

Automata and Formal Languages (5)

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

Deterministic Finite Automata

A DFA $A(Q, q_0, F, \delta)$ consists of:

- a finite set of states Q
- one of these states is the start state q_0
- some states are accepting states F
- there is transition function δ

Deterministic Finite Automata

A DFA $A(Q, q_0, F, \delta)$ consists of:

- a finite set of states Q
- one of these states is the start state q_0
- some states are accepting states F
- there is transition function δ

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

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

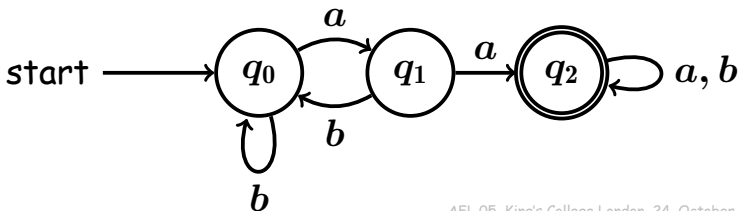
Deterministic Finite Automata

A DFA $A(Q, q_0, F, \delta)$ consists of:

- a finite set of states Q
- one of these states is the start state q_0
- some states are accepting states F
- there is transition function δ

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

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



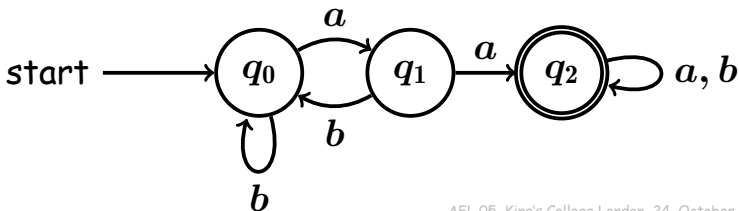
Deterministic Finite Automata

A DFA $A(Q, q_0, F, \delta)$ consists of:

- a finite set of states Q
- one of these states is the start state q_0
- some states are accepting states F
- there is transition function δ

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

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



Deterministic Finite Automata

A DFA $A(Q, q_0, F, \delta)$ consists of:

- a finite set of states Q
- one of these states is the start state q_0
- some states are accepting states F
- there is transition function δ

$$\hat{\delta}(q, \epsilon) = q$$

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

$$L(A) \stackrel{\text{def}}{=} \{s \mid \hat{\delta}(q_0, s) \in F\}$$

Non-Deterministic Finite Automata

An NFA $A(Q, q_0, F, \delta)$ consists again of:

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

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

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

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

Non-Deterministic Finite Automata

An NFA $A(Q, q_0, F, \delta)$ consists again of:

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

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

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

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

A string s is accepted by an NFA, if there is a "lucky" sequence to an accepting state.

Last Week

Last week I showed you

- an algorithm for automata minimisation
- an algorithm for transforming a regular expression into an NFA
- an algorithm for transforming an NFA into a DFA (subset construction)

This Week

Go over the algorithms again, but with two new things and ...

- with the example: what is the regular expression that accepts every string, except those ending in aa ?
- Go over the proof for $L(\text{rev}(r)) = \text{Rev}(L(r))$.
- Anything else so far.

Proofs By Induction

- P holds for \emptyset , ϵ and c
- P holds for $r_1 + r_2$ under the assumption that P already holds for r_1 and r_2 .
- P holds for $r_1 \cdot r_2$ under the assumption that P already holds for r_1 and r_2 .
- P holds for r^* under the assumption that P already holds for r .

$$P(r) : L(\text{rev}(r)) = \text{Rev}(L(r))$$

What is the regular expression that accepts every string, except those ending in `aa`?

What is the regular expression that accepts every string, except those ending in **aa**?

$(a + b)^*ba$

$(a + b)^*ab$

$(a + b)^*bb$

What is the regular expression that accepts every string, except those ending in **aa**?

$(a + b)^*ba$

$(a + b)^*ab$

$(a + b)^*bb$

a

$''$

What is the regular expression that accepts every string, except those ending in **aa**?

$(a + b)^*ba$

$(a + b)^*ab$

$(a + b)^*bb$

a

$''$

What are the strings to be avoided?

What is the regular expression that accepts every string, except those ending in **aa**?

$(a + b)^*ba$

$(a + b)^*ab$

$(a + b)^*bb$

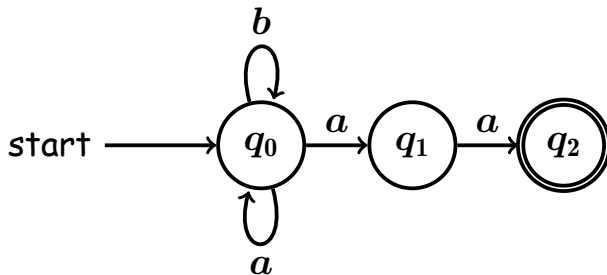
a

$''$

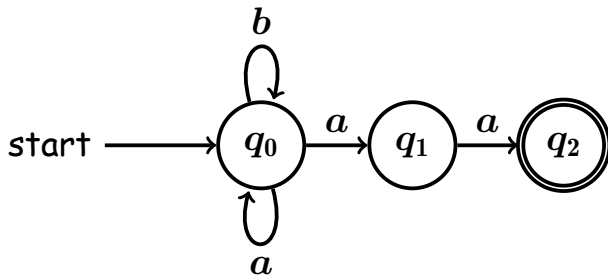
What are the strings to be avoided?

$(a + b)^*aa$

An NFA for $(a + b)^*aa$



An NFA for $(a + b)^*aa$



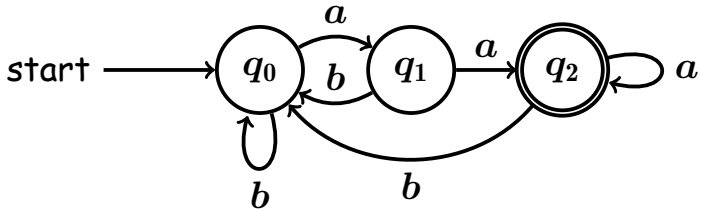
Minimisation for DFAs

Subset Construction for NFAs

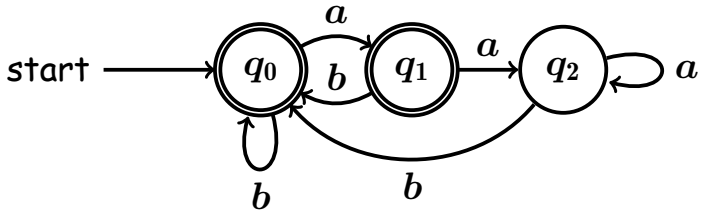
DFA Minimisation

- 1 Take all pairs (q, p) with $q \neq p$
- 2 Mark all pairs that accepting and non-accepting states
- 3 For all unmarked pairs (q, p) and all characters c tests whether
 $(\delta(q,c), \delta(p,c))$
are marked. If yes, then also mark (q, p) .
- 4 Repeat last step until nothing changed.
- 5 All unmarked pairs can be merged.

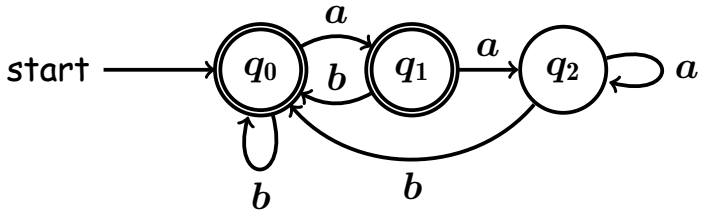
Minimal DFA $(a + b)^*aa$



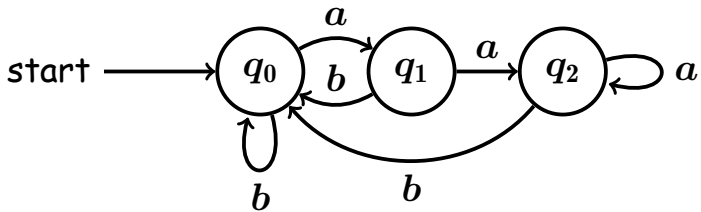
Minimal DFA **not** $(a + b)^*aa$

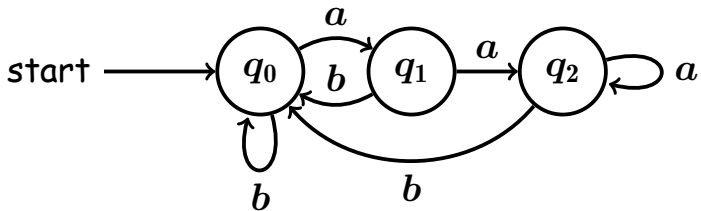


Minimal DFA **not** $(a + b)^*aa$



How to get from a DFA to a regular expression?

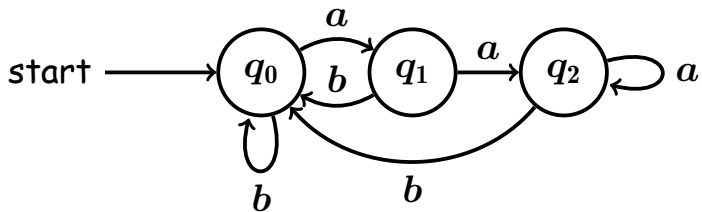


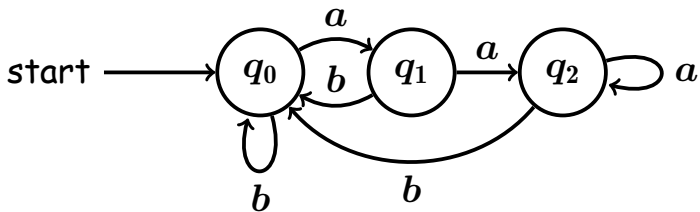


$$q_0 = 2q_0 + 3q_1 + 4q_2$$

$$q_1 = 2q_0 + 3q_1 + 1q_2$$

$$q_2 = 1q_0 + 5q_1 + 2q_2$$

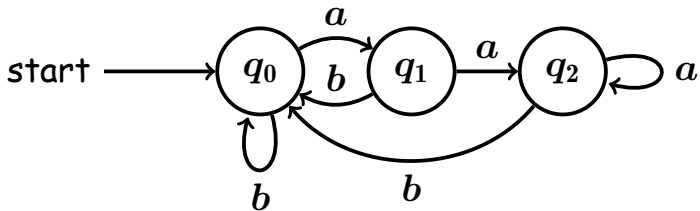




$$q_0 = \epsilon + q_0 b + q_1 b + q_2 b$$

$$q_1 = q_0 a$$

$$q_2 = q_1 a + q_2 a$$



$$q_0 = \epsilon + q_0 b + q_1 b + q_2 b$$

$$q_1 = q_0 a$$

$$q_2 = q_1 a + q_2 a$$

Arden's Lemma:

$$\text{If } q = q r + s \text{ then } q = s r^*$$

Algorithms on Automata

- Reg \rightarrow NFA: Thompson-McNaughton-Yamada method
- NFA \rightarrow DFA: Subset Construction
- DFA \rightarrow Reg: Brzozowski's Algebraic Method
- DFA minimisation: Hopcroft's Algorithm
- complement DFA