Лексер + Парсер. Часть 1.

Илья Филиппов
21.09.2015

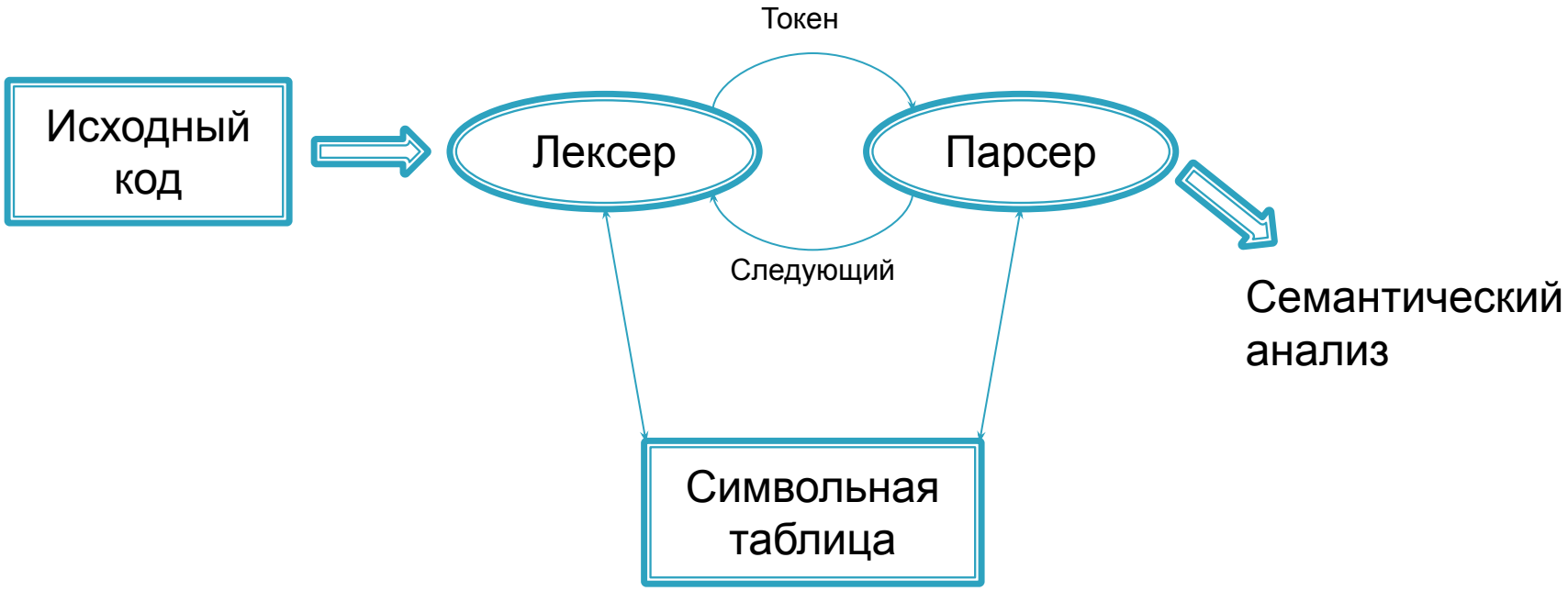
Этапы компиляции

- Фронтэнд
- Машинно независимые оптимизации
- Код генерация + машинно зависимые оптимизации

Этапы фронтэнда

- Лексический анализатор (лексер)
- Синтаксический анализатор (парсер)
- Семантический анализатор
- Генерация промежуточного представления

Схема работы



Исходный код

- if (a == b) then
  a += 5;
  else
  a -= 5;

- if (a == b) then\n\ta += 5;\nelse\n\ta-=5;

Схема работы

- Лексер формирует последовательности входных символов в лексемы, определяет их тип и отправляет токены парсеру.
  - Лексема – минимальная единица языка, имеющая самостоятельный смысл.
  - Токен – тип лексемы + аттрибут
- Парсер формирует исходное выражение языка, запрашивая токены.

Примеры лексем и токенов

- Лексема -- Токен
- 12345 (число, 12345)
- temp_1 (идентификатор, указатель на симтаб)
- += (оператор, plus_assign)
- + (оператор, plus)
- const (ключевое слово, const)
- Void (ключевое слово, void)
- var_name (идентификатор, указатель на симтаб)
- * (оператор, star)

Схема работы

- Обрабатываемые лексером и парсером последовательности символов и токенов напрямую зависят от спецификации языка.
- Необходим способ описания
  - «что в языке может быть»
  - «что в языке не может быть»

Строгие определения. Грамматики.

- Алфавит – множество символов, используемых в языке
- Терминальный символ - символ из алфавита
- Нетерминальный символ – символ не из алфавита
- Цепочка — последовательность символов
- Терминальная цепочка – цепочка, состоящая из терминальных символов
- Язык – множество терминальных цепочек

Грамматика — G = (N, T, P, S)

- N — множество нетерминальных символов (напр. A, B, C...)
- T (иногда E) — алфавит терминальных символов ($N \cap T = \varnothing$)
- P — конечное множество правил вывода
  - $P = \{\alpha \to \beta \mid \alpha \in (N \cup T)^+; \beta \in (N \cup T)^*\}$
- $S \in N$ — аксиома (или начальный символ) грамматики

Пример грамматики:

S->aQbZ

Q->ab | cc | Qd

Z -> aQa | c | ε

Классификация грамматик по Хомскому

- Тип 0: неограниченные
- Тип 1: контекстно-зависимые / неукорачивающие
- Тип 2: контекстно-свободные
- Тип 3: регулярные: праволинейные/леволинейные

Строгие определения. Типы грамматик.

- Регулярные грамматики:
  - праволинейные ($A \to w$, $A \to wB$, $A,B \in N$, $w \in T^*$)
  - леволинейные ($A \to w$, $A \to Bw$, $A,B \in N$, $w \in T^*$)
- Контекстно-свободные грамматики:
  - ($A \to w$, $A \in N$, $w \in (T \cup N)^*$)

Соответствие языков и грамматик

- Тип 0 (неограниченные): естественные языки:
  - Русский
  - Английский
- Тип 2 (контекстно-свободные): большинство языков программирования:
  - Java
  - C++
- Тип 3: (регулярные): описание отдельных лексем в языках программирования:
  - Идентификатор
  - Числовая константа

Способы разбора грамматик

- Тип 2 контекстно – свободная грамматика:
  - может быть описана с помощью конечного автомата с магазинной памятью
  - Используется для анализа последовательности токенов синтаксическим анализатором
- Тип 3 регулярная грамматика:
  - Может быть описана с помощью конечного автомата
  - Используется для формирования лексемы лексическим анализатором

Строгие определения. Регулярные множества.

Регулярное множество в алфавите T определяется следующим образом:

- $\varnothing$ — регулярное множество в алфавите T
- $\{a\}$ — регулярное множество в алфавите T для каждого a ∈ T
- $\{\epsilon\}$ — регулярное множество в алфавите T
- Если P и Q — регулярные множества в алфавите T, то регулярны множества
  - $P \cup Q$ (объединение)
  - PQ (конкатенация, $\{pq|p \in P, q \in Q\}$)
  - P^* (итерация: $P^* = \{\epsilon\} \cup P \cup PP \cup PPP \cup ...)$
- Ничто другое не является регулярным множеством в алфавите T

Регулярное выражение — форма записи регулярного множества.

Регулярное выражение и обозначаемое им регулярное множество определяются следующим образом:

- $\varnothing$ — обозначает множество $\varnothing$
- ϵ — обозначает множество $\{\epsilon\}$
- a — обозначает множество $\{a\}$
- Если регулярные выражения *p* и *q* обозначают множества *P* и *Q* соответственно, то:
  - $(p|q)$ обозначает $P \cup Q$
  - (pq) обозначает PQ
  - (p^*) обозначает P^*
- Ничто другое не является регулярным выражением в данном алфавите

Пример регулярного выражения

- Выражению «(a(b|c))*c» удовлетворяют:
  - c
  - ababacc
  - abacabc
- Не удовлетворяют:
  - ac
  - abbc
  - abacac

Строгие определения. Конечные автоматы.

Недетерминированный конечный автомат - $M = (Q, \Sigma, D, q_0, F)$

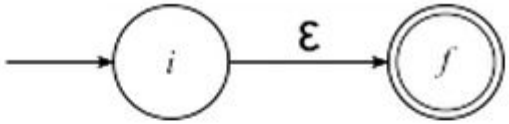
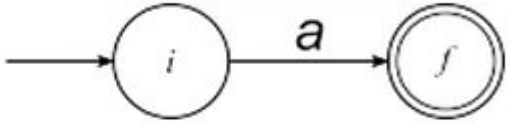
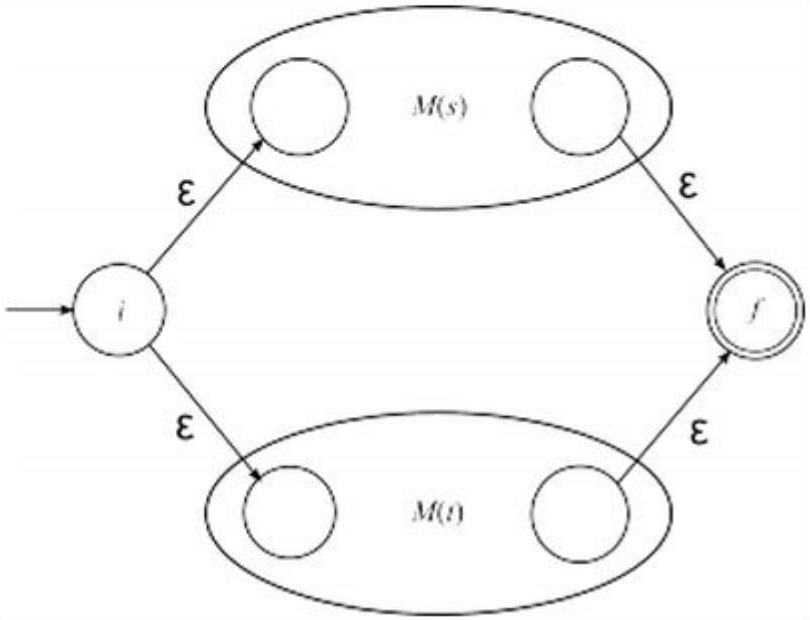
- Q — конечное непустое множество состояний
- Σ — входной алфавит
- D — правила перехода
  - $Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$
- $q_0 \in Q$ — начальное состояние
- $F \subseteq Q$ — множество конечных состояний

Детерминированный конечный автомат - $M = (Q, \Sigma, D, q_0, F)$

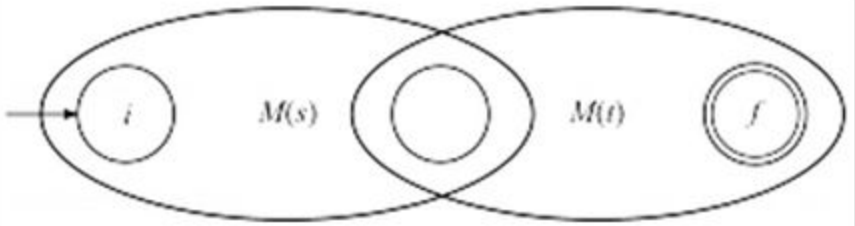
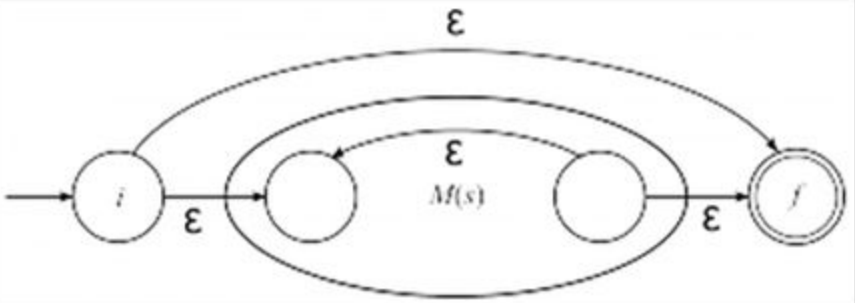
- Q — конечное непустое множество состояний
- Σ — конечный входной алфавит
- D — правила перехода
  - $Q \times \Sigma \to Q$
- $q_0 \in Q$ — начальное состояние
- $F \subseteq Q$ — множество конечных состояний

Схема построения лексера

Регулярное Выражение -> НКА

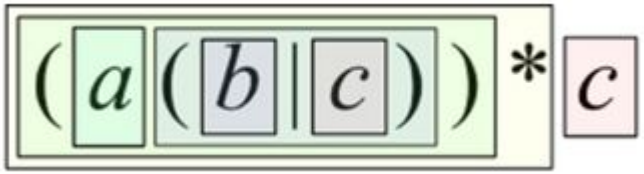
подвыражение РВ	автомат
ε	
$a, a \in T$	
$s\|t$	

Регулярное Выражение -> НКА

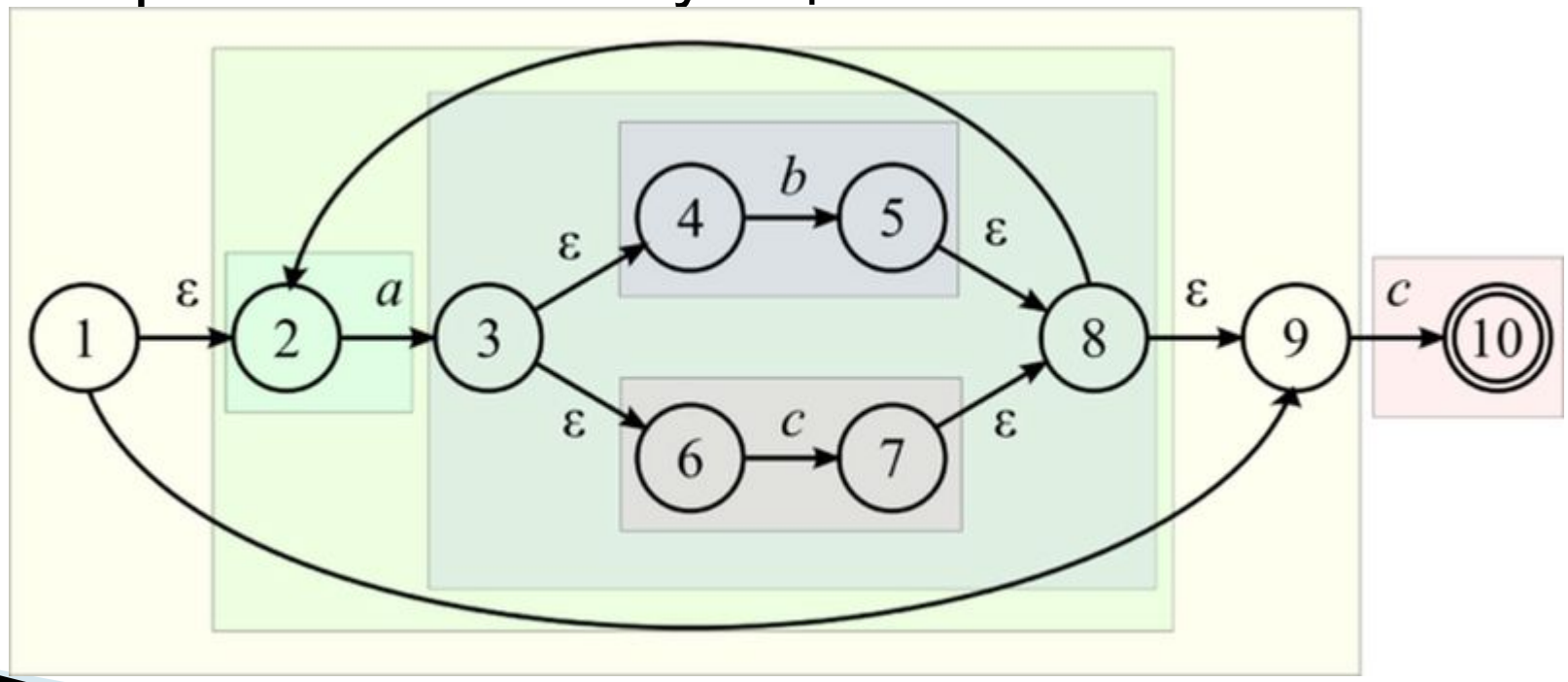
подвыражение РВ	автомат
st	
*s**	

Пример

- Рассмотрим регулярное выражение:



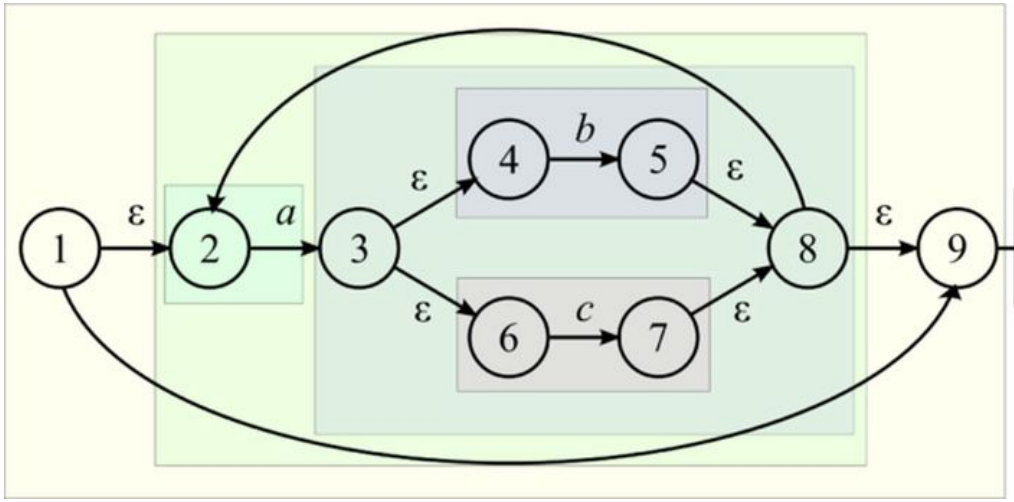
- Построим соответствующий НКА:

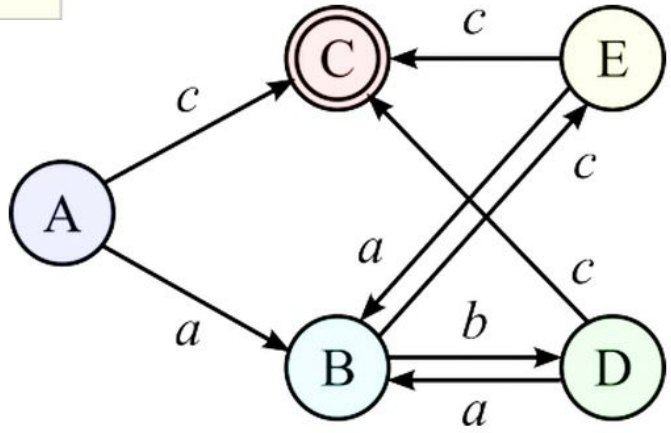


НКА -> ДКА

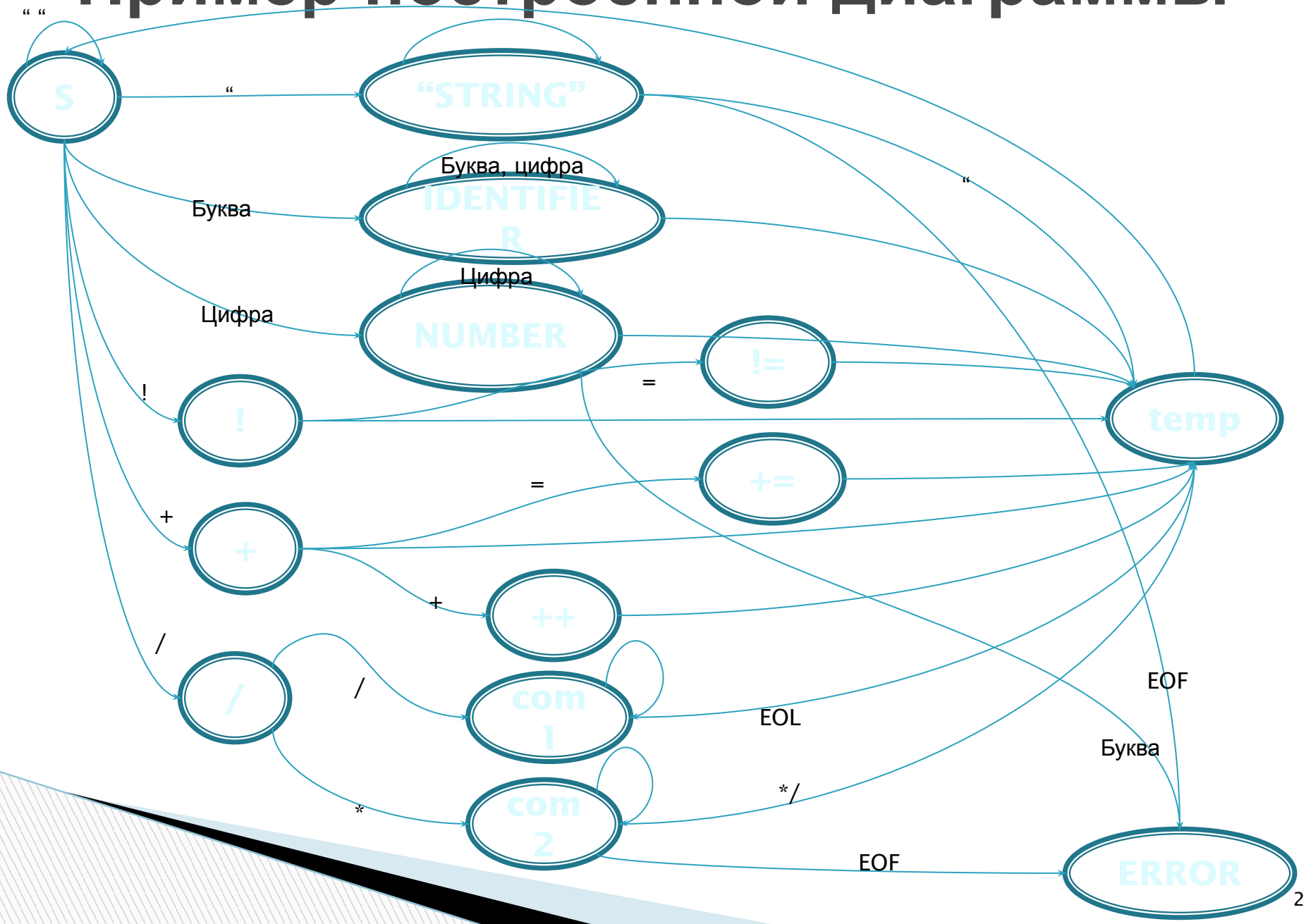
- ε-замыкание(S) — множество состояний, которые достижимы из S путём переходов по ε
- Начальное состояние ДКА - ε-замыкание начального состояния НКА
- While(есть нерассмотренное состояние ДКА: «cur»)
  - Для каждого состояния "B1" НКА, входящего в “cur”:
    - Для каждого перехода “P” из "B" в “B2":
      - Добавить состояние “new” ε-замыкание(B2)
      - Добавить переход “cur” -> “new” по P
- Конечные состояния ДКА – состояния, содержащие конечные состояния НКА

НКА -> ДКА Пример





Пример построенной диаграммы



Ошибки находящиеся лексером

- Неполная лексема
  - Конец файла между /* … */
  - Конец файла внутри строки в кавычках
  - Буквенный символ в цифровой константе: 123q
  - Некорректный символ: @

Должны быть ясны вопросы:

- Место выполнения лексического анализатора
- Схема работы лексического анализатора
- Какие ошибки обрабатываются лексическим анализатором
- Идейная схема построения лексического анализатора
- Предназначение символьной таблицы для лексера