Automata and Formal Languages (4)

Email:	christian.urban at kcl.ac.uk
Office:	S1.27 (1st floor Strand Building)
Slides:	KEATS (also home work is there)

Last Week

Last week I showed you

- a tokenizer taking a list of regular expressions
- tokenization identifies lexeme in an input stream of characters (or string) and cathegorizes them into tokens

Two Rules

- Longest match rule (maximal munch rule): The longest initial substring matched by any regular expression is taken as next token.

- Rule priority: For a particular longest initial substring, the first regular expression that can match determines the token.

"if true then then 42 else +"

KEYWORD:

"if","then","else",

WHITESPACE:

" ","\n",

IDENT:

LETTER · (LETTER + DIGIT + "_")*

NUM:

(NONZERODIGIT · DIGIT*) + "0"

OP:

"+"

COMMENT:

"/*" · (ALL* · "*/" · ALL*) · "*/"

`"if true then then 42 else +"`

KEYWORD(if),
WHITESPACE,
IDENT(true),
WHITESPACE,
KEYWORD(then),
WHITESPACE,
KEYWORD(then),
WHITESPACE,
NUM(42),
WHITESPACE,
KEYWORD(else),
WHITESPACE,
OP(+)

```
"if true then then 42 else +"
```

KEYWORD(if),
IDENT(true),
KEYWORD(then),
KEYWORD(then),
NUM(42),
KEYWORD(else),
OP(+)

There is one small problem with the tokenizer. How should we tokenize:

"x - 3"

OP:
"+", "-"

NUM:
(NONZERODIGIT · DIGIT*) + "0"

NUMBER:
NUM + ("-" · NUM)

Deterministic Finite Automata

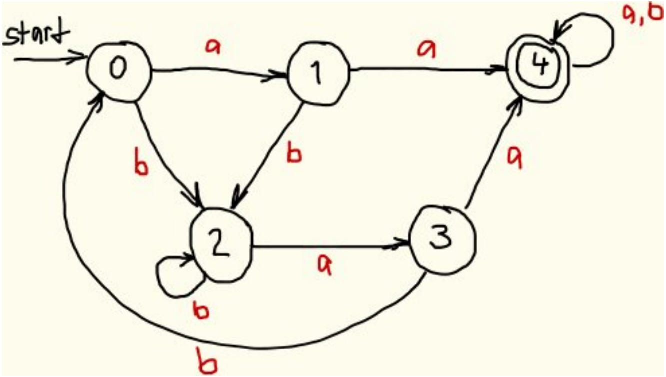
A deterministic finite automaton consists of:

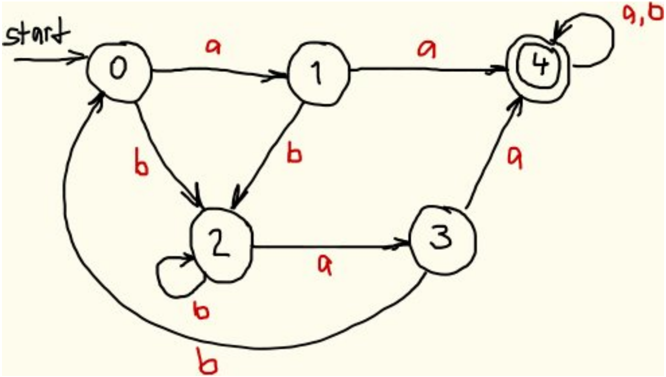
- a finite set of states
- one of these states is the start state
- some states are accepting states, and
- there is transition function

  which takes a state and a character as arguments and produces a new state

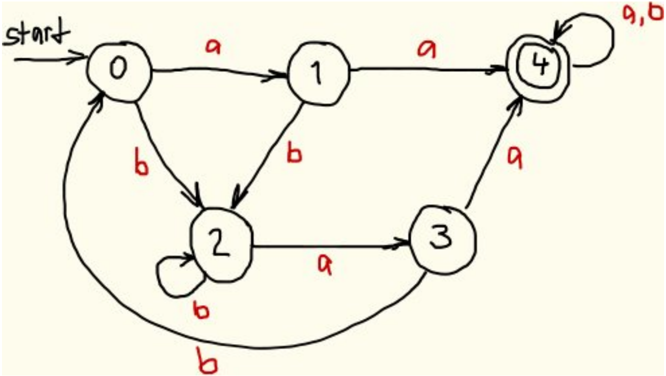
  this function might not always be defined everywhere

$$A(Q, q_0, F, \delta)$$


start
0
a
1
a
4
a,b
b
b
a
2
a
3
b
b



- start can be an accepting state
- there is no accepting state
- all states are accepting



for this automaton δ is the function

$$\begin{array}{lll} (q_0, a) \rightarrow q_1 & (q_1, a) \rightarrow q_4 & (q_4, a) \rightarrow q_4 \\ (q_0, b) \rightarrow q_2 & (q_1, b) \rightarrow q_2 & (q_4, b) \rightarrow q_4 \quad \ldots \end{array}$$

Accepting a String

Given

$$A(Q, q_0, F, \delta)$$

you can define

$$\begin{array}{l} \hat{\delta}(q, "\,") = q \\ \hat{\delta}(q, c :: s) = \hat{\delta}(\delta(q, c), s) \end{array}$$

Accepting a String

Given

$$A(Q, q_0, F, \delta)$$

you can define

$$\hat{\delta}(q, "\,") = q$$
$$\hat{\delta}(q, c :: s) = \hat{\delta}(\delta(q, c), s)$$

Whether a string s is accepted by A?

$$\hat{\delta}(q_0, s) \in F$$

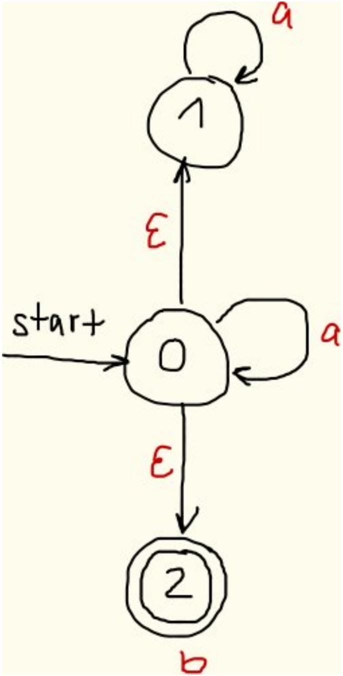
Non-Deterministic Finite Automata

A non-deterministic finite automaton consists again of:

- a finite set of states
- one of these states is the start state
- some states are accepting states, and
- there is transition relation

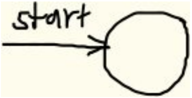
$$(q_1, a) \rightarrow q_2$$
$$(q_1, a) \rightarrow q_3$$

$$(q_1, \epsilon) \rightarrow q_2$$

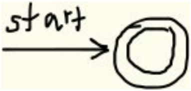


a
1
ε
start
0
a
ε
2
b

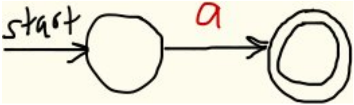
$\varnothing$



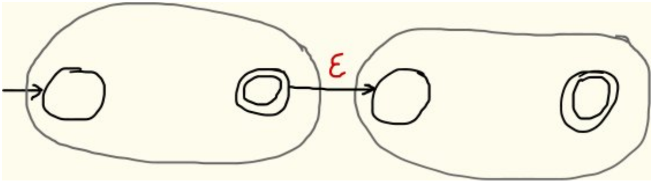
ϵ

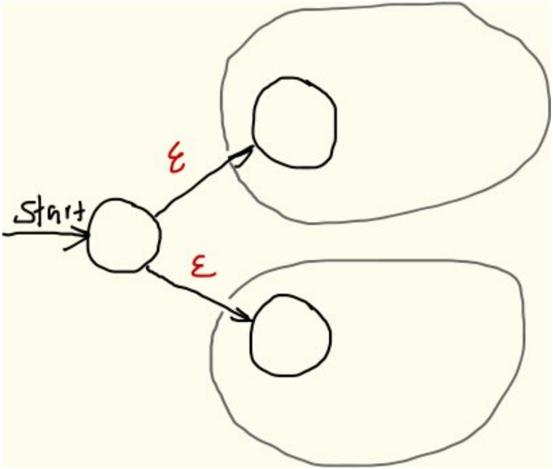


c



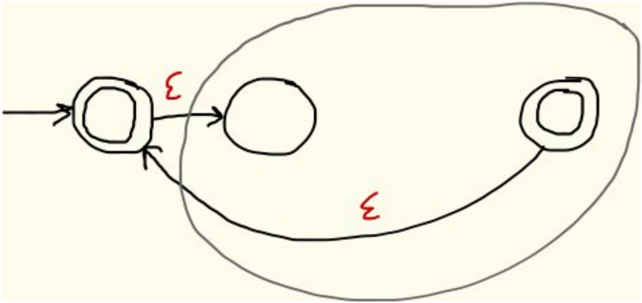
$r_1 \cdot r_2$

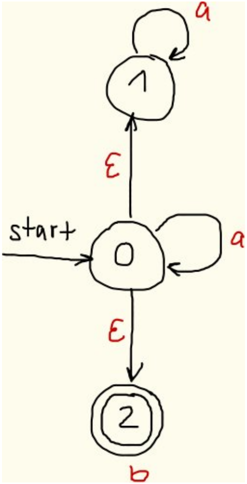


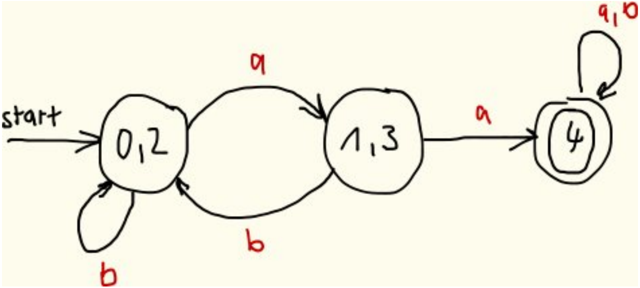


$r_1 + r_2$

r^*




a
1
ε
start
0
a
ε
2
b



start
0,2
1,3
4
a
a
b
b
a,b

Languages

A language is regular iff there exists a regular expression that recognises all its strings.

Languages

A language is regular iff there exists a regular expression that recognises all its strings.

not all languages are regular, e.g. $a^n b^n$.

- Assuming you have the alphabet {a, b, c}
- Give a regular expression that can recognise all strings that have at least one b.

- The star-case in our proof needs the following lemma

$$\text{Der}\, c\, A^* = (\text{Der}\, c\, A) @ \; A^*$$

- If "" $\in A$, then $\text{Der}\, c\, (A \; @ \; B) = (\text{Der}\, c\, A) \; @ \; B \cup (\text{Der}\, c\, B)$

- If "" $\notin A$, then $\text{Der}\, c\, (A \; @ \; B) = (\text{Der}\, c\, A) \; @ \; B$