Programowanie 2010

Lista zadań nr 7

Na zajęcia 20–22 kwietnia 2010

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 2010

Problem set no. 7

Due April 20, 2010

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

Problem 4 (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 5 (1 pkt). Napisz w notacji BNF jednoznaczną gramatykę opisującą wyrażenia złożone z literałów całkowitoliczbowych, identyfikatorów, nawiasów i następujących operatorów binarnych:

Problem 5 (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 6 (1 pkt). *Literał zmiennopozycyjny* w Pascalu to napis nad alfabetem

Problem 6 (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.