НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Факультет компьютерных наук
Департамент программной инженерии
Курсовая работа
Программа взаимного преобразования конечных автоматов и регулярных выражений

Выполнил студент группы 101 ПИ

Бадретдинов Тимур Ринатович

Научный руководитель:

Преподаватель департамента программной инженерии, кандидат физико-математических наук

Каленкова Анна Алексеевна

ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Предметная область

Программа является результатом работы в области формальных грамматик и лексического анализа.

Постановка задачи

Создать программу, которая позволит строить конечные автоматы на основе регулярного выражения, проводить дальнейшие преобразования и получать регулярные выражения на основе любого из полученных автоматов.

LR(1) грамматика

LR

LR — это подвид контекстно-свободной грамматики.
LR в данном случае значит, что анализатор грамматики такого вида читает строку слева (**L**eft) направо, стараясь обработать как можно больше символов справа (**R**ightmost).

LR(1)

Некоторое число k в записи LR(k) значит, что анализатор такой грамматики будет считывать k символов для принятия решения. То есть, LR(1) анализатор использует только 1 символ.

Описание библиотеки

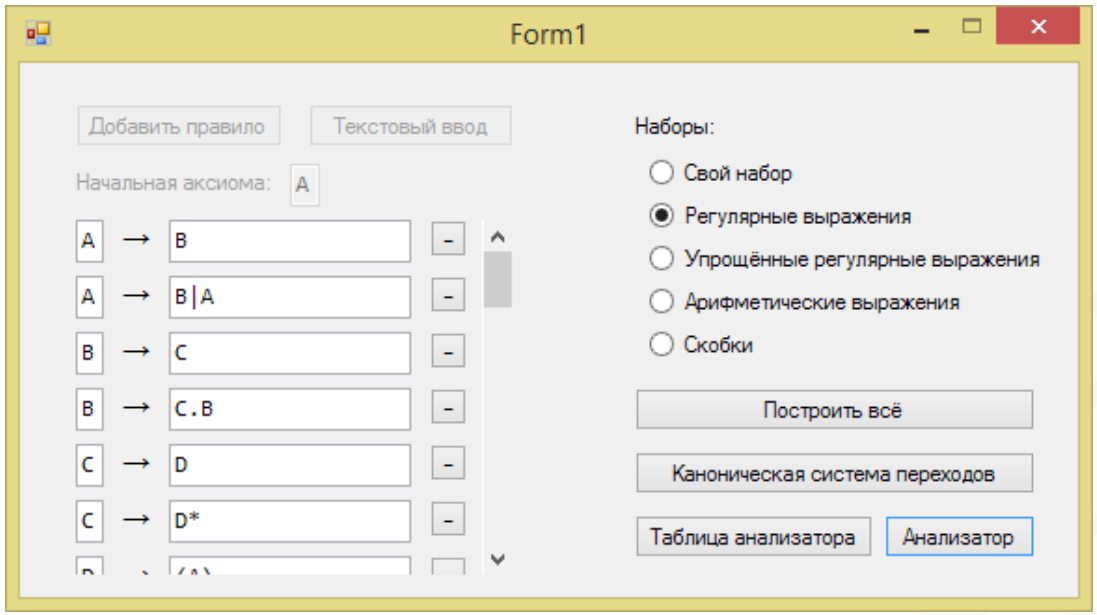
Библиотека предоставляет возможности LR(1) анализатора. С помощью неё можно проверить строку на принадлежность той или иной грамматике. В случае, если строка принадлежит ей, библиотека возвращает в качестве результата своей работы дерево разбора.

Программа для работы с библиотекой

Описание программы

Программа предоставляет все возможности для работы с библиотекой: создание (импорт) собственных грамматик (с проверкой каждой грамматики на принадлежность к классу LR(1), проверка строки на выбранную грамматику, отображение дерева разбора строки.

Главное окно программы



Преобразование регулярного выражения в недетерминированный конечный автомат

Описание алгоритма

На вход алгоритм получает дерево разбора регулярного выражения. Для каждой вершины алгоритм «склеивает» дочерние вершины. Алгоритм проходится по дереву поиском в глубину. Алгоритм заканчивает свою работу в корне дерева.

Вычислительная сложность

Алгоритм выполняется за полиномиальное время.

Преобразование регулярного выражения в детерминированный конечный автомат

Описание алгоритма

На вход алгоритм получает дерево разбора регулярного выражения. Каждому листу дерева присваивается число-позиция. Затем для каждого листа вычисляются функции, значения которых зависят от позиции этих листьев в дереве. В том числе, одна из функций задана на множестве позиций. Полученное множество значений этой функции есть множество позиций полученных состояний.

Вычислительная сложность

Алгоритм выполняется за экспоненциальная время.

Вначале Q и D пусты. Выполнить шаги 1-6:

(1) Построить синтаксическое дерево для пополненного регулярного выражения $(r)\#$.

(2) Обходя синтаксическое дерево, вычислить значения функций $nullable$, $firstpos$, $lastpos$ и $followpos$.

(3) Определить $q_0 = firstpos(root)$, где $root$ – корень синтаксического дерева.

(4) Добавить q_0 в Q как непомеченное состояние.

(5) Выполнить следующую процедуру:

```
while (в Q есть непомеченное состояние R){
  пометить R;
  for (каждого входного символа a ∈ T, такого, что
       в R имеется позиция, которой соответствует a){
    пусть символ a в R соответствует позициям
    p_1, ..., p_n, и пусть S = ⋃_{1≤i≤n} followpos(p_i);
    if (S ≠ ∅){
      if (S ∉ Q)
        добавить S в Q как непомеченное состояние;
      определить D(R, a) = S;
    }
  }
}
```

(6) Определить F как множество всех состояний из Q, содержащих позиции, связанные с символом $\#$.

Преобразование недетерминированного конечного автомата в детерминированный конечный автомат

Описание алгоритма

На вход алгоритм получает недетерминированный конечный автомат. В основе работы алгоритма лежат две функции: e-closure и move. Первая находит множество состояний, в которые можно перейти по пустому символу. Вторая получает множество состояний, в которые можно перейти по заданному символу. Алгоритм помечает обработанные состояния. Алгоритм завершает свою работу, когда во входном автомате нету непомеченных состояний.

Вычислительная сложность

Алгоритм выполняется за экспоненциальное время.

Вначале Q' и D' пусты. Выполнить шаги 1-4:

(1) Определить $q'_0 = e\text{-}closure(\{q_0\})$.

(2) Добавить q'_0 в Q' как непомеченное состояние.

(3) Выполнить следующую процедуру:

```
while (в Q' есть непомеченное состояние R){
  пометить R;
  for (каждого входного символа a ∈ T){
    S = e-closure(move(R, a));
    if (S ≠ ∅){
      if (S ∉ Q')
        добавить S в Q' как непомеченное состояние;
      определить D'(R, a) = S;

    }
  }
}
```

(4) Определить $F' = \{S | S \in Q', S \cap F \neq \varnothing\}$.

Минимизация детерминированного конечного автомата

Описание алгоритма

На вход программа получает детерминированный конечный автомат. Алгоритм разделяет множество всех состояний на 2 группы. Затем, группы разделяются на более мелкие до тех пор, пока это возможно.

Вычислительная сложность

Алгоритм выполняется за полиномиальное время.

(1) Построить начальное разбиение Π множества состояний из двух групп: заключительные состояния Q и остальные $Q-F$, т.е. $\Pi = \{F, Q-F\}$.

(2) Применить к Π следующую процедуру и получить новое разбиение Π_{new}:

for (каждой группы G в Π){

разбить G на подгруппы так, чтобы

состояния s и t из G оказались

в одной подгруппе тогда и только тогда,

когда для каждого входного символа a

состояния s и t имеют переходы по a

в состояния из одной и той же группы в Π;

заменить G в Π_{new} на множество всех

полученных подгрупп;

}

(3) Если $\Pi_{\text{new}} = \Pi$, полагаем $\Pi_{\text{res}} = \Pi$ и переходим к шагу 4, иначе повторяем шаг 2 с $\Pi := \Pi_{\text{new}}$.

(4) Пусть $\Pi_{\text{res}} = \{G_1, \ldots, G_n\}$. Определим:

$Q' = \{G_1, \ldots, G_n\}$;

$q'_0 = G$, где группа $G \in Q'$ такова, что $q_0 \in G$;

$F' = \{G | G \in Q' \text{ и } G \cap F \neq \varnothing\}$;

$D'(p', a) = q'$, если $D(p, a) = q$, где $p \in p'$ и $q \in q'$.

Таким образом, каждая группа в Π_{res} становится состоянием нового автомата M'. Если группа содержит начальное состояние автомата M, эта группа становится начальным состоянием автомата M'. Если группа содержит заключительное состояние M, она становится заключительным состоянием M'. Отметим, что каждая группа Π_{res} либо состоит только из состояний из F, либо не имеет состояний из F. Переходы определяются очевидным образом.

(5) Если M' имеет "мертвое" состояние, т.е. состояние, которое не является допускающим и из которого нет путей в допускающие, удалить его и связанные с ним переходы из M'. Удалить из M' также все состояния, недостижимые из начального.

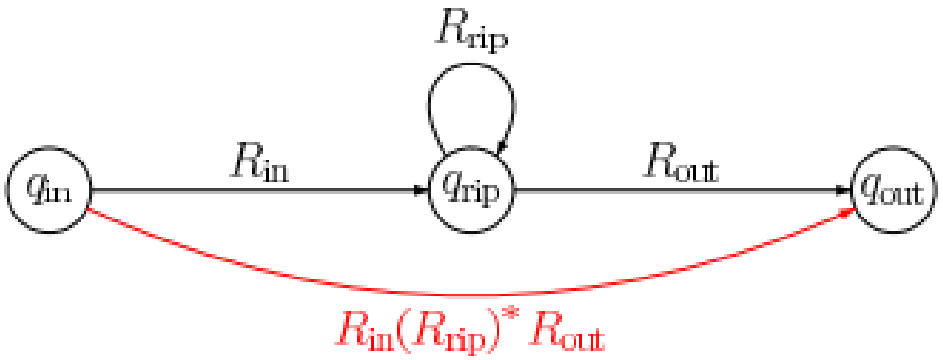
Получение регулярного выражения из конечного автомата

Подзаголовок

На вход алгоритм получает конечный автомат (детерминированный или недетерминированный). Алгоритм поочерёдно удаляет из автомата состояния, обновляя все переходы, до тех пор, пока не останется только два состояния. Набор оставшихся переходов и будет составлять регулярное выражение.

Вычислительная сложность

Алгоритм выполняется за экспоненциальное время



Отрисовка конечных автоматов

План

1. Перевод автомата в формат Dot
2. Использование библиотеки GraphViz для получения разметки графика.
3. Отрисовка рисунка автомата на основе полученной разметки.

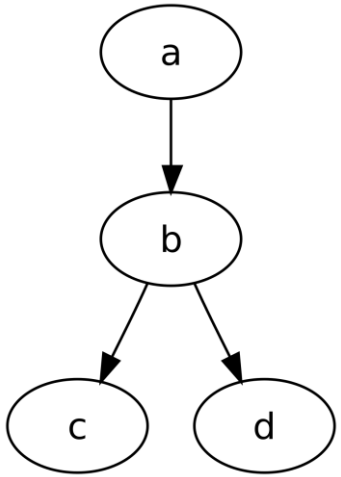
Язык DOT

Определение

DOT — язык описания графов. Это простой способ описания графа, который понятен одновременно людям и компьютеру.

Пример

```
digraph graphname
{
    a -> b -> c;
    b -> d;
}
```



GraphViz

Описание

GraphViz — пакет программ для автоматической визуализации графов. Также в составе пакета можно найти библиотеки для использования с другими приложениями. GraphViz — свободное ПО, которое распространяется по лицензии Eclipse Public License.

Сайт проекта

http://graphviz.org

Программа dot

Описание

В пакете GraphViz присутствует программа dot для визуализации неориентированных графов. При создании разметки графа dot учитывает несколько параметров: минимизация пересечений, удобство восприятия рисунка, сохранение иерархии входного графа, что очень важно для визуализации конечного автомата.

Входные данные

Программа принимает на вход описание графа в формате DOT.

Результат работы программы

Программа может выдавать результат во многих форматах, в том числе векторных: текст, PostScript, png, svg.

Демонстрация программы

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Спасибо за внимание!

101000, Россия, Москва, Мясницкая ул., д. 20
Тел.: (495) 621-7983, факс: (495) 628-7931
www.hse.ru