Генерация анализатора по грамматике в расширенной БНФ

Выполнил:
Орлов Илья, 444 группа

Научный руководитель:
Григорьев Семён Вячеславович

Рецензент:
Полозов Сергей Викторович

Грамматики в расширенной форме Бэкуса-Наура (EBNF)

- Регулярные выражения в правых частях правил
- Понятное и удобное описание грамматики
- Генерация синтаксических анализаторов по грамматике в EBNF форме

Генераторы синтаксических анализаторов по грамматике в EBNF

- Поддерживают EBNF:
  - Удаление регулярных выражений
  - Преобразование исходной грамматики
  - Вывод не в терминах исходной грамматики, заданной пользователем
  - Увеличение таблиц состояний и переходов автомата анализатора

Постановка задачи

- Цель
  - Разработка генератора синтаксических анализаторов, сохраняющего EBNF форму
- Задачи
  - Разработка алгоритма
  - Реализация алгоритма в рамках проекта YaccConstructor
  - Апробация решения

Существующие алгоритмы

- LR
  - Masataka S., Ikuo N. - “A Simple Realization of LR-Parsers for Regular Right Part Grammar”
  - Paul Walton Purdom Jr., Cynthia A.Brown - “Parsing extended LR(k) grammars”
  - Поддерживают не полный класс контекстно-свободных грамматик
- GLR
- RNGLR
  - Неоднозначные грамматики

Существующие технологии

- LR
  - Yacc, Bison
  - RNGLR (реализация в YaccConstructor)
    - Преобразуют исходную грамматику
  - RNGLR.EBNF (реализация в YaccConstructor)
    - Используются специальные множества меток на стеке при свертке
    - Затраты на время и память при поиске меток в множестве
- LL
  - ANTLR

Алгоритм

- За основу взят RNGLR.EBNF
- Отличие:
  - Нет специальных меток для определения начала нового правила при свертке
  - Работа со стеком: распознавание цепочки при помощи инвертированного автомата
- Преимущество:
  - Нет дополнительной работы по вычислению специальных меток

Построение парсера

- Цепочка, порождаемая правой частью продукции, не имеет фиксированной длины
- Решение – распознавание цепочки при помощи реверсированного ДКА
  - Создание эквивалентных НКА
    - Для создания реверсированного ДКА
    - Для подсчета семантики
  - Создание реверсированных НКА
  - Преобразование реверсированного НКА в ДКА
    - Для работы со стеком при свертке

Работа интерпретатора

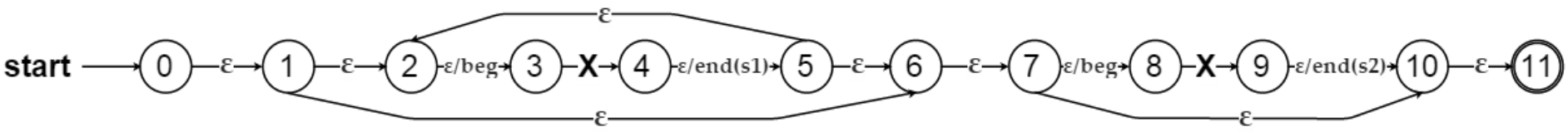
- Распознавание цепочки при свертке при помощи ДКА
- Возникает несколько возможных вариантов принятия строки автоматом
  - Разрешается при помощи средств GLR (GSS)
- НКА используется для подсчета семантики по распознанной части
  - Интерпретатор НКА
  - Добавление промежуточных узлов в лес разбора (SPPF)

Пример

start: (X{s1})* (X{s2})?

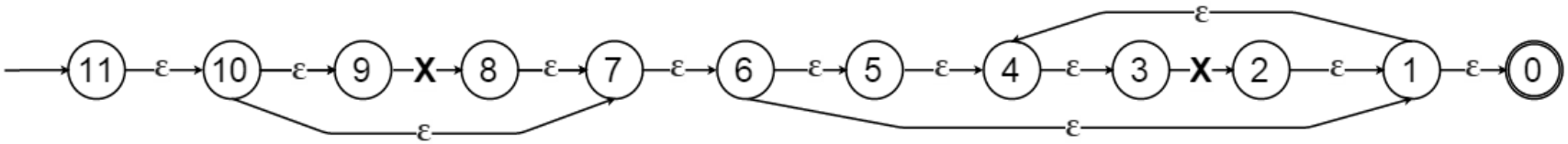
Вход: X X X

НКА



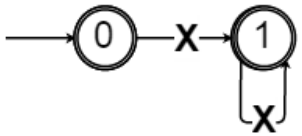
⇩

Реверсированный НКА



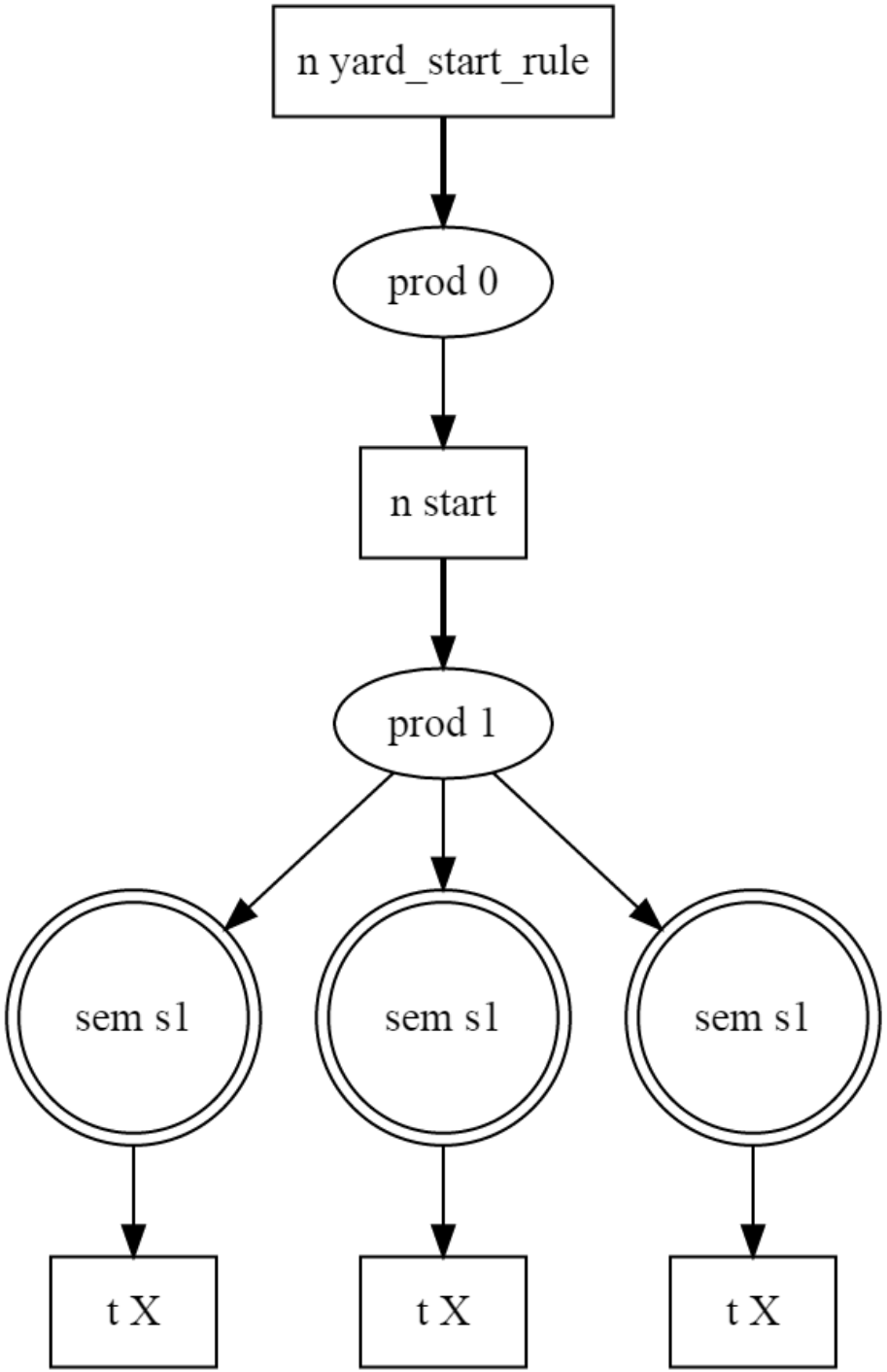
⇩

ДКА



Пример

- Несколько способов распознавания строки НКА
- Необходима жадность автомата



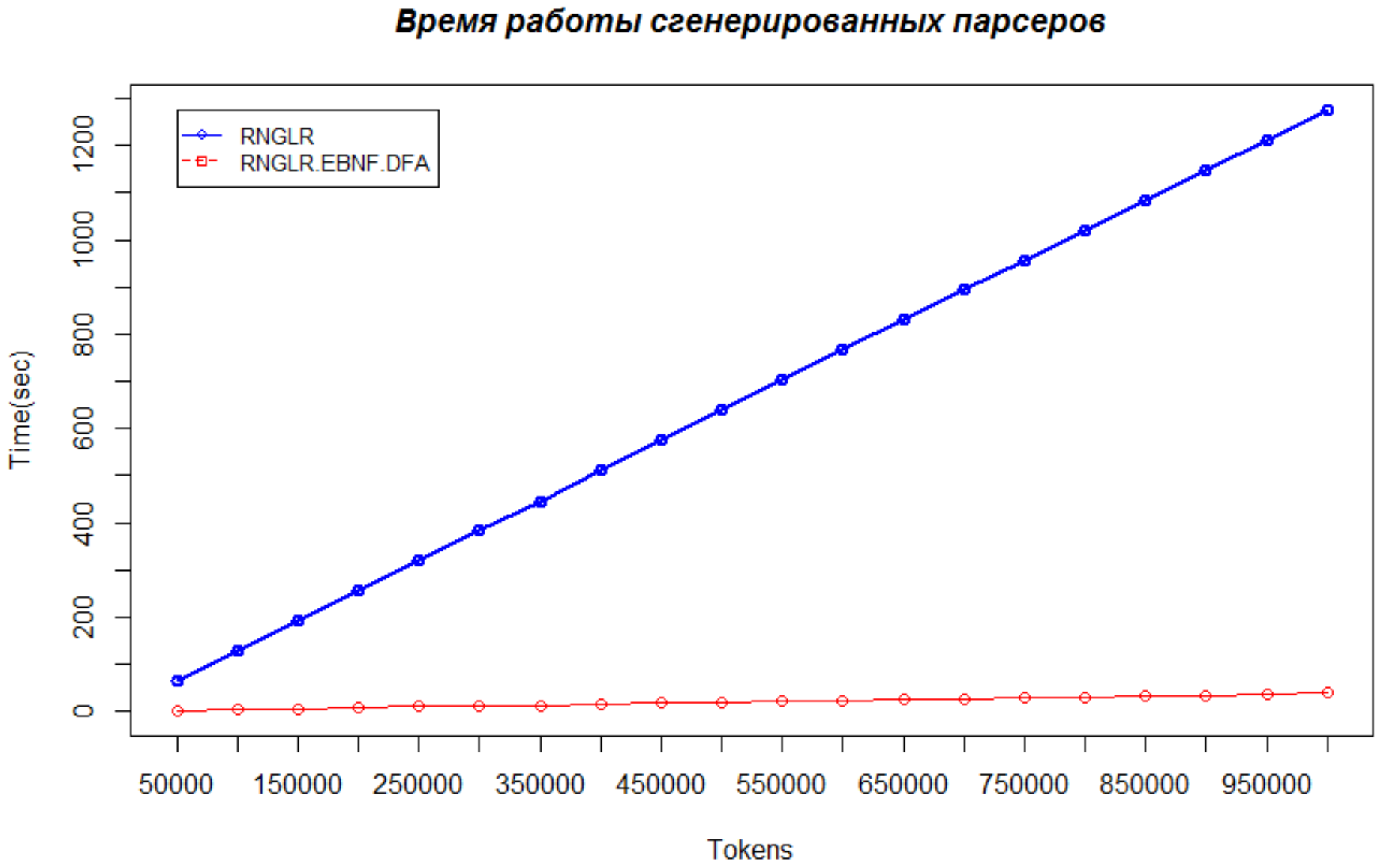
Апробация

- Сравнение по скорости работы интерпретатора
  - Грамматика вида:

    start: X* X X?

  - Входные строки: 50000 – 1000000 символов
  - Алгоритмы:
    - RNGLR
    - RNGLR.EBNF
    - RNGLR.EBNF.DFA

Апробация


Время работы сгенерированных парсеров
RNGLR
RNGLR.EBNF.DFA
Time(sec)
0
200
400
600
800
1000
1200
50000
150000
250000
350000
450000
550000
650000
750000
850000
950000
Tokens

Результаты

- Разработан алгоритм
  - Работа с грамматиками в EBNF без преобразования
- Алгоритм реализован в рамках проекта YaccConstructor
- Проведена апробация решения