \in \Sigma^* \mid \forall v \leq w.\text{balance}(v) \geq 0\} \\ G_1 &= \langle \Sigma, \{S\}, S, \{S \to \epsilon, S \to aSbS, S \to aS\}\rangle \\ G_2 &= \langle \Sigma, \{N\}, N, \{N \to \epsilon, N \to aNbN\}\rangle \\ G_3 &= \langle \Sigma, \{N, T\}, T, \{N \to \epsilon, N \to aNbN, T \to \epsilon, T \to aNbT, T \to aT\}\rangle \end{aligned}$$

where $v \leq w$ means, that the word v is a prefix of w. Prove that

1. $\mathfrak{L}(G_1) = L$,
2. $\mathfrak{L}(G_3) = L$,
3. $\mathfrak{L}(G_2) = L \cap \{w \in \Sigma^* \mid \text{balance}(w) = 0\}$.
4. G_1 is ambiguous,
5. G_2 is unambiguous,
6. G_3 is unambiguous (*hint*: consider two derivation trees of the same word; prove that their roots have to be identical; continue by induction on the structure of the trees to show that the whole trees are identical).

From the considerations above we can conclude that $\mathfrak{L}(G_1) = \mathfrak{L}(G_3)$ and G_3 is unambiguous.

An optional phrase `else` in the grammar

$$\begin{aligned} \langle \textit{instrukcja} \rangle \quad ::= \quad & \texttt{if}\ \langle \textit{wyrażenie logiczne} \rangle\ \texttt{then}\ \langle \textit{instrukcja} \rangle\ \texttt{else}\ \langle \textit{instrukcja} \rangle \\ \mid \quad & \texttt{if}\ \langle \textit{wyrażenie logiczne} \rangle\ \texttt{then}\ \langle \textit{instrukcja} \rangle \\ \mid \quad & \langle \textit{instrukcja prosta} \rangle \end{aligned}$$

makes the grammar ambiguous. Use the conclusion stated above and write an unambiguous grammar describing the same language, in which the `else` phrase matches the closest unmatched `then`.

Problem 3 (2 p). Consider a context-free grammar $G = \langle \Sigma, V, S, P\rangle$. We say that the grammar is *left-recursive* if for some symbol $A \in V$ and a word $u \in (\Sigma \cup V)^*$ we have $A \overset{+}{\Rightarrow}_G Au$. Show that for any context-free language there exists a grammar that generates this language and is not left-recursive. *Hint:* show that we can remove productions that lead to left recursion from any such grammar G, that for any symbol $A \in V$ we have $(A \to \epsilon) \notin P$ and it is not true that $A \overset{+}{\Rightarrow}_G A$.

Zadanie 4 (2 pkt). Wyprowadzenie $u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_n$ z gramatyki bezkontekstowej jest *skrajnie lewe*, jeżeli dla każdego $i < n$ jeśli $u_i = u'_i A u''_i$ oraz $u'_{i+1} = u'_i w u''_i$, gdzie $A \to w$ jest produkcją gramatyki, to u'_i nie zawiera symboli nieterminalnych (innymi słowy przepisaniu zawsze podlega skrajny lewy symbol nieterminalny). Pokaż, że każde słowo ma tyle samo drzew rozbioru i skrajnie lewych wyprowadzeń (innymi słowy gramatyka jest jednoznaczna wtedy i tylko wtedy, gdy każde słowo posiada co najwyżej jedno skrajnie lewe wyprowadzenie).

Problem 4 (2 p). A derivation $u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_n$ from a context-free grammar is *leftmost* if for any $i < n$ if $u_i = u'_i A u''_i$ and $u'_{i+1} = u'_i w u''_i$, where $A \to w$ is a production of the grammar, then u'_i contains no nonterminals (in other words we always rewrite the leftmost nonterminal). Show that any word has the same number of derivation trees and leftmost derivations (in other words a grammar is unambiguous if and only if every word has at most one leftmost derivation).

Zadanie 5 (2 pkt). Czy istnieją języki bezkontekstowe, dla których każda gramatyka je opisująca jest nieskończenie niejednoznaczna (patrz zadanie 5 dla grup rozszerzonych)? Jeśli nie, to czy istnieje algorytm, który dla danej gramatyki G buduje gramatykę G' opisującą ten sam język, jednak taką, że każde słowo posiada tylko skończenie wiele drzew wyprowadzenia?

Problem 5 (2 p). Are there any context-free languages, for which every context-free grammar is infinitely ambiguous (see Problem 5 for extended groups)? If not, does there exist an algorithm which for a given grammar G builds a grammar G' describing the same language but such, that any word has only finitely many derivation trees?

Zadanie 6 (2 pkt). Pokaż, że każdy język bezkontekstowy nie zawierający słowa pustego można opisać gramatyką bezkontekstową w postaci Chomsky'ego (patrz zadanie 6 dla grup rozszerzonych).

Problem 6 (2 p). Prove that any context-free language that does not contain the empty word can be described by a grammar in Chomsky normal form (see Problem 6 for extended groups).

Zadanie 7 (2 pkt). Zaprojektuj algorytm, który dla podanej gramatyki bezkontekstowej rozstrzyga, czy język przez nią generowany jest nieskończony.

Problem 7 (2 p). Devise an algorithm which for a given context-free grammar decides if it generates an infinite language.

Zadanie 8 (2 pkt). Udowodnij, że język bezkontekstowy nad alfabetem $\Sigma = \{a, b, c, d\}$:

Problem 8 (2 p). Prove that the context-free language:

$$L \quad = \quad \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

jest *istotnie niejednoznaczny*, tzn. nie istnieje jednoznaczna gramatyka, która go opisuje.

is *inherently ambiguous*, *i.e.*, there is no unambiguous grammar for this language.

Zadanie 9 (2 pkt). Gramatyka jest *nieograniczenie niejednoznaczna*, jeżeli dla każdej liczby $n \in \mathbb{N}$ istnieje słowo posiadające co najmniej n drzew wyprowadzenia. Język jest *nieograniczenie niejednoznaczny*, jeżeli każda gramatyka, która go opisuje jest nieograniczenie niejednoznaczna. Pokaż, że istnieje taki język.

Problem 9 (2 p). A grammar is *unboundedly ambiguous* if for any number $n \in \mathbb{N}$ there exists a word possessing at least n derivation trees. A language is *unboundedly ambiguous* if any grammar that describes this language is *unboundedly ambiguous*. Show that there exists such a language.

Zadanie 10 (2 pkt). Gramatyka bezkontekstowa $G = \langle \Sigma, V, S, P \rangle$ jest *w postaci operatorowej*, jeżeli każda jej produkcja jest postaci $X \to w$, gdzie $w \neq \epsilon$ i w nie zawiera dwóch sąsiadujących symboli nieterminalnych. Pokaż, że każdy język bezkontekstowy nie zawierający słowa pustego posiada generującą go gramatykę operatorową.

Problem 10 (2 p). A context-free grammar is in *operator form* if all its productions have the form $X \to w$ where $w \neq \epsilon$ and w contains no two consecutive nonterminals. Prove that every context-free language not containing the empty word possesses a context-free grammar in operator form.