Лабораторная работа №5
"РБНФ в грамматику" (Вариант 5)
по курсу Теория Формальных Языков

Златовратский Иван

Март 2023

Содержание

1 Условие лабораторной работы

1.1 Базовое условие

1. Необходимо предложить грамматику описания РБНФ.
2. Параметризованными токенами грамматики могут выступать, например:
   - символы начала и конца нетерминала, а также начала и конца итерации, опционального вхождения и альтернативы;
   - символы разделения левой и правой части правила и правил друг от друга;
   - способ объявления стартового нетерминала и обозначение для ε.
3. Из входа читается описание РБНФ в заданном синтаксисе.
4. Требуется построить классическую КС-грамматику, ему эквивалентную.
5. Имена для новых нетерминалов должны генерироваться так, чтобы был понятен их смысл, и откуда они берутся.

1.2 Дополнительные задания

- (+2 балла) Язык нетерминальных и терминальных символов (в форме регулярных выражений)
- (+1–3 балла) Описание грамматики, документацию и пример работы программы оформить в latex. Число баллов зависит от качества оформления.
- (+3 балла) Параметры синтаксиса выходной грамматики также читать из (другого) файла.
- (+3 балла) Язык Lua

2 Входные данные

Входные данные задаются в четырех файлах:

- syntaxRBNF.txt
- syntaxCF.txt
- regex.txt
- RBNF.txt

2.1 Синтаксис входной РБНФ грамматики

Синтаксис входной РБНФ грамматики задается в файле "syntaxRBNF.txt".

Формат файла

- В файле уже заданы названия токенов и их значения по умолчанию.
- Пользователю разрешено изменять только значения токенов, которые задаются в кавычках и количество пробелов вокруг знака равенства, а также, порядок токенов.

Значения токенов

Токен	Опиание
nonTerminalStart = "_"	токен начала нетерминала
nonTerminalEnd = "_"	токен конца нетерминала
arrow = ":="	токен, стоящий между именем нетерминала и его значением
iterStart = "{"	токен начала итерации
iterEnd = "}"	токен конца итерации
optionalStart = "["	начала опционального вхождения
optionalEnd = "]"	токен конца опционального вхождения
necessaryStart = "("	токен начала обязательного вхождения
necessaryEnd = ")"	токен конца обязательного вхождения
alternative = "|"	токен, разделяющий альтернативные правила
epsilon = "eps"	токен пустого слова

Таблица 1: Файл "syntaxRBNF.txt"

Формат токенов

- Количество символов в токене должно быть от одного или больше.
- Значения токенов не должны повторяться (могут повторяться только nonTerminalStart и nonTerminalEnd)
- Из ASCII с 33 по 126 символы:
  - Разрешены: { } [ ] ( ) / | ! ? “ ‘ , . : ; _ < = > & # @ 0-9 A-Z a-z
  - Запрещены: % $ ^ * + -

2.2 Синтаксис выходной КС грамматики

Синтаксис выходной грамматики задается в файле "syntaxCF.txt".

Формат файла

- В файле уже заданы названия токенов и их значения по умолчанию.
- Пользователю разрешено изменять только значения токенов, которые задаются в кавычках и количество пробелов вокруг знака равенства, а также, порядок токенов.

Значения токенов

Токен	Опиание
nonTerminalStart = "["	токен начала нетерминала
nonTerminalEnd = "]"	токен конца нетерминала
arrow = >"	токен, стоящий между именем нетерминала и его значением
alternative = "|"	токен, разделяющий альтернативные правила
epsilon = "$"	токен пустого слова

Таблица 2: Файл "syntaxCF.txt"

Формат токенов

- Количество символов в токене должно быть от одного или больше.
- Значения токенов могут повторяться.
- Из ASCII с 33 по 126 символы:
  - Разрешены: { } [ ] ( ) / | ! ? “ ‘ , . : ; _ < = > & # @ 0-9 A-Z a-z $ ^ * + -
  - Запрещены: %

2.3 Ввод регулярного выражения

Регулярное выражение задается в файле "syntaxCF.txt".

Формат файла

- В файле уже заданы названия токенов и их значения по умолчанию.
- Пользователю разрешено изменять только значения токенов, которые задаются в кавычках и количество пробелов вокруг знака равенства, а также
- Порядок токенов изменять нельзя.

Значения токенов

Токен	Опиание
Nonterminal ::= [A-Z|0|1][A-z]*	Регулярное выражение для нетерминалов
Terminal ::= [a-z]*	Регулярное выражение для терминалов

Таблица 3: Файл "regex.txt"

Формат токенов

- Классические регулярные выражения.
- Могут быть одинаковыми для нетерминалов и терминалов.

2.4 Ввод РБНФ грамматики

РБНФ грамматика задается в файле "syntaxCF.txt".

Формат файла

Пользователь должен ввести РБНФ грамматику в соответствии с синтаксисом и терминалы, нетерминалы в ней, удовлетворяющие регулярным выражениям.

Пример РБНФ грамматики

```
_DightNotNull_ := 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
_Dight_        := 0 | _DightNotNull_
_Natural_      := (_DightNotNull_) {_Dight_}
_Int_          := 0 | [minus] _Natural_
```

3 Выходные данные

3.1 Возможные ошибки

Если входные данные неверны, то программа выведет ошибку.

Варианты ошибок

- 'Error (syntax): Not right arrow: _A_ :=== _A_b'
- 'Error (syntax):] <- not found in : (_DightNotNull_)[_Dight_'
- 'Error (syntax): Not equal number of { and } in: (_DightNotNull_){_Dight_}}'
- 'Error (syntax): No expression in brackets'
- 'Error (regex): '#a' <- Left nonterm doesn't match to it's regex'
- 'Error (regex): '#b' <- Right nonterm doesn't match to it's regex'
- 'Error (regex): 'A' <- Term doesn't match to it's regex'
- 'Error (regex): 'B' <- Term doesn't match to it's regex'

3.2 Вывод при верных данных

Если входные данные верны, программа выведет этапы работы парсера, где можно увидеть процесс замены вхождений на новые нетерминалы, и после этого - КС грамматику.

Простой пример

Синтаксис РБНФ и КС грамматик, а также, регулярные выражения заданы по умолчанию.

RBNF.txt:

```
_R_ := {_S_}[[_S_](_S_)]
```

Результет работы программы:

```
Parser:
    _R_ := {_S_}[[_S_](_S_)]

        {_S_}[[_S_](_S_)]       change: {_S_} -> _Nt1_
        _Nt1_[[_S_](_S_)]

        _Nt1_[[_S_](_S_)]       change: [_S_] -> _Nt2_
        _Nt1_[_Nt2_(_S_)]

        _Nt1_[_Nt2_(_S_)]       change: (_S_) -> _S_
        _Nt1_[_Nt2__S_]

        _Nt1_[_Nt2__S_]       change: [_Nt2__S_] -> _Nt3_
        _Nt1__Nt3_


CF grammar:
    [Nt1] -> [S][Nt1] | $
    [Nt2] -> [S] | $
    [Nt3] -> [Nt2][S] | $
    [R] -> [Nt1][Nt3]
```

Пример "Integer"

Синтаксис РБНФ и КС грамматик, а также, регулярные выражения заданы по умолчанию.

RBNF.txt:

```
_DightNotNull_ := 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9
_Dight_        := 0 | _DightNotNull_
_Natural_      := (_DightNotNull_) {_Dight_}
_Int_          := 0 | [minus] _Natural_
```

Результет работы программы:

```
Parser:
    _DightNotNull_ := 1|2|3|4|5|6|7|8|9

    _Dight_ := 0|_DightNotNull_

    _Natural_ := (_DightNotNull_){_Dight_}

        (_DightNot0_){_Dight_}     change: {_Dight_} -> _Nt1_
        (_DightNot0_)_Nt1_

        (_DightNot0_)_Nt1_    change: (_DightNot0_) -> _DightNot0_
        _DightNot0__Nt1_

    _Int_ := 0|[minus]_Natural_

        0|[minus]_Natural_     change: [minus] -> _Nt2_
        0|_Nt2__Natural_

CF grammar:
    [DightNot0] -> 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
    [Dight] -> 0 | [DightNot0]
    [Nt1] -> [Dight][Nt1] | $
    [Natural] -> [DightNot0][Nt1]
    [Nt2] -> minus | $
    [Int] -> 0 | [Nt2][Natural]
```

Пример грамматики с очень нестандартным синтаксисом

syntaxRBNF.txt:

```
nonTerminalStart = "<"
nonTerminalEnd = ">"
arrow = ":===="
iterStart = "IS"
iterEnd = "IE"
optionalStart = "OS"
optionalEnd = "OE"
necessaryStart = "NS"
necessaryEnd = "NE"
alternative = "|||"
epsilon = "eps"
```

syntaxCF.txt:

```
nonTerminalStart = "-+"
nonTerminalEnd = "+-"
arrow = "->^"
alternative = "*"
epsilon = "$"
```

regex.txt:

```
Nonterminal ::= [A-Z|0|1][A-z]*
Terminal ::= [a-z]*
```

RBNF.txt:

```
<R> :===== OS<S>OEIS<S>IENS<S>NE ||| eps
<Q> :===== NS<X>NENSaNEb
```

Результет работы программы:

```
Parser:
    <R> := OS<S>OEIS<S>IENS<S>NE|||eps

        OS<S>OEIS<S>IENS<S>NE|||eps       change: IS<S>IE -> <Nt1>
        OS<S>OE<Nt1>NS<S>NE|||eps

        OS<S>OE<Nt1>NS<S>NE|||eps       change: OS<S>OE -> <Nt2>
        <Nt2><Nt1>NS<S>NE|||eps

        <Nt2><Nt1>NS<S>NE|||eps       change: NS<S>NE -> <S>
        <Nt2><Nt1><S>|||eps
```

<Q> := NS<X>NENSaNEb

NS<X>NENSaNEb change: NS<X>NE –> <X>
<X>NSaNEb

<X>NSaNEb change: NSaNE –> a
<X>ab

CF grammar:

—+Nt1+— –>^ —+S+—+Nt1+— * $
—+Nt2+— –>^ —+S+— * $
—+R+— –>^ —+Nt2+—+Nt1+—+S+— * $
—+Q+— –>^ —+X+—ab

4 Запуск программы

Все входные данные находятся в папке "/input":

```
/input/syntaxRBNF.txt
/input/syntaxCF.txt
/input/regex.txt
/input/RBNF.txt
```

Программа находится в файле "/lua/main.lua"

Запуск прграммы осуществяется окмандой:

```
lua54 main.lua
```