> *09.04.2019*

Лексический анализ

Лексема — элементарная смысловая единица текста. Могут объединяться в классы эквивалентности — **токены**. Описываются токены с помощью **шаблонов**.

> Классы идентификаторов. Одноэлементные классы ключевых слов.

Токен	Пример (лексема)	Шаблон
if	if	`if`
cop	<=, >=, ==, ...	`<=?|>=?|[=!]=`
id	pi, var23	`[a-z]\d*`
number	1.09, -2.75e-30	`-?0|[1-9]\d*(.\d{1,})?(e-?[1-9]\d*)?`

> Первая колонка — имя, которое будет приходить на вход синтаксическому анализатору.
> comparison operator

Вся собранная информация сохраняется в таблице символов.

Буферы для чтения текста. Два, чтобы можно было заменять один на другой?

| | | | | | | | | | | | | | | | | | | | ↑↑ *begin current*

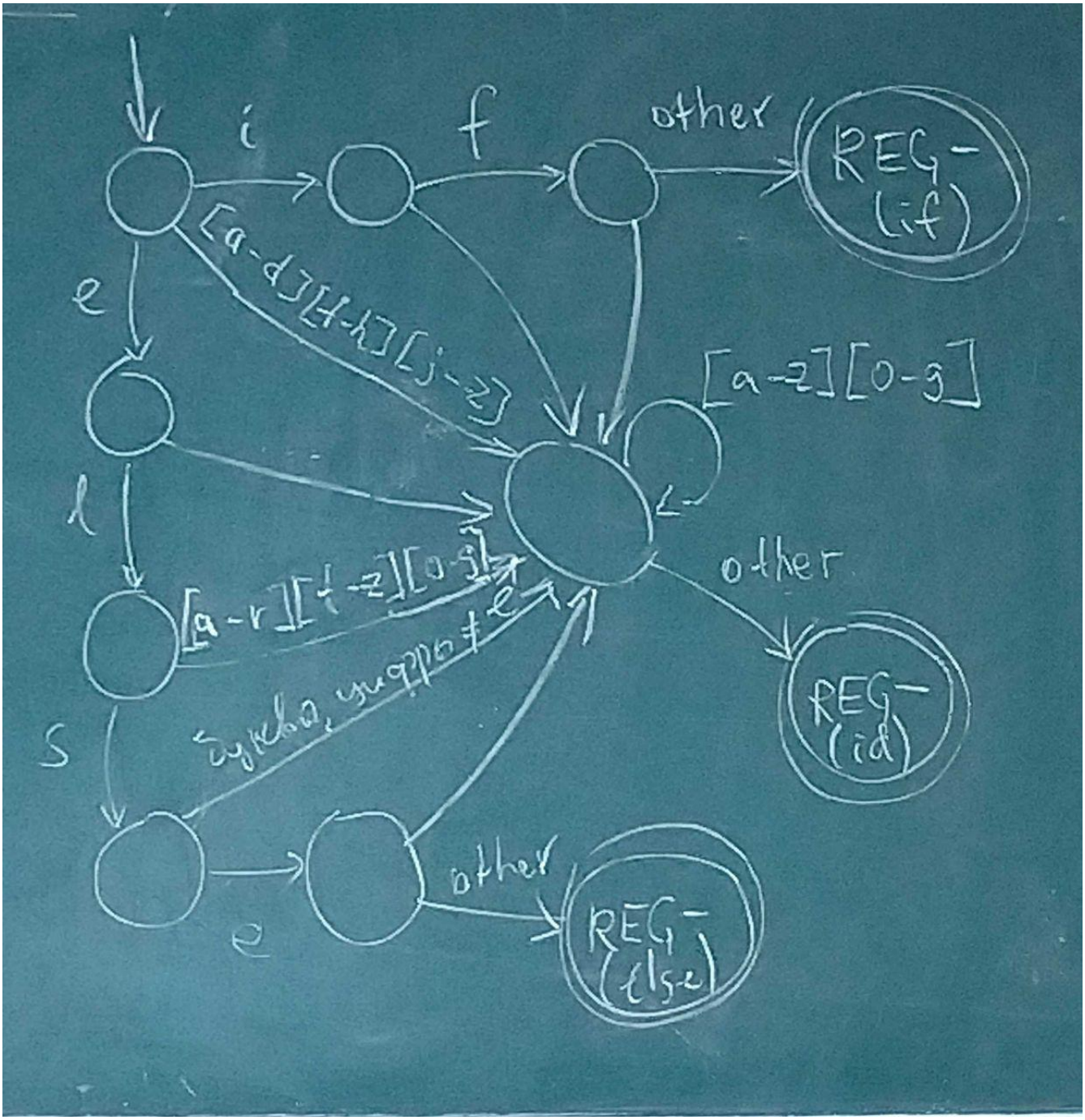
Регистрация токена

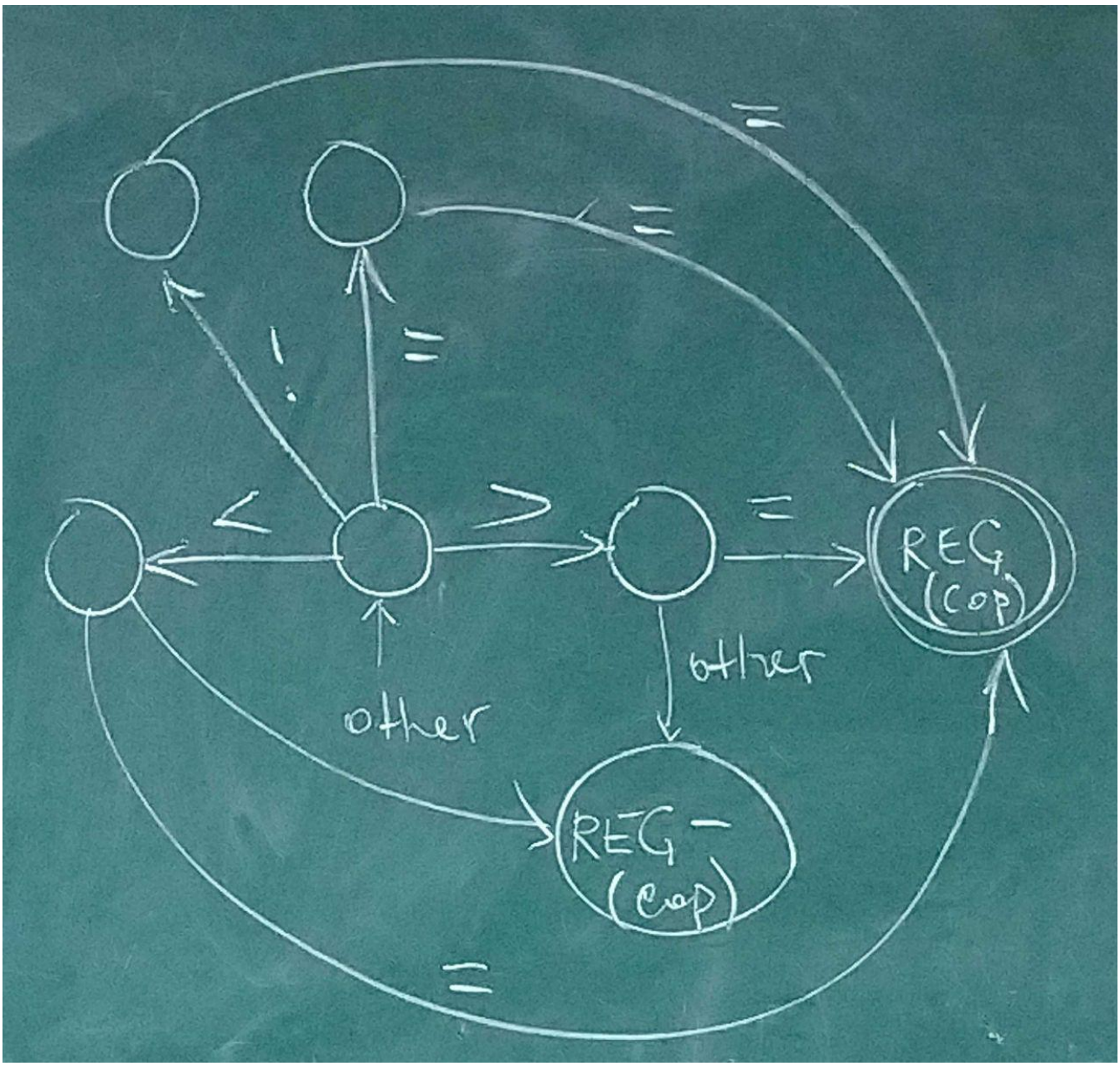
REG — функция, которая точно знает где начался и закончился токен. Делает запись в таблицу символов: какой это был оператор, где он находился в исходном тексте. Нужно для обработчика ошибок.

REG- — до предпоследнего символа.

Токен = <имя, атрибут>. Атрибут — ссылка на запись в таблице символов. Имя используется в лексическом анализе.

Примеры


i
f
other
REG-
(if)
[a-d][f-h][j-z]
e
[a-z][0-9]
l
other
[a-r][t-z][0-9]
Буква, цифра ≠ e
REG-
(id)
s
other
REG-
(else)
e



Ошибки

- несуществующий переход в автомате

Обработка ошибок

> Не хотим сразу навсегда ломаться, а выдавать все ошибки за раз.

- режим паники — пропускаем всё, пока не встретим корректные символы

Синтаксический анализ

Задача — определить принадлежность слова языку, который задан некоторой КС грамматикой. Грамматика описывает **синтаксис**! Также нужно построить дерево вывода и сообщить об ошибках.

Типы анализаторов

- универсальные (алгоритм КЯК);
- нисходящие (восстановление дерева от корня к листьям);
- восходящие.

Разделённая грамматика

$\forall A \in \Gamma : (\forall (A \to \gamma) \in P$ все γ начинаются с разных терминалов$)$

> До этого было: $A \to \gamma \; \forall a \in Z : (A, a) \to (\gamma, —)$