Лабораторная работа №8
«Множества FIRST для РБНФ»

Скоробогатов С.Ю.

20 августа 2013

1 Цель работы

Целью данной работы является изучение алгоритма построения множеств FIRST для расширенной формы Бэкуса-Наура.

2 Исходные данные

В данной лабораторной работе требуется разработать программу, которая по описанию грамматики, записанному на входном языке в РБНФ, строит множества FIRST для всех нетерминалов грамматика.

В качестве *входного языка* должен выступать язык представления правил грамматики, варианты лексики и синтаксиса которого можно восстановить по примерам из таблицы 1.

3 Задание

Выполнение данной лабораторной работы состоит из следующих этапов:

1. Составление описаний лексической структуры и грамматики входного языка на основе примера из таблицы 1.
2. Разработка лексического анализатора для входного языка.
3. Разработка синтаксического анализатора для входного языка методом рекурсивного спуска.
4. Реализация алгоритма вычисления множества FIRST для всех нетерминальных символов грамматики.

Отметим, что парсер входного языка должен выдавать сообщения об обнаруженных ошибках, включающие координаты ошибки. Восстановление при ошибках реализовывать не нужно.

В качестве языков реализации разрешается использовать C++, Java, Go, Ruby или Python.

Таблица 1: Варианты входного языка в примерах описаний грамматик

1

```
non-terminal E, T, F;
terminal '+', '-', '*', '/',
  '(', ')', n;

E ::= T ( ('+' | '-') T )*;
T ::= F ( ('*' | '/') F )*;
F ::= n | '-' F | '(' E ')';
```

2

```
$NTERM T F E
$TERM  "+"  "-"  "*"  "/"
$TERM  "("  ")"  "n"
$RULE  E = T { ("+" | "-") T }
$RULE  T = F { ("*" | "/") F }
$RULE  F = "n" | "-" F | "(" E ")"
```

3

```
(E) = (T) [ [\+|-] (T) ]*.
(T) = (F) [ [\*|/] ]*.
(F) = n | -(F) | \( (E) \).
```

4

```
E  ( T {("+","-") T} )
T  ( F {("*","/") F} )
F  ( "n", "-" F,
     "(" E ")" )
```

5

```
tokens <plus>, <minus>,
       <star>, <slash>,
       <lparen>, <rparen>, <n>.
<E> is <T> repeat(
             alt(
               <plus>, <minus>
             ) <T>
           ).
<T> is <F> repeat(
             alt(
               <star>, <slash>
             ) <F>
           ).
<F> is alt(
         <n>,
         <minus> <F>,
         <lparen> <E> <rparen>
       ).
```

6

```
[ E  : T [ ["+" : "-"] T ]* ]
[ T  : F [ ["*" : "/"] F ]* ]
[ F  : "n"
     : "-" F
     : "(" E ")"
]
```

7

```
<E    <T {<
             <+>
             <->
           > T}>>
<T    <F {<
             <*>
             </>
           > F}>>
<F    <n>
      <- F>
      <( E )>>
```

8

```
E  -> T 'repeat
            ("+" 'or "-") T
        'end
     'end
T  -> F 'repeat
            ("*" 'or "/") F
        'end
     'end
F  -> "n" 'or
      "-" F 'or
      "(" E ")" 'end
```