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:

$LETTER \cdot (LETTER + DIGIT + "\_")^*$

NUM:

$(NONZERODIGIT \cdot DIGIT^*) + "0"$

OP:

"+"

COMMENT:

$"/*" \cdot (ALL^* \cdot "*/" \cdot ALL^*) \cdot "*/"$

```
"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)

Negation

Assume you have an alphabet consisting of the letters a, b and c only. Find a regular expression that matches all strings except ab, ac and cba.

Deterministic Finite Automata

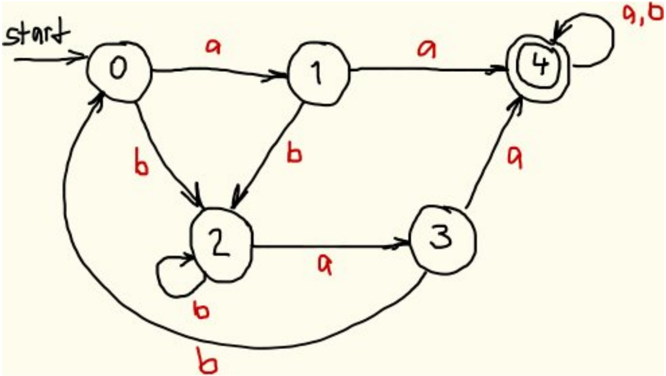
A deterministic finite automaton consists of:

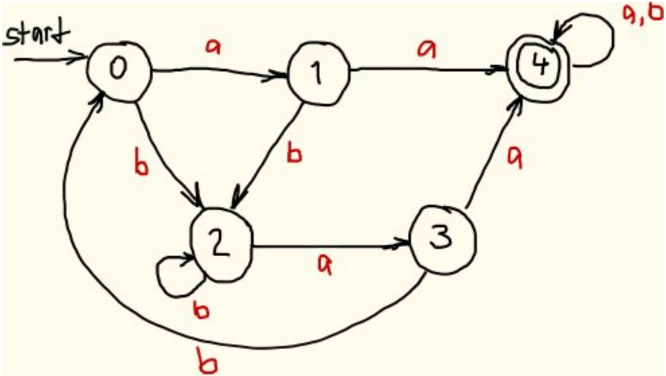
- a finite set of states
- one of these states is the start state
- some states are accepting states, and
- there is transition function

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

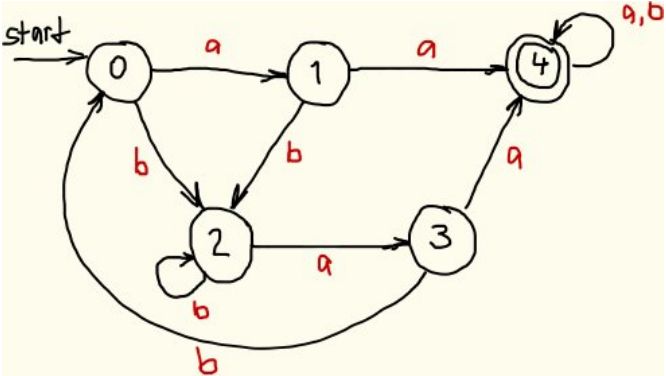
  this function might not always be defined everywhere

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


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



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



for this automaton δ is the function

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

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

$$\begin{aligned}\hat{\delta}(q, \texttt{""}) &= q \\ \hat{\delta}(q, c :: s) &= \hat{\delta}(\delta(q, c), s)\end{aligned}$$

Whether a string s is accepted by A?

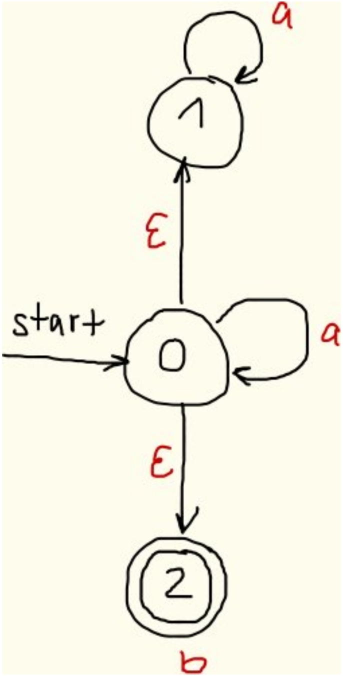
$$\hat{\delta}(q_0, s) \in F$$

Non-Deterministic Finite Automata

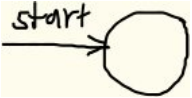
A non-deterministic finite automaton consists again of:

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

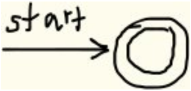
$$(q_1, a) \rightarrow q_2 \qquad (q_1, \epsilon) \rightarrow q_2$$
$$(q_1, a) \rightarrow q_3$$


a
1
ε
start
0
a
ε
2
b

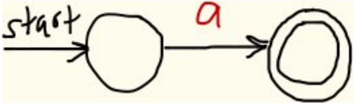
$\varnothing$



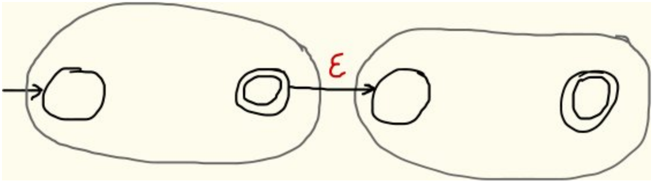
ϵ

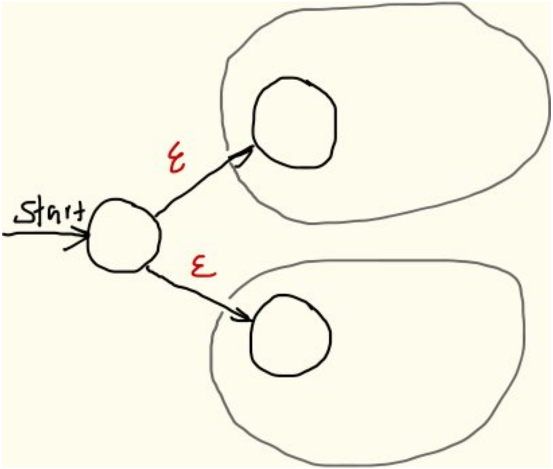


c



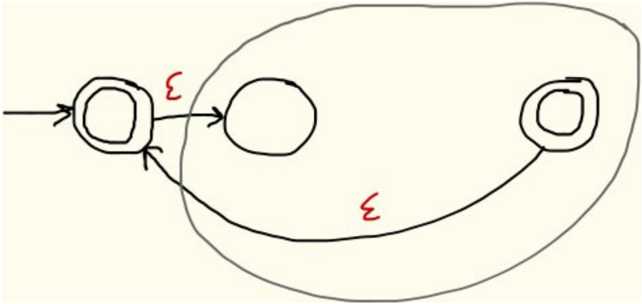
$\mathbf{r_1 \cdot r_2}$



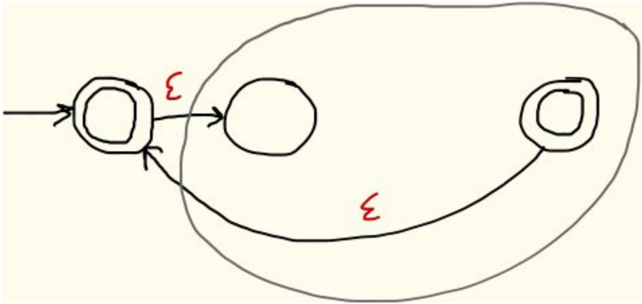


$r_1 + r_2$

r^*

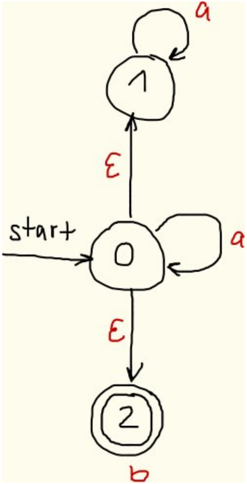


r^*



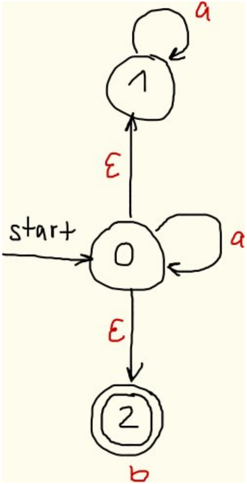
Why can't we just have an epsilon transition from the accepting states to the starting state?

Subset Construction



	a	b
$\varnothing$	$\varnothing$	$\varnothing$
$\{0\}$	$\{0, 1, 2\}$	$\{2\}$
$\{1\}$	$\{1\}$	$\varnothing$
$\{2\}$	$\varnothing$	$\{2\}$
$\{0, 1\}$	$\{0, 1, 2\}$	$\{2\}$
$\{0, 2\}$	$\{0, 1, 2\}$	$\{2\}$
$\{1, 2\}$	$\{1\}$	$\{2\}$
$\{0, 1, 2\}$	$\{0, 1, 2\}$	$\{2\}$

Subset Construction



	a	b
$\varnothing$	$\varnothing$	$\varnothing$
$\{0\}$	$\{0,1,2\}$	$\{2\}$
$\{1\}$	$\{1\}$	$\varnothing$
$\{2\}$ *	$\varnothing$	$\{2\}$
$\{0,1\}$	$\{0,1,2\}$	$\{2\}$
$\{0,2\}$ *	$\{0,1,2\}$	$\{2\}$
$\{1,2\}$ *	$\{1\}$	$\{2\}$
s: $\{0,1,2\}$ *	$\{0,1,2\}$	$\{2\}$

Regular Languages

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

or equivalently

A language is regular iff there exists a deterministic finite automaton that recognises all its strings.

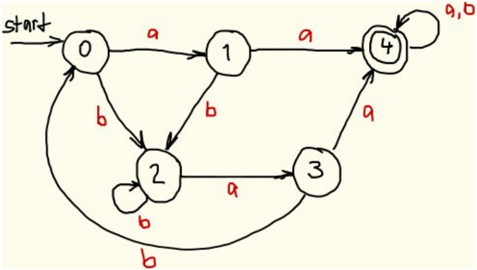
Regular Languages

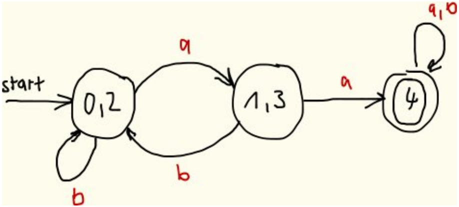
A language is regular iff there exists a regular expression that recognises all its strings.

or equivalently

A language is regular iff there exists a deterministic finite automaton that recognises all its strings.

Why is every finite set of strings a regular language?





minimal automaton

Given the function

$$rev(\varnothing) \stackrel{\text{def}}{=} \varnothing$$
$$rev(\epsilon) \stackrel{\text{def}}{=} \epsilon$$
$$rev(c) \stackrel{\text{def}}{=} c$$
$$rev(r_1 + r_2) \stackrel{\text{def}}{=} rev(r_1) + rev(r_2)$$
$$rev(r_1 \cdot r_2) \stackrel{\text{def}}{=} rev(r_2) \cdot rev(r_1)$$
$$rev(r^*) \stackrel{\text{def}}{=} rev(r)^*$$

and the set

$$Rev\ A \stackrel{\text{def}}{=} \{s^{-1} \mid s \in A\}$$

prove whether

$$L(rev(r)) = Rev(L(r))$$

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

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

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

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

Assume you have an alphabet consisting of the letters a, b and c only. (a) Find a regular expression that recognises the two strings ab and ac. (b) Find a regular expression that matches all strings except these two strings. Note, you can only use regular expressions of the form

$$r ::= \varnothing \mid \epsilon \mid c \mid r_1 + r_2 \mid r_1 \cdot r_2 \mid r^*$$