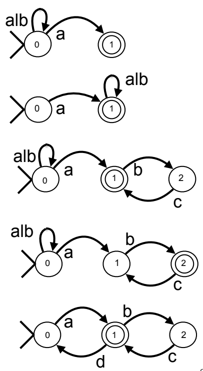
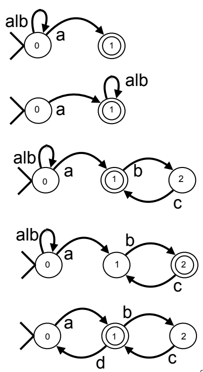
regex Arden’s lemma

- NFA DFA regex
- Arden’s lemma helps us find a regex for an NFA

Informatics 1
School of Informatics, University of Edinburgh





(a|b)*a

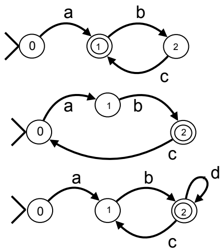
a(a|b)*

(a|b)*a(bc)*

(a|b)*ab(cb)*

a(da|bc)*

Is there a regular expression for every FSM?

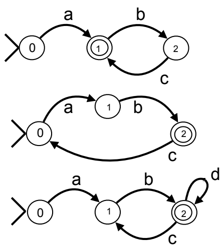


?

?

?

Is there a regular expression for every FSM?



a(bc)*

ab(cab)*

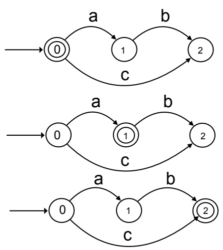
ab(cb | d)*

Is there a regular expression for every FSM?

L_0

L_1

L_2

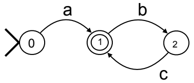


Let L_i be the language accepted if i is the accepting state

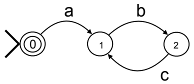
$L_0 = \varepsilon$
$L_1 = L_0\, a$
$L_2 = L_1\, b \mid L_0\, c$

$L_2 = L_0\, a\, b \mid \varepsilon\, c$
$L_2 = \varepsilon\, a\, b \mid \varepsilon\, c$
$L_2 = a\, b \mid c$

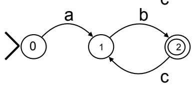
Is there a regular expression for every FSM?



$L_1 =$



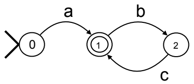
$L_0 = \varepsilon$



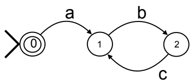
$L_2 =$

Let L_i be the language accepted if i is the accepting state

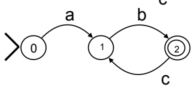
Is there a regular expression for every FSM?



$L_1 = L_0\ a \mid L_2\ c$

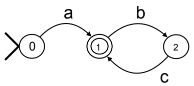


$L_0 = \varepsilon$

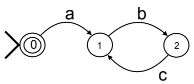


$L_2 = L_1\ b$

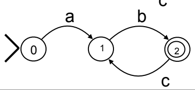
Is there a regular expression for every FSM?



$L_1 = L_0\ a \mid L_2\ c$
$\quad = a \mid L_1\ bc$

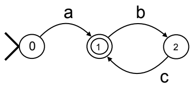


$L_0 = \varepsilon$



$L_2 = L_1\ b$

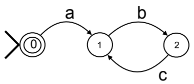
Is there a regular expression for every FSM?



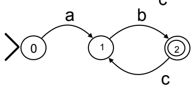
$L_1 = L_0\ a\ |\ L_2\ c$

$\quad = a\ |\ L_1\ bc$

$L_1 = a(bc)^*$



$L_0 = \varepsilon$



$L_2 = L_1\ b$

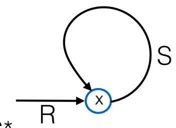
Arden's Lemma

If R and S are regular expressions then the equation

$\mathbf{X} = R\ |\ \mathbf{X}\ S$

has a solution $\mathbf{X} = R\ S^*$

If $\varepsilon \notin L(S)$ then this solution is unique.

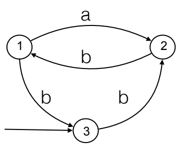


Is there a regular expression for every FSM?

$L_1 = L_2\ b$

$L_2 = L_3\ b\ |\ L_1\ a$

$L_3 = \varepsilon\ |\ L_1\ b$



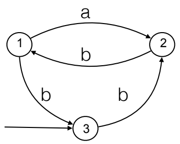
Is there a regular expression for every FSM?

$L_1 = L_2\, b$

$L_2 = L_3\, b \mid L_1\, a$

$L_3 = \varepsilon \mid L_1\, b$

$\quad = \varepsilon \mid L_2\, b\, b$



$L_2 = (\varepsilon \mid L_2\, b\, b)\, b \mid L_2\, b\, a$

$\quad = b \mid L_2\, b\, b\, b \mid L_2\, b\, a$

$\quad = b \mid L_2\, (b\, b\, b \mid b\, a)$

Arden's Lemma

If R and S are regular expressions then the equation

$$\mathbf{X} = R \mid \mathbf{X}\, S$$

has a solution $\mathbf{X} = R\, S^*$

If $\varepsilon \notin L(S)$ then this solution is unique.

$L_2 = b \mid L_2\, (b\, b\, b \mid b\, a)$

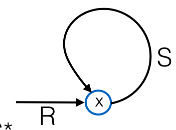
$L_2 = b\, (b\, b\, b \mid b\, a)^*$

Arden's Lemma

If R and S are regular expressions then the equation

$\mathbf{X} = R \mid \mathbf{X}\, S$

has a solution $\mathbf{X} = R\, S^*$



If $\varepsilon \notin L(S)$ then this solution is unique.

L0 = L1 a | L2 b | ε
L1 = L0 b | L2 a
L2 = L1 b | L0 a

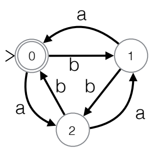
L1 = L0 b | L1 ba | L0 aa
L1 = L0 (b | aa) | L1 ba
L1 = L0 (b | aa) (ba)*

L2 = L0 bb | L2 ab | L0 a
L2 = L0 (bb | a) | L2 ab
L2 = L0 (bb | a) (ab)*

L0 = ε | L0 (b | aa) (ba)* a | L0 (bb | a) (ab)* b
L0 = ε | L0 ((b | aa) (ba)* a) | ((bb | a) (ab)* b)
L0 = ((b (ba)* a) | (aa (ba)* a) | (bb (ab)* b) | (a (ab)* b))*

Since
a(ab)*aa = aa(ba)*a
and
b(ba)*bb = bb(ab)*b
these solutions agree :-)

| (a(ab)*b
a(ab)*aa
b(ba)*bb
b(ba)*a) *



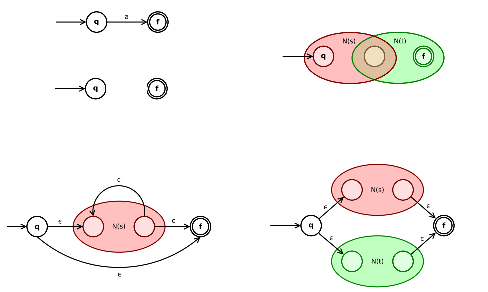
a(ab)*b = ab | aa(ba)*bb

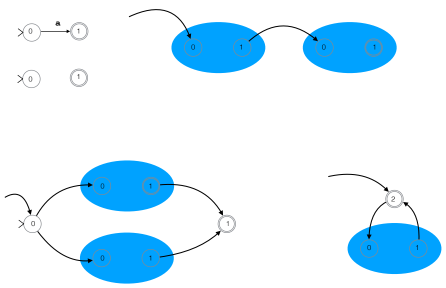
| (ab
aa(ba)*a
aa(ba)*bb
b(ba)*bb
b(ba)*a) *

(ab | (aa | b) (ba)* (a | bb)) *

Lecture 17
NFA DFA regex
Michael Fourman

NFA DFA regex
language — corresponding to NFA, DFA, regex
trace for a string in NFA or DFA

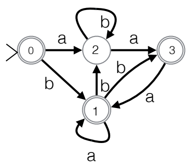




Definition FSM

finite state automaton FSM

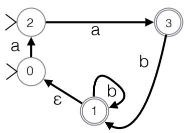
states — a set of states
sigma — a set of symbols
delta ⊆ (states × sigma × states)
start ⊆ states — starting states
accept ⊆ states — accepting states



Definition ε-FSM or DFA

finite state automaton FSM
with ε-transitions

states — a set of states
sigma — a set of symbols
delta ⊆ (states × sigma × states)
eps ⊆ (states × states)
start ⊆ states — starting states
accept ⊆ states — accepting states



Definition DFA

is a finite state automaton
(FSA, without ε)

states — a set of states
sigma — a set of symbols
delta ⊆ (states × sigma × states)
start ⊆ states — starting states
accept ⊆ states — accepting states

A deterministic machine has

- no ε-transitions
- exactly one starting state
- for each (state, symbol) pair, (q, s)
  exactly one transition of the form (q, s, q')

For any FSM DFA NFA, with or without epsilon this is the definition

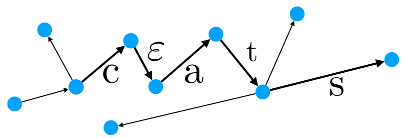
A **trace** from q to q' consists of

n transitions $q_i \xrightarrow{s_i} q_{i+1}$ for $i < n$

with $q = q_0$ and $q_n = q'$

Each trace determines a string, $\sigma \in \Sigma^*$
consisting of the concatenation of all the non-ε symbols s_i.

σ `= [` s_i `| i < n,` $s_i \neq \varepsilon$ `]`



For any FSM DFA NFA, with or without epsilon this is the definition

A **trace** from q to q' consists of

n transitions $q_i \xrightarrow{s_i} q_{i+1}$ for $i < n$

with $q = q_0$ and $q_n = q'$

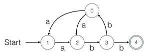
Each trace determines a string, $\sigma \in \Sigma^*$
consisting of the concatenation of all the non-ε symbols s_i.

σ `= [` s_i `| i < n,` $s_i \neq \varepsilon$ `]`

If q is a starting state and q' is an accepting state we say the machine **accepts** σ.

When we check whether a machine accepts a string we use various algorithms
but ultimately, this is the definition.

6. (a) Which of the following strings are accepted by the NFA in the diagram? (The start state is indicated by an arrow and the accepting state by a double border.)



   i. abb
   ii. abbabbabbaaabb
   iii. abbabbaabbabbabb
   iv. abbabaabbabbabb [*3 marks*]

   (b) Write a regular expression for the language accepted by this NFA. [*3 marks*]

   (c) Draw a DFA that accepts the same language. Label the states of your DFA to make clear their relationship to the states of the original NFA. [*10 marks*]

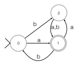
   (d) For each of the following regular expressions, draw a non-deterministic finite state machine that accepts the language described by the regular expression.
   i. x^*y
   ii. $(x^*|y)$
   iii. $(x^*y)^*$ [*9 marks*]

---

5. Each diagram shows an FSM. In each case give a regular expression for the language accepted by the FSM, make a mark in the check box against each string that it accepts (and no mark against those strings it does not accept), make a mark in the DFA check box if it is deterministic, and draw an equivalent DFA if it is not.

[*4 marks*]

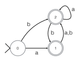
(a)



aab ☐
aba ☐
bab ☐
aaa ☐
bbb ☐
DFA ☐ regex:

[*4 marks*]

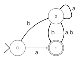
(b)



aab ☐
aba ☐
bab ☐
aaa ☐
bbb ☐
DFA ☐ regex:

[*4 marks*]

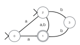
(c)



aab ☐
aba ☐
bab ☐
aaa ☐
bbb ☐
DFA ☐ regex:

[*4 marks*]

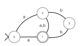
(d)



aab ☐
aba ☐
bab ☐
aaa ☐
bbb ☐
DFA ☐ regex:

[*4 marks*]

(e)



aab ☐
aba ☐
bab ☐
aaa ☐
bbb ☐
DFA ☐ regex: