Регулярные выражения

Лекция 4

Содержание

1. Регулярные выражения. Основные понятия и примеры.
2. Конечные автоматы
3. Регулярные языки и связь регулярных выражений и конечных автоматов

1. Регулярные выражения. Основные понятия

Для чего нужны регулярные выражения

- Обновить цену товара в прайс-листе:
  для конкретного товара за 1000р. сделать 999.99р.
- Заменить все вхождения одного слова в тексте на другое
  - для части слова (Википедия -> Энциклопедия)
  - с учетом контекста
- Найти сообщения о терроризме
- Фильтрация нецензурных высказываний на форумах

Регулярные выражения. Примеры

- Regular Expressions (RegExp)
- Языки программирования

(Python, Perl, Ruby, Java, .Net)

- Текстовые редакторы

(Vim, EmEdit)

- Утилиты (grep, sed)

Регулярные выражения. Примеры

- Регулярные выражения - алгебраическая нотация для записи множества строк
- Функции Python

```
#encoding=CP1251
import re
re.search("в","пиво").group(0) # в
re.sub("о","ко","пиво") # пивко
re.findall("cd","abcdcde") # [“cd”,”cd”]
```

Регулярные выражения. Примеры

- Последовательность букв: abcd
- Чувствительны к регистру: “Пиво” “пиво”
- Дизъюнкция: [П|п]иво,[abc],[1234567890]
- Интервал: [A-Z], [0-9], [A-Za-z]

```
for letter in re.findall("[а-о]","пиво"):
    print letter,
> и в о
```

Регулярные выражения. Примеры

- Знак ^: [^a] - все кроме “a”
- “.” - любой символ, кроме \n
- ? - условие для 0 или 1 вхождения символа

```
re.findall("пивк?о", "пиво или пивко")
> [“пиво”,”пивко”]
```

Регулярные выражения. Примеры

- Как найти “Gooooogle”?
- Счетчики

–Gooo*gle

–Goo+gle

```
print re.sub("Goo+gle","Google","Gooooogle")
> Google
```

Регулярные выражения. Примеры

- Дизъюнкция

```
re.sub("cat|dog","chicken","cat dog")
> chicken chicken
re.search("пив(о|ко)","пивко") # Match
```

- Якоря

^ - начало строки

$ - конец строки

```
re.search("^cat(1|2)","cat1 cat2").group(0)
> cat1
re.search("cat(1|2)$","cat1 cat2").group(0)
> cat 2
```

Регулярные выражения. Примеры

- Память (Memory)

```
text="A conditional random field (CRF) is a ..."
print re.search("\([^\)]+\)",text).group(0)
> (CRF)
print re.search("\(([^\)]+)\)",text).group(1)
> CRF
```

```
text1 = "the faster they ran, the faster we ran"
text2 = "the faster they ran, the faster we ate"
re.search("the (.*)er they (.*), the \\1er we \\2", text1) # Match
re.search("the (.*)er they (.*), the \\1er we \\2", text2) # Not match
```

Регулярные выражения. Примеры

- Приоритет операций

Круглые скобки	()
Счетчики	* + ? {}
Последовательности и якоря	the ^my end$
Дизъюнкция	|

Регулярные выражения. Практические примеры

- Найти все вхождения слова в тексте –слово “the” в тексте 7 раз

Регулярные выражения. Практические примеры

- ELIZA

User: Men are all alike.
ELIZA: IN WHAT WAY?
User: They're always bugging us about something other
ELIZA: CAN YOU THINK OF A SPECIFIC EXAMPLE?
User: My boyfriend made me come here
ELIZA: YOUR BOYFRIEND MADE YOU COME HERE
User: He says I'm depressed much of the time
ELIZA: I AM SORRY TO HEAR YOU ARE DEPRESSED

```
sentence=re.sub("I'm","YOU ARE", sentence)
...
sentence=re.sub(".* YOU ARE (depressed|sad) .*","I'M SORRY TO HEAR YOU ARE \\1",sentence)
sentence=re.sub(".* all .*","IN WHAT WAY?",sentence)
sentence=re.sub(".* always .*","CAN YOU THINK OF A SPECIFIC EXAMPLE?",sentence)
```

2. Конечные автоматы

Конечные автоматы

- Finite-state automation (FSA)
- Один из важнейших инструментов для обработки текстов
- Могут быть использованы для реализации регулярных выражений

Использование КА для распознавания языка

- Научимся говорить с овцами

–бээ!

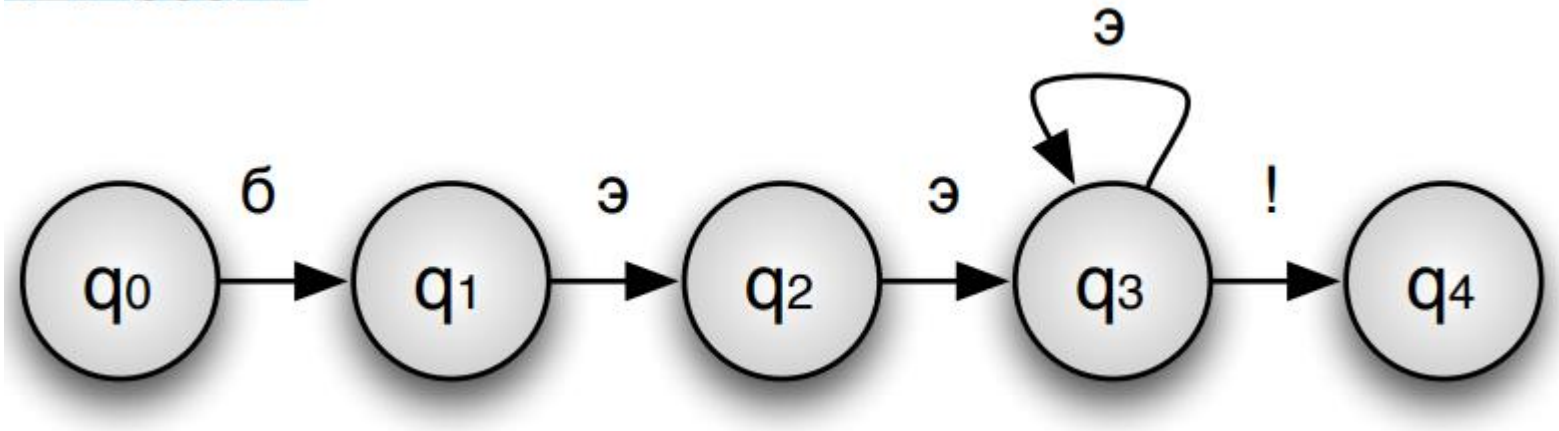
–бэээ!

–бээээ!

–бэээээ!

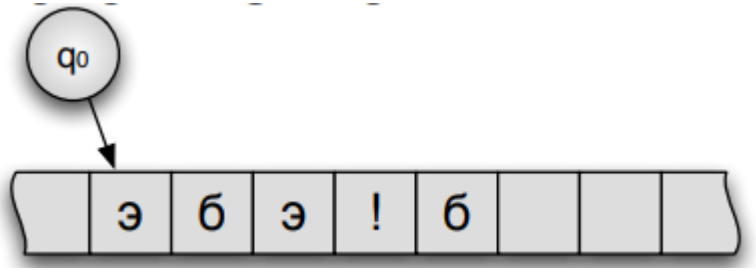
–...

- RE: “бээ+!”



Представление автоматов

- Текст: лента с ячейками



- Таблица переходов между состояниями

Состояние	Вход		
	б	э	!
0	1	∅	∅
1	∅	2	∅
2	∅	3	∅
3	∅	3	4
4	∅	∅	∅

Формальное определение

$Q = q_0 q_1 q_2 \ldots q_{N-1}$	конечное множество из N **состояний**
Σ	конечный **входной алфавит**
q_0	**начальное состояние**
F	множество **конечных состояний**
$\delta(q, i) : Q \times \Sigma \rightarrow Q$	**функция перехода** или матрица перехода между состояниями

Алгоритм распознавания для ДКА

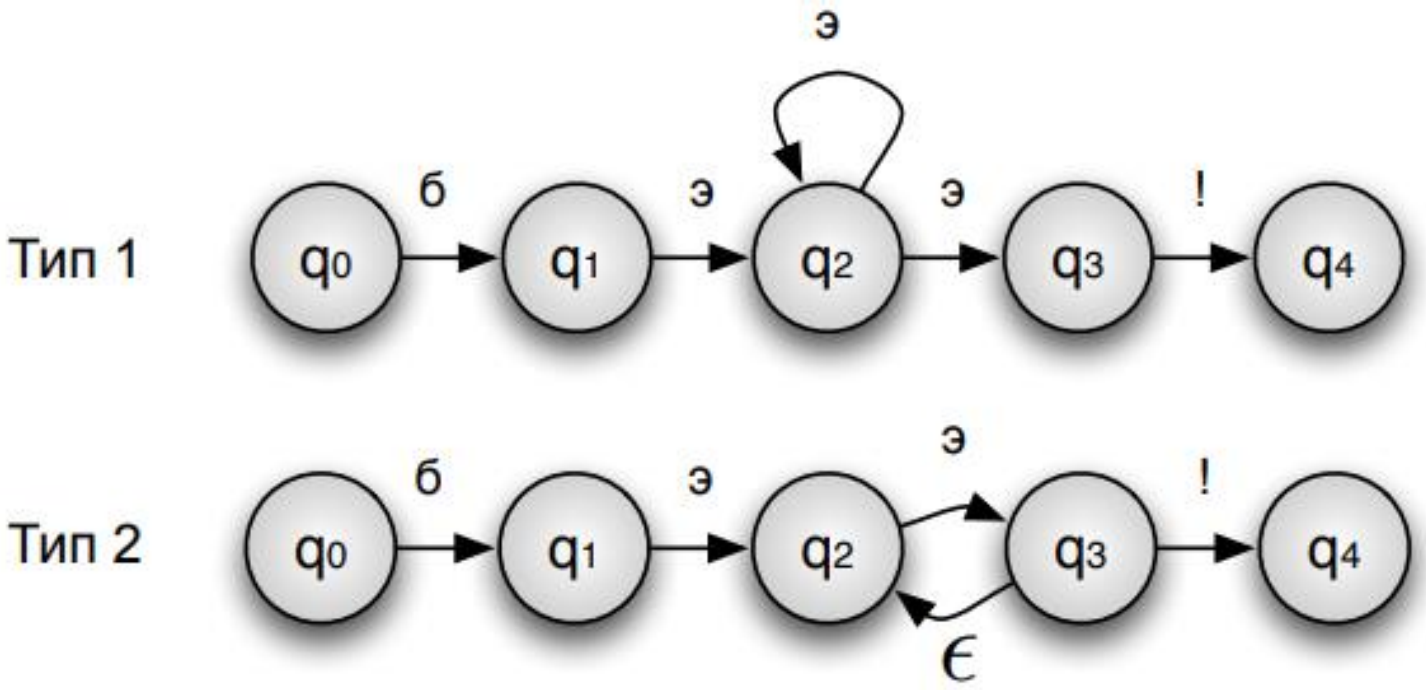
```
#encoding=CP1251
def recognize(tape, machine, acceptStates):
    index = 0 # Beginning of tape
    currentState = 0 # Initial state of machine
    while True:
        if index == len(tape):
            if currentState in acceptStates:
                return True
            else:
                return False
        elif machine[currentState].has_key(tape[index]):
            currentState = machine[currentState][tape[index]]
            index+=1
        else:
            return False

machineSheep = {0:{"б":1}, 1:{"э":2}, 2:{"э":3}, 3:{"э":3,"!":4}, 4:{}}
print recognize("бэээээээ!",machineSheep, [4])
```

	Вход		
	б	э	!
0	1	∅	∅
1	∅	2	∅
2	∅	3	∅
3	∅	3	4
4	∅	∅	∅

Недетерминированные КА

- Обобщение ДКА
- Недетерминизм двух типов



Распознавание для НКА

- Подходы к решению проблемы недетерминизма

–Сохранение состояний (backup)

- поиск в глубину и ширину

–Просмотр будущих состояний (look-ahead)

–Параллелизм

Состояние	Вход			
	б	э	!	ϵ
0	1	∅	∅	∅
1	∅	2	∅	∅
2	∅	2,3	∅	∅
3	∅	∅	4	∅
4	∅	∅	∅	∅

ДКА и НКА

- ДКА и НКА эквивалентны
- Существует простой алгоритм для преобразования НКА в ДКА
- Идея:
  - взять все параллельные ветки НКА
  - в них взять все состояния, в которых одновременно может находиться НКА
  - объединить их в новое состояние ДКА
- В худшем случае НКА с N состояниями преобразуется в ДКА с 2^N состояниями

3. Регулярные языки и связь регулярных выражений и конечных автоматов

Регулярные языки

1. $\varnothing$ - регулярный язык
2. $\forall a \in \Sigma \cup \epsilon, \{a\}$ - регулярный язык
3. Для любых регулярных языков L_1 и L_2, такими также являются:
   (a) $L_1 \cdot L_2 = \{xy \mid x \in L_1, y \in L_2\}$, соединение L_1 и L_2
   (b) $L_1 \cup L_2$, объединение или дизъюнкция L_1 и L_2
   (c) L_1^*, замыкание (Клини) языка L_1

Регулярные языки

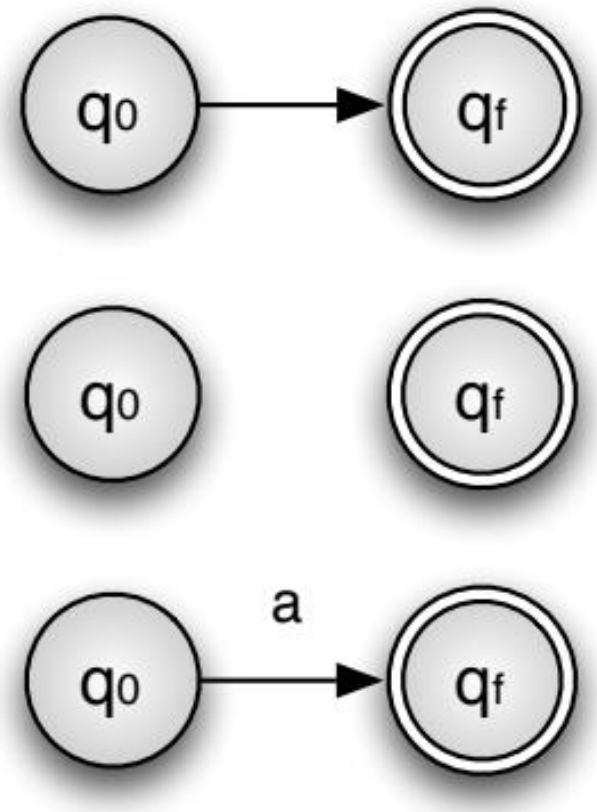
$\sum$ - Алфавит, ϵ - пустая строка

- регулярные языки также замкнуты относительно операций
  - пересечения
  - разности
  - дополнения
  - инверсии

Регулярные языки и регулярные выражения

- Любые операторы регулярных выражений (кроме памяти) можно выразить с помощью трех операций регулярных языков:
  - –соединение
  - –объединение
  - –замыкание Клини
- Пример: счетчики (*, +, {n,m}) являются частным случаем повторения плюс замыкание Клини

Построение автомата для регулярных выражений

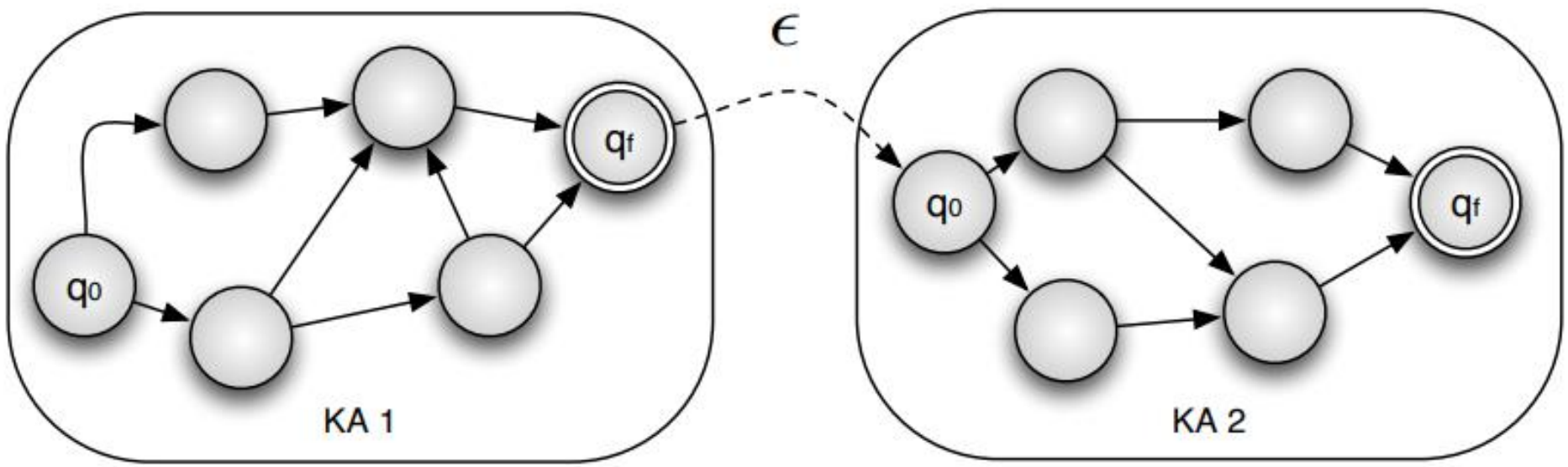


$r = \epsilon$

$r = \varnothing$

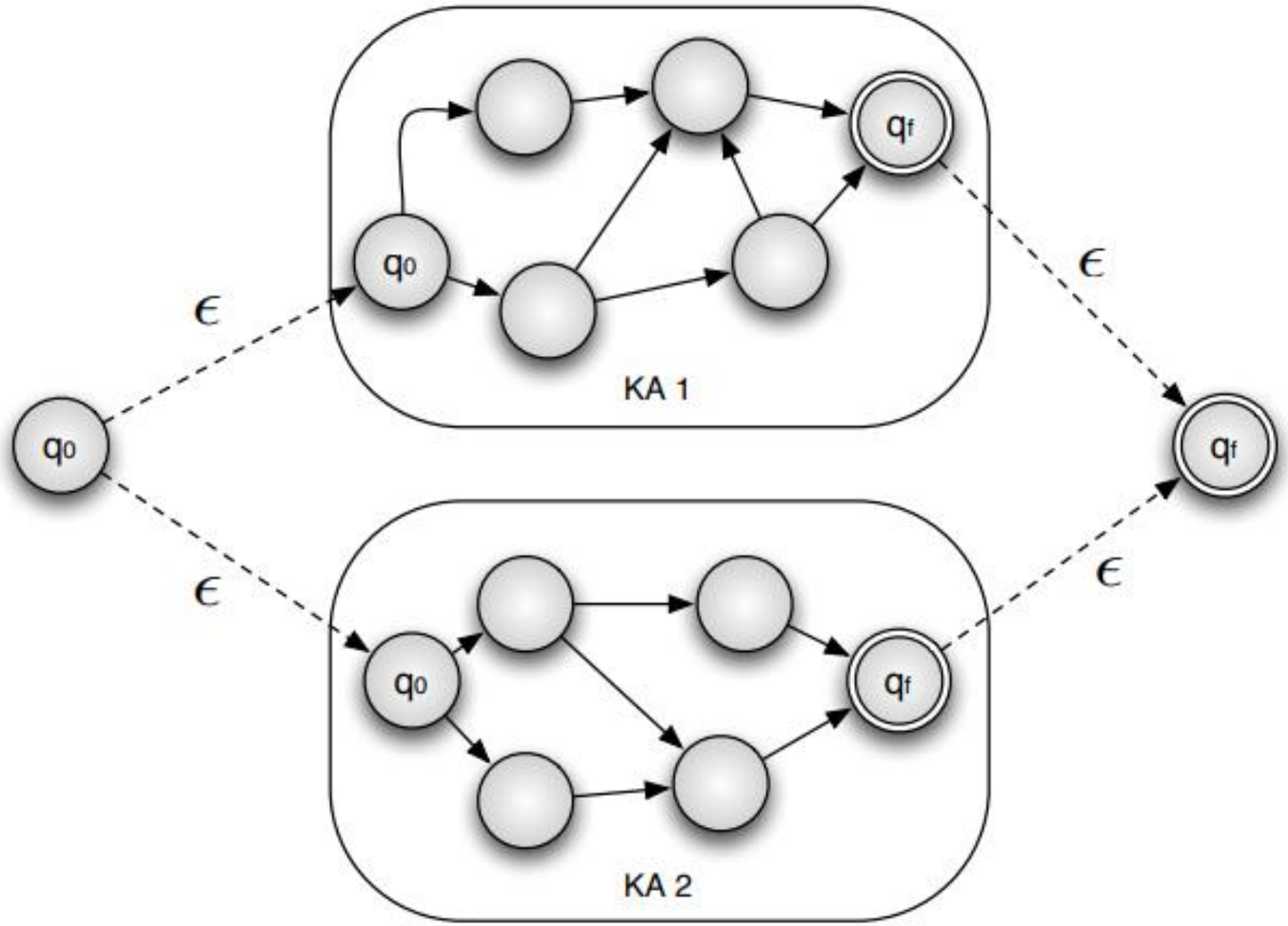
$r = a$

Построение автомата для регулярных выражений



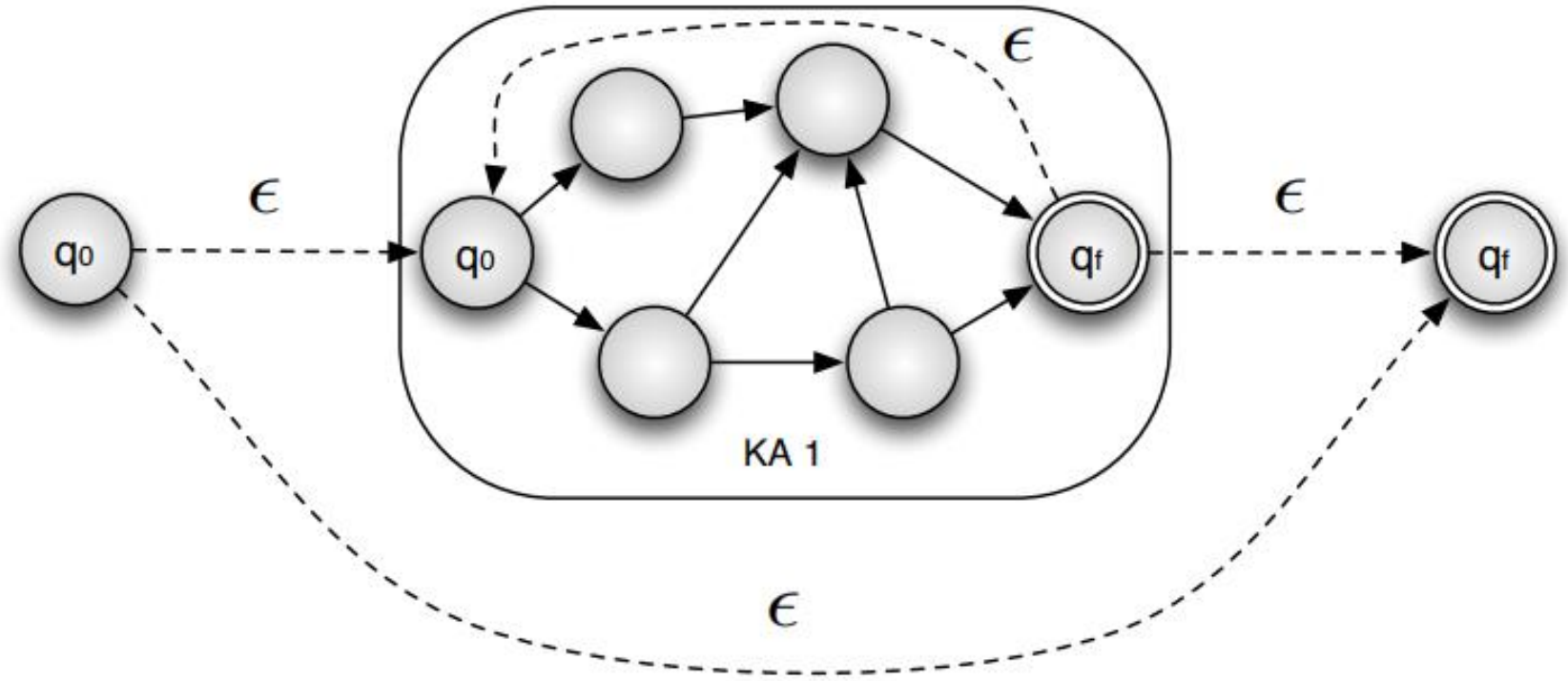
Последовательное соединение двух конечных автоматов

Построение автомата для регулярных выражений



Объединение двух конечных автоматов

Построение автомата для регулярных выражений



Замыкание конечного автомата

Регулярные выражения. Примеры использования

Разделение строки с помощью разделителя (delimiter)

```
#define C_ALL(X) cbegin(X), cend(X)

vector<string> split(const string& str, string_view pattern) {
    const auto r = regex(pattern.data());
    return vector<string>{
        sregex_token_iterator(C_ALL(str), r, -1),
        sregex_token_iterator()
    };
}

int main() {
    assert((split("/root/home/vishal", "/")
                == vector<string>{"", "root", "home", "vishal"}));
    return EXIT_SUCCESS;
}
```

Регулярные выражения. Примеры использования

Удаление пробелов из строки

```
string trim(string_view text) {
    static const auto r = regex(R"(\s+)");
    return regex_replace(text.data(), r, "");
}

int main() {
    assert(trim("12   3 4      5"s) == "12345"s);
    return EXIT_SUCCESS;
}
```

Регулярные выражения. Примеры использования

Поиск строк, содержащих или НЕ содержащих определенные слова, из файла

```
string join(const vector<string>& words, const string& delimiter) {
    return accumulate(next(begin(words)), end(words), words[0],
            [&delimiter](string& p, const string& word)
            {
                return p + delimiter + word;
            });
}
```

Регулярные выражения. Примеры использования

Поиск строк, содержащих или НЕ содержащих определенные слова, из файла

```
vector<string> lines_containing(const string& file, const vector<string>& words) {
    auto prefix = "^.*?\\b("s;
    auto suffix = ")\\b.*$"s;

    //  ^.*?\b(one|two|three)\b.*$
    const auto pattern = move(prefix) + join(words, "|") + move(suffix);

    ifstream        infile(file);
    vector<string>  result;

    for (string line; getline(infile, line);) {
        if(regex_match(line, regex(pattern))) {
            result.emplace_back(move(line));
        }
    }

    return result;
}
```

Регулярные выражения. Примеры использования

Поиск строк, содержащих или НЕ содержащих определенные слова, из файла

```
int main() {
   assert((lines_containing("test.txt", {"one","two"})
                                        == vector<string>{"This is one",
                                                          "This is two"}));

    return EXIT_SUCCESS;
}
/* test.txt
This is one
This is two
This is three
This is four
*/
```

Регулярные выражения. Примеры использования

Поиск файлов в папке

```
namespace fs = std::filesystem;

vector<fs::directory_entry> find_files(const fs::path &path, string_view rg) {
    vector<fs::directory_entry> result;
    regex r(rg.data());
    copy_if(
        fs::recursive_directory_iterator(path),
        fs::recursive_directory_iterator(),
        back_inserter(result),
        [&r](const fs::directory_entry &entry) {
            return fs::is_regular_file(entry.path()) &&
                   regex_match(entry.path().filename().string(), r);
        });
    return result;
}
```

Регулярные выражения. Примеры использования

Поиск файлов в папке

```
int main() {
    const auto dir         = fs::temp_directory_path();
    const auto pattern     = R"(\w+\.png)";
    const auto result      = find_files(fs::current_path(), pattern);
    for (const auto &entry : result) {
        cout << entry.path().string() << endl;
    }
    return EXIT_SUCCESS;
}
```

Ссылка на лабораторную работу

https://teccxx.neocities.org/mx1/13_task#%D1%86%D0%B5%D0%BB%D0%B8-%D0%B8-%D0%BA%D1%80%D0%B8%D1%82%D0%B5%D1%80%D0%B8%D0%B8