Podstawy Kompilatorów

Laboratorium 5

Uwaga: Do wykonania poniższych zadań związanych z implementacją niezbędny jest kompilator języka C.

Dla środowiska Linux może to być:

- ***Kompilator – GCC, do pobrania na stronie:*** **http://gcc.gnu.org/**

Dla środowiska Windows zalecane jest środowisko cygwin zawierające w swoich pakietach kompilator GCC. Środowisko można pobrać za darmo ze strony: **http://www.cygwin.com/**

Kompilacji można dokonać za pomocą polecenia:

```
gcc parser.c -o parser.exe
```

Dla analizatora o nazwie parser.exe plik z danymi wejściowymi dane.txt będzie przesłany na wejście następująco:

```
scan.exe < dane.txt
```

Zad. 1:

Dana jest gramatyka:

```
S → X Y
S → ( A )
Y → ;
Y → ε
A → 0 X
X → M + Y
M → ε
M → 3 M 1
```

Proszę obliczyć zbiór FIRST(S)

Zad. 2:
Proszę wyeliminować lewostronną rekurencję z gramatyki.

```
S → X Y
X → X a b
X → c d e
Y → f g
```

Zad. 3:
Proszę przeprowadzić lewostronną faktoryzację w gramatyce.

S → X Y
X → a b c
X → a d
Y → f g

Zad. 4:
Proszę napisać w języku C analizator składniowy działający metodą zejść rekurencyjnych dla języka $a^n b^n$, gdzie n>0. Analizator powinien wypisywać OK dla danych wejściowych zawierających jednakową liczbę liter a co liter b, gdzie litery b występują po literach a. Jeśli dane wejściowe są niezgodne z tym formatem analizator powinien wypisać „Syntax error”.

Odpowiedzi do zadań

Zad. 1:

FIRST(S) = FIRST(X) + {‘(‘}
= FIRST (M + Y) + {‘(‘}
= FIRST (M) + {+, ‘(‘}
= { 3, +, ‘(‘ }

Zad. 2:

```
 S → X Y
 X → c d e X2
X2 → a b X2
X2 → ε
 Y → f g
```

Zad. 3:

```
 S → X Y
 X → a X2
X2 → d
X2 → b c
 Y → f g
```

Zad. 4:

Zacznijmy od przygotowania gramatyki dla analizatora składniowego.

```
S → a b
S → a S b
```

Ponieważ symbol S rozpoczyna się w obu produkcjach od terminala „a”, należy przeprowadzić lewostronną faktoryzację. W jej wyniku otrzymujemy gramatykę:

```
S → a S2
S2 → b
S2 → S b
```

Obliczmy zbiory FIRST:
FIRST(S) = {a}
FIRST(S2) = {b, a}

Przygotujmy funkcje pomocnicze:

```
char biezacy;

void sygnalizuj_blad() {
  puts("Syntax Error!");
  exit(-1);
}

char nastepny_symbol() {
     return getchar();
};

void wczytaj(char symbol) {
   if (biezacy == symbol) {
       biezacy = nastepny_symbol();
   } else {
       sygnalizuj_blad();
   }
}
```

Przygotujmy funkcję main zawierającą inicjalizację analizatora składniowego oraz sprawdzenie czy analiza zakończyła się sukcesem:

```
int main() {
  biezacy = nastepny_symbol();
  S();
  if (biezacy != EOF) {
    sygnalizuj_blad();
  } else {
    puts("OK");
  }
  return 0;
}
```

Wykonajmy transformację pierwszej produkcji do kodu w języku C:
S → a S2

```
void S() {
  if (biezacy == 'a') {
    wczytaj('a');
    S2();
  } else {
    sygnalizuj_blad();
  }
}
```

Wykonajmy transformację drugiej produkcji do kodu w języku C:

S2 → b

S2 → S b

```
void S2() {
  if (biezacy == 'b') {
    wczytaj('b');
  } else if (biezacy == 'a') {
    S();
    wczytaj('b');
  } else {
    sygnalizuj_blad();
  }
}
```

Z uwagi na to, że funkcje zawierają wzajemne odwołania do siebie – należy dodać na początku pliku deklaracje funkcji, która występuje w pliku jako druga.

```
void S2();
```

Teraz należy skompilować program za pomocą polecenia:

```
gcc parser.c –o parser.exe
```

Dane można przygotować za pomocą unikowego polecenia echo (należy zwrócić uwagę, by w pliku nie znalazł się symbol nowej linii). Np.:

```
echo –n "aaabbb">dane.txt
```

Następnie można uruchomić program analizatora przesyłając na wejście odpowiednio przygotowane dane wejściowe.

```
./parser.exe<dane.txt
```

Poniżej znajduje się kompletny kod analizatora:

```
#include <stdio.h>
char biezacy;
void S(), S2();

void sygnalizuj_blad() {
  puts("Syntax Error!");
  exit(-1);
}

char nastepny_symbol() {
     return getchar();
};

void wczytaj(char symbol) {
   if (biezacy == symbol) {
       biezacy = nastepny_symbol();
   } else {
       sygnalizuj_blad();
   }
}

void S() {
  if (biezacy == 'a') {
    wczytaj('a');
    S2();
  } else {
    sygnalizuj_blad();
  }
}

void S2() {
  if (biezacy == 'b') {
    wczytaj('b');
  } else if (biezacy == 'a') {
    S();
    wczytaj('b');
  } else {
    sygnalizuj_blad();
  }
}

int main() {
  biezacy = nastepny_symbol();
  S();
  if (biezacy != EOF) {
    sygnalizuj_blad();
  } else {
    puts("OK");
  }
  return 0;
}
```