**Теория формальных языков для текста и коммуникации**

**Задания на лабораторные работы**

**Лабораторная работа 1. Генерация детерминированного конечного автомата по регулярному выражению**

Необходимо написать программу, которая принимает на вход регулярное выражение и строит по нему детерминированный конечный автомат. Далее на вход программе подается строка. Полученный конечный автомат должен определить, подходит ли введенная строка под заданное ранее регулярное выражение.

Критерии оценки:

3 - Регулярное выражение в недетерминированный КА

2 - избавление от е-дуг

2 - детерминированный КА

3 - разбор по КА

2 - оптимальный КА (бонусная задача)

**Лабораторная работа 2. Написание контекстно-свободной грамматики для естественного или формального языка**

Необходимо написать контекстно-свободную грамматику, описывающую фрагмент выбранного студентом языка. Объем грамматики: 15-20 правил. Грамматика может описывать как формальный, так и естественный язык.

Критерии оценки:

6 - полнота охвата выбранной группы (в рамках дюжины правил)

4 - красота решения (субъективное мнение преподавателя)

**Лабораторная работа 3. Использование программных инструментов для разбора цепочек при помощи контекстно-свободной грамматики**

Необходимо написать программу с использованием контекстно-свободного парсера строк, которая бы реализовала разбор грамматики, написанной во второй лабораторной работе.

Критерии оценки:

2 - рабочая программа

2 - с использованием специальной библиотеки

2 - парсер входного языка, который просто проверяет входное предложение

2 - парсер входного языка, который совершает какие-то действия над входным предложением

2 - красота решения (субъективное мнение преподавателя)

**Лабораторная работа 4. Снятие омонимии с использованием методов машинного обучения**

Необходимо написать программу, которая принимает на вход корпус со снятой омонимией (что-то с соревнований, НКРЯ, Universal Dependencies, …), тренирует модель машинного обучения (любую, можно и с глубинными сетями, по умолчанию CRF или Random Forest) и снимает омонимию с поданного на вход текста (например, с тестовой части корпуса).

Критерии оценки:

2 - использован размеченный корпус

4 - обучена модель

2 - снимает омонимию

2 - красота решения (субъективное мнение преподавателя)

**Лабораторная работа 1. Генерация детерминированного конечного автомата по регулярному выражению**

***Краткое теоретическое введение***

Детерминированный конечный автомат (ДКА) – это пятерка (K, VT, M, S, Z), где K – алфавит элементов, называемых состояниями, VT – входной алфавит, M – отображение K×VT→K, S∈K – начальное состояние, Z={z}, z∈K – непустое множество конечных состояний. Такой автомат называется детерминированным, так как на каждом шаге терминал из входной цепочки однозначно определяет следующее текущее состояние.

В общем виде, конечный автомат можно представить в виде графа. На практике КА удобно представлять в виде матрицы. Элемент B[i,j] содержит в себе число, обозначающее состояние Ks такое, что M[Si,Tj]=Ks. При этом еще один столбец матрицы хранит информацию о том, является символ конечным или нет.

Пусть дано следующее регулярное выражение.

\d+(.\d\*)?

Оно может быть представлено в виде следующей грамматики.

A ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

0A | 1A | 2A | 3A | 4A | 5A | 6A | 7A | 8A | 9A |

.B

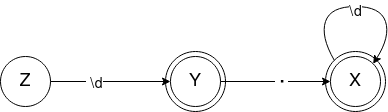
B ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

0B | 1B | 2B | 3B | 4B | 5B | 6B | 7B | 8B | 9B

Матрица, описывающая то же выражение, приведена в таблице ниже. В ней первый столбец показывает, какой из символов является начальным символом грамматики, второй символ содержит нетерминалы (выделены серым), а последний столбец показывает, какой терминал может являться конечным состоянием. Остальные столбцы содержат терминалы (терминалы выделены зеленым). Собственно матрица переходом находится в середине таблицы и показана песочным цветом.

|  |  | **\d** | **.** |  |
| --- | --- | --- | --- | --- |
| **→** | **Z** | Y |  | **0** |
|  | **Y** | Y | X | **1** |
|  | **X** | X |  | **1** |

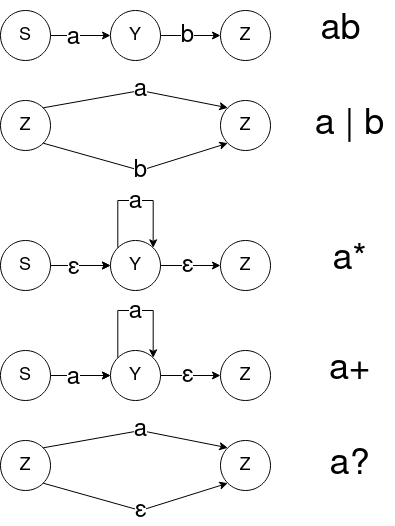
Этот же конечный автомат в виде графа показан на рисунке ниже.



Здесь специально изменены названия нетерминалов, так как два подхода (грамматика против конечного автомата) используют разные умолчания. Для грамматики подразумевается, что для продукций, в которых стоит только терминал, существует переход в некоторое скрытое состояние по пустому символу. Также считается, что должен прийти хотя бы один символ, иначе правило A не применится. Для конечного автомата второе умолчание раскрывается в явном виде - вводится нетерминал Z, который не является конечным состоянием (то есть для того, чтобы цепочка разобралась, надо чтобы пришел хотя бы один символ).

***Метод генерации недетерминированного конечного автомата по регулярному выражению***

Существует несколько стандартных шаблонов, позволяющих сгенерировать КА по описанию регулярного множества. Последовательно применяя данные шаблоны можно создать КА для любого регулярного множества.



Вершины, соединяемые с помощью подвыражения, принимаются за начальную и конечную вершины данного подвыражения. Алгоритм завершается, когда не останется ни одной строки, содержащей более одного терминала или символы итерации.

В начале алгоритма создаются начальная и конечная вершина (назовем их S и Z), которые соединяются стрелкой, помеченной строкой со всем выражением. Далее строка разделяется на фрагменты по следующей грамматике (нетерминалы приведены синим, терминалы - коричневым, элементы БНФ - красным).

**A ::= B `|` A | B**

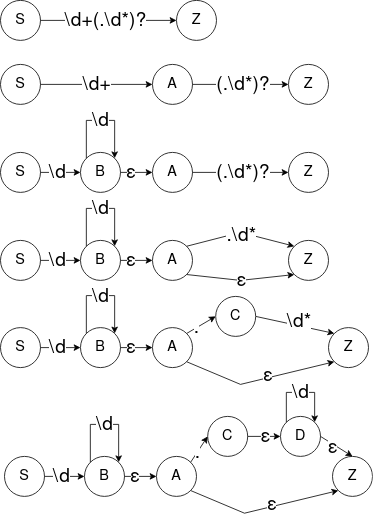
**B ::= C B | C**

**C := (A)|(A)+|(A)\*| a | a+ | a\***

То есть вначале выделяются выражения, соединенные знаком `|` (он специально дан в апострофах) и находящиеся вне скобок, далее терминалы и выражения в скобках идущие последовательно. Выражение в скобках разбирается как самостоятельное выражение. Заметьте, в данном случае очень удобно использовать разбор при помощи рекурсивного спуска, когда для каждого нетерминала пишется отдельная функция, разбирающая фрагмент входной строки. При этом несколько менее эффективным с точки зрения скорости, но более понятным с точки зрения реализации является подход, при котором снимается копия со всей интересующей нас подстроки (например, выделить всё подвыражение в скобках в отдельную строку и передать ее в качестве аргумента при вызове функции).

При выделении каждого подвыражения мы перестраиваем граф по шаблонам, приведенным выше. Выражение, которое приписывалось дуге, разделяется на части, которые приписываются вновь созданным дугам.

Разберем пример генерации недетерминированного конечного автомата при помощи предложенного метода (см. рис).



Аналогичные преобразования можно провести при помощи табличного представления конечного автомата. При этом мы отдельно можем выписывать еще не разобранные части выражения, а также состояния, которые эти части объединяют.

|  | \d | . | ε |  |
| --- | --- | --- | --- | --- |
| S |  |  |  | 0 |
| Z |  |  |  | 1 |

S → Z : \d+(.\d\*)?

|  | \d | . | ε |  |
| --- | --- | --- | --- | --- |
| S |  |  |  | 0 |
| Z |  |  |  | 1 |
| A |  |  |  | 0 |

S → A : \d+

A → Z : (.\d\*)?

|  | \d | . | ε |  |
| --- | --- | --- | --- | --- |
| S | B |  |  | 0 |
| Z |  |  |  | 1 |
| A |  |  |  | 0 |
| B | B |  | A | 0 |

A → Z : (.\d\*)?

|  | \d | . | ε |  |
| --- | --- | --- | --- | --- |
| S | B |  |  | 0 |
| Z |  |  |  | 1 |
| A |  |  | Z | 0 |
| B | B |  | A | 0 |

~~A → Z : ε~~

A → Z : .\d\*

|  | \d | . | ε |  |
| --- | --- | --- | --- | --- |
| S | B |  |  | 0 |
| Z |  |  |  | 1 |
| A |  | C | Z | 0 |
| B | B |  | A | 0 |
| C |  |  |  | 0 |

C → Z : \d\*

|  | \d | . | ε |  |
| --- | --- | --- | --- | --- |
| S | B |  |  | 0 |
| Z |  |  |  | 1 |
| A |  | C | Z | 0 |
| B | B |  | A | 0 |
| C |  |  | D | 0 |
| D | D |  | Z | 0 |

***Метод генерации детерминированного конечного автомата из недетерминированного***

Полученный конечный автомат является недетерминированным, так как он обладает переходами по пустым цепочкам. Помимо этого, недетерминированный конечный автомат может обладать несколькими переходами в разные состояния по одному и тому же символу из одного и того же состояния. Чтобы избавиться от недетерминизма, необходимо провести несколько шагов.

1. Избавление от е-дуг.
2. Построение детерминированного конечного автомата.
3. Оптимизация конечного автомата.

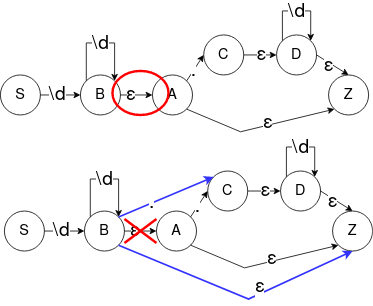
*Избавление от е-дуг*

Здесь мы пользуемся свойством эквивалентности конечных автоматов, которые должны допускать одни и те же цепочки.

Если у нас есть состояние, из которого исходит е-дуга, то при разборе мы перейдем по ней в новое состояние, не сдвинувшись по входной цепочке. После этого мы можем применить ее дуги для разбора.

Если мы скопируем все переходы, которыми обладает состояние, в которое ведет е-дуга, то мы можем начать разбор сразу с текущего состояния. Таким образом, конечный автомат будет допускать те же цепочки, но мы получаем возможность избавиться от е-дуги.

Итак, пусть имеется дуга A-ε→B. Чтобы избавиться от этой е-дуги в графе, необходимо скопировать все дуги, выходящие из В (состояния, в которое входит е-дуга), так, чтобы они начинались из состояния А (из которого эта е-дуга исходит), а заканчивались там, где заканчивались перед этим. (На рисунке ниже это показано преобразование показано графически.) Обратите внимание, это будет именно копирование дуг, так как мы можем попасть в В по какому-то другому маршруту. Из-за этого дуги нельзя удалять, а надо именно дублировать. Возможно, что как на рисунке ниже, у нас появится недостижимая вершина, но с этой проблемой мы справимся позже.



Заметим, что копировать надо и другие е-дуги. От них также можно избавиться при повторении этого шага.

Эта же операция может быть сделана и в таблице. Здесь мы объединяем две строки: состояние из которого выходит е-дуга с состоянием в которое она входит.

|  | \d | . | ε |  |
| --- | --- | --- | --- | --- |
| S | B |  |  | 0 |
| Z |  |  |  | 1 |
| A |  | C | Z | 0 |
| B | B |  | A | 0 |
| C |  |  | D | 0 |
| D | D |  | Z | 0 |

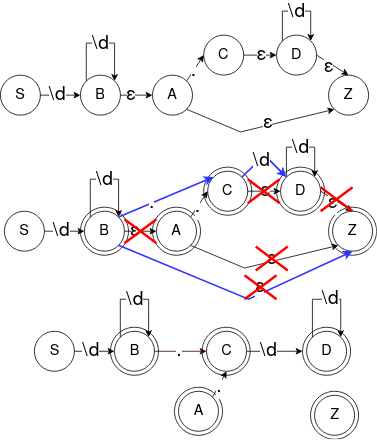
|  | \d | . | ε |  |
| --- | --- | --- | --- | --- |
| S | B |  |  | 0 |
| Z |  |  |  | 1 |
| A |  | C | Z | 0 |
| B | B | С | ~~A~~, Z | 0 |
| C |  |  | D | 0 |
| D | D |  | Z | 0 |

Если е-дуга ведет в конечное состояние, также копируется и метка конечного состояния.

|  | \d | . | ε |  |
| --- | --- | --- | --- | --- |
| S | B |  |  | 0 |
| Z |  |  |  | 1 |
| A |  | C | Z | 0 |
| B | B | C | ~~A~~, Z | 0 |
| C |  |  | D | 0 |
| D | D |  | Z | 0 |

|  | \d | . | ε |  |
| --- | --- | --- | --- | --- |
| S | B |  |  | 0 |
| Z |  |  |  | 1 |
| A |  | C | Z | 0 |
| B | B | С | ~~A~~, ~~Z~~ | 1 |
| C |  |  | D | 0 |
| D | D |  | Z | 0 |

Данный алгоритм применяется до тех пор, пока в графе или таблице имеется хоть одна е-дуга. В итоге мы получим следующие граф и таблицу.



|  | \d | . | ε |  |
| --- | --- | --- | --- | --- |
| S | B |  |  | 0 |
| Z |  |  |  | 1 |
| A |  | C | ~~Z~~ | 1 |
| B | B | С | ~~A~~, ~~Z~~ | 1 |
| C | D |  | ~~D~~, ~~Z~~ | 1 |
| D | D |  | ~~Z~~ | 1 |

*Построение детерминированного конечного автомата*

Как уже отмечалось выше, в детерминированном конечном автомате не должно быть дуг, маркированных одним и тем же терминалом и ведущих в разные состояния. Для преобразования мы снова будем использовать свойство эквивалентности автоматов.

Пусть имеется состояние, из которого исходит две дуги, помеченные одним и тем же символом. При разборе по такому автомату, мы должны перейти сперва по одной из дуг и попытаться продолжить разбор, принимая в расчет исходящие из нее дуги. Если разбор будет неуспешным, мы должны будем использовать переход во второе состояние. Получается, что в итоге мы переберем все возможные переходы состояний, в которые ведут данные дуги.

Вместо этого мы можем объединить все состояния, в который ведут дуги, помеченные одним и тем же символом. За счет этого мы получим новое состояние, которое предоставляет доступ сразу ко всем возможным переходам.

Итак, для того, чтобы избавиться от нескольких дуг, помеченных одним и тем же символом, необходимо создать новое состояние, в которое будут добавлены все дуги, исходящие из всех состояний, в которые ведут выбранные дуги. Формально, для списка состояний {s1, s2, …, sn} и нового состояния S’, , где *а* ∊VT.