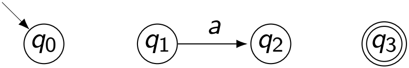
Section 2

(Deterministic) Finite Automata

1. (Deterministic) Finite Automata
   - Examples
   - FA definition
   - Boolean operations
   - Decision problems
   - Distinguishing strings
   - Equivalence classes
   - Minimization
   - Pumping lemma

From lecture 1:



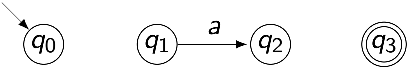
Example

$L_1 = \{ x \in \{a, b\}^* \mid x \text{ ends with } aa \}$

. . .

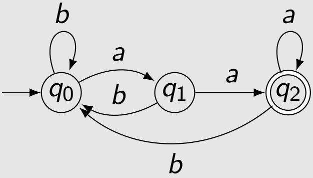
[M] E. 2.1

From lecture 1:



Example

$L_1 = \{ x \in \{a, b\}^* \mid x \text{ ends with } aa \}$



δ	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

[M] E. 2.1

Example

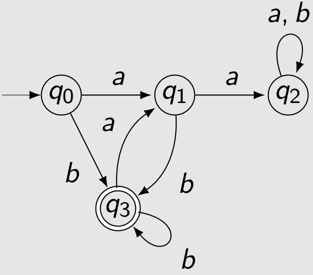
$L_2 = \{ x \in \{a, b\}^* \mid x \text{ ends with } b \text{ and does not contain } aa \}$

...

[M] E. 2.3

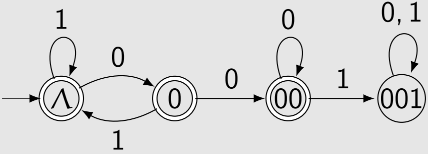
Example

$L_2 = \{ x \in \{a, b\}^* \mid x \text{ ends with } b \text{ and does not contain } aa \}$



[M] E. 2.3

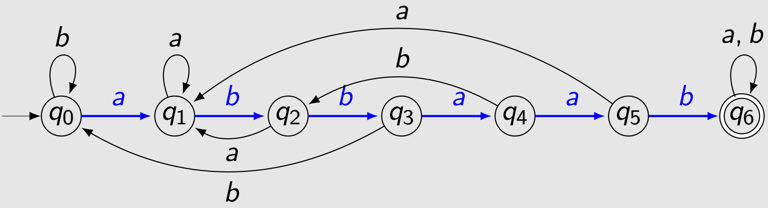
Example (Strings not containing 001)



[L] E 2.4

Example (Similar to Knuth-Morris-Pratt string search)

$L_3 = \{ x \in \{a, b\}^* \mid x \text{ contains the substring } abbaab \}$



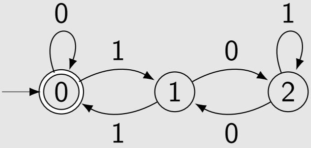
[M] E. 2.5

$w \in \{0, 1\}^* \longrightarrow val(w) \in \mathbb{N}$

$val(w0) = \ldots$

$val(w1) = \ldots$

Example



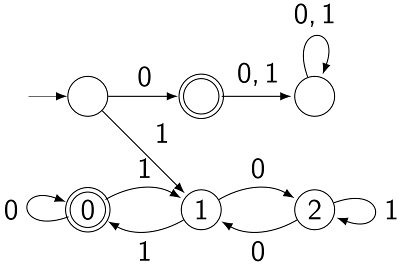
δ	0	1
x	$2x$	$2x+1$
0	0	1
1	2	0
2	1	2

$w \in \{0,1\}^* \longrightarrow val(w) \in \mathbb{N}$

$val(w0) = 2 \cdot val(w)$
$val(w1) = 2 \cdot val(w) + 1$

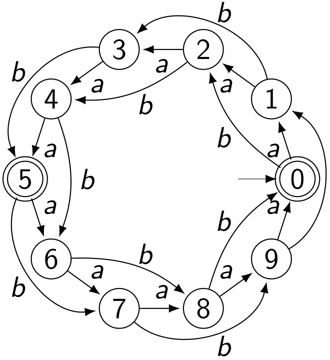
states represent $val(w)$ modulo 3

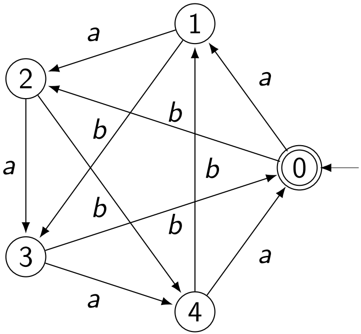
Disallows leading 0's in binary representations, e.g., 0001, and the null string.



[M] E. 2.7

$\{ x \in \{a, b\}^* \mid n_a(x) + 2n_b(x) \equiv 0 \mod 5 \}$



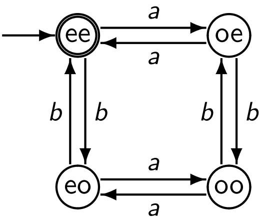


☒cs.SE Planar regular languages

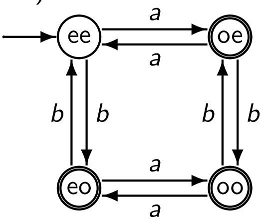
A student once asked if all finite automata can be drawn without crossing transitions. The automaton to the right has the form of K_5 (the complete graph on five nodes), which is known to be non-planar.

The same language can also be accepted a planar automaton (to the left). There are, however, languages that do not have a planar automaton.

2.1(g) All strings over $\{a, b\}$ in which both the number of a's and the number of b's is even.



2.1(g2) All strings over $\{a, b\}$ in which either the number of a's or the number of b's is odd (or both).



[M] Exercise 2.1(g)

Definition (FA)

[deterministic] finite automaton 5-tuple $M = (Q, \Sigma, q_0, A, \delta)$,

– Q finite set *states*;

– Σ finite *input alphabet*;

– $q_0 \in Q$ *initial* state;

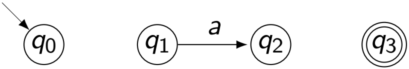
– $A \subseteq Q$ *accepting* states;

– $\delta : Q \times \Sigma \to Q$ *transition function*.

[M] D 2.11 Finite automaton

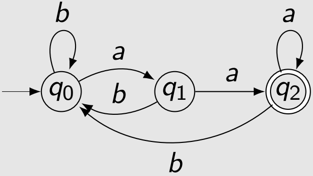
[L] D 2.1 Deterministic finite accepter, has 'final' states

From lecture 1:



Example

$L_1 = \{\ x \in \{a, b\}^* \mid x \text{ ends with } aa\ \}$



δ	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

[M] E. 2.1

Definition (FA)

[deterministic] finite automaton   5-tuple $M = (Q, \Sigma, q_0, A, \delta)$,

- Q finite set   *states*;
- Σ finite *input alphabet*;
- $q_0 \in Q$ *initial* state;
- $A \subseteq Q$ *accepting* states;
- $\delta : Q \times \Sigma \to Q$ *transition function*.

[M] D 2.11 Finite automaton

[L] D 2.1 Deterministic finite accepter, has 'final' states

FA $M = (Q, \Sigma, q_0, A, \delta)$

Definition

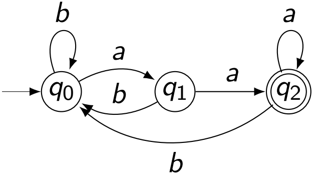
extended transition function $\delta^* : Q \times \Sigma^* \to Q$, such that
– $\delta^*(q, \Lambda) = q$ for $q \in Q$
– $\delta^*(q, y\sigma) = \delta(\ \delta^*(q, y), \sigma\ )$ for $q \in Q, y \in \Sigma^*, \sigma \in \Sigma$

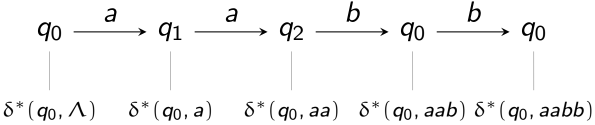
[M] D 2.12 [L] p.40/1

Theorem

$q = \delta^*(p, w)$ *iff there is a path in [the transition graph of] M from p to q with label w.*

[L] Th 2.1





$\delta^*(q_0, aabb) = q_0$:

$$\delta^*(q_0, \Lambda) = q_0$$

$$\delta^*(q_0, a) = \delta^*(q_0, \Lambda\, a) = \delta(\delta^*(q_0, \Lambda), a) = \delta(q_0, a) = q_1$$

$$\delta^*(q_0, aa) = \delta(\delta^*(q_0, a), a) = \delta(q_1, a) = q_2$$

$$\delta^*(q_0, aab) = \delta(\delta^*(q_0, aa), b) = \delta(q_2, b) = q_0$$

$$\delta^*(q_0, aabb) = \delta(\delta^*(q_0, aab), b) = \delta(q_0, b) = q_0$$

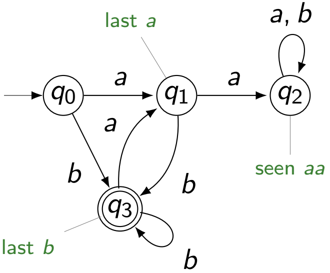
Definition

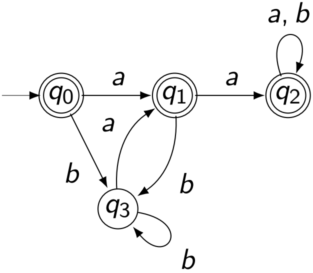
Let $M = (Q, \Sigma, q_0, A, \delta)$ be an FA, and let $x \in \Sigma^*$. The string x is *accepted* by M if $\delta^*(q_0, x) \in A$.
The *language accepted* by $M = (Q, \Sigma, q_0, A, \delta)$ is the set
$L(M) = \{ x \in \Sigma^* \mid x \text{ is accepted by } M \}$

[M] D 2.14 [L] D 2.2

Seen previously:

$L_2 = \{ x \in \{a, b\}^* \mid x \text{ ends with } b \text{ and does not contain } aa \}$





$$\neg(P \wedge Q) = \neg P \vee \neg Q$$

$L_2^c = \{ x \in \{a, b\}^* \mid x \text{ does not end with } b \text{ or contains } aa \}$

Construction

FA $M = (Q, \Sigma, q_0, A, \delta)$,

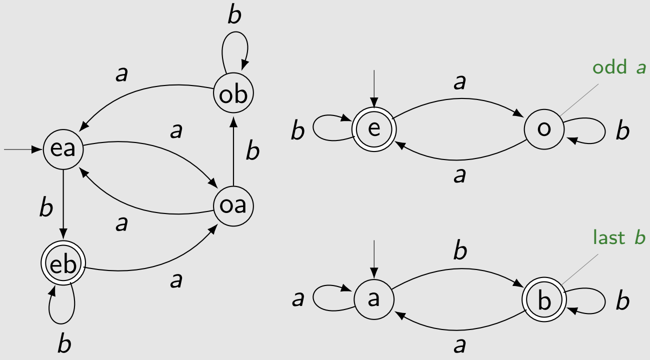
let $M^c = (Q, \Sigma, q_0, Q - A, \delta)$

Theorem

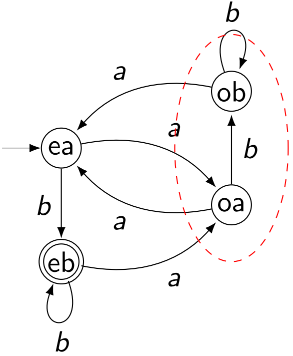
$L(M^c) = \Sigma^* - L(M)$

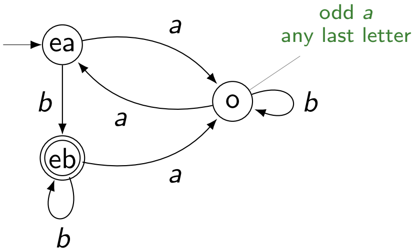
Proof. . .

Example (Even number of *a*, and ending with *b*)



Even number of *a* and ending with *b*





FA $M_i = (Q_i, \Sigma, q_i, A_i, \delta_i) \quad i = 1, 2$

Product construction

construct FA $M = (Q, \Sigma, q_0, A, \delta)$ such that

– $Q = Q_1 \times Q_2$

– $q_0 = (q_1, q_2)$

– $\delta(\ (p, q), \sigma) = (\ \delta_1(p, \sigma), \delta_2(q, \sigma)\ )$

– A as needed

Theorem (2.15 Parallel simulation)

– $A = \{(p, q) \mid p \in A_1 \text{ or } q \in A_2\}$, *then* $L(M) = L(M_1) \cup L(M_2)$

– $A = \{(p, q) \mid p \in A_1 \text{ and } q \in A_2\}$, *then* $L(M) = L(M_1) \cap L(M_2)$

– $A = \{(p, q) \mid p \in A_1 \text{ and } q \notin A_2\}$, *then* $L(M) = L(M_1) - L(M_2)$

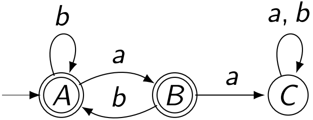
Proof. . .

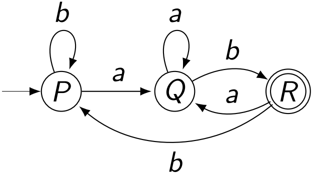
[M] Sect 2.2

Exercise 2.11.
Use induction to show that for every $x \in \Sigma^*$ and every $(p, q) \in Q$,
$\delta^*((p, q), x) = (\delta_1^*(p, x), \delta_2^*(q, x))$

Example: intersection '<u>and</u>' (product construction)

not substring *aa*

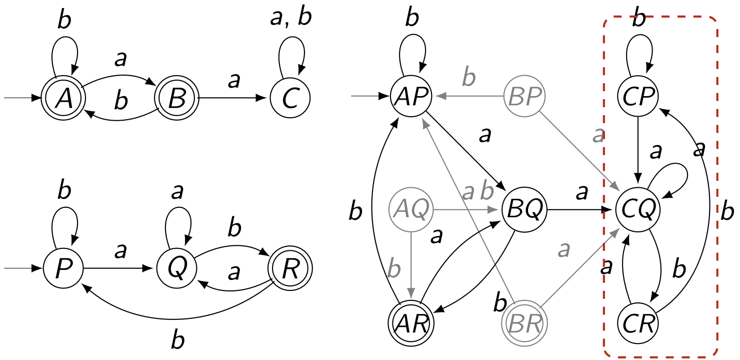




ends with *ab*

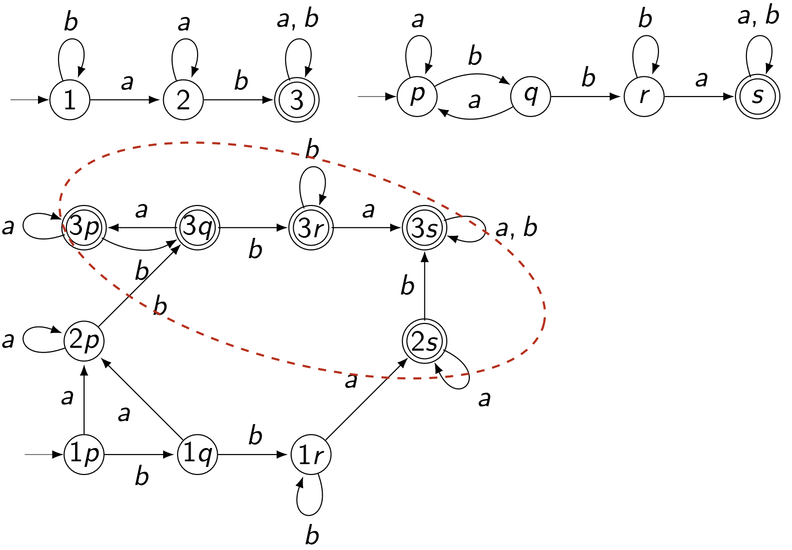
[M] E 2.16

not substring *aa*



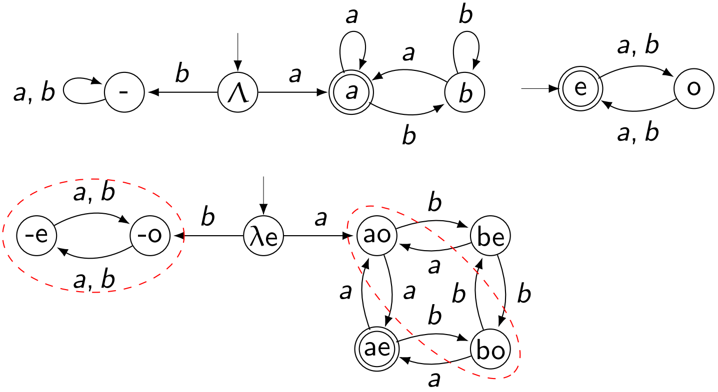
ends with *ab*

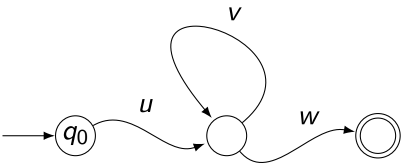
[M] E 2.16



[M] E. 2.18, see also ↪subset construction

$L = \{ w \in \{a,b\}^* \mid w \text{ starts and ends with an } a, \text{ and } |w| \text{ is even} \}$





[M] Fig. 2.28

Pumping lemma for regular languages

Regular language is language accepted by an FA.

Theorem

Suppose L is a language over the alphabet Σ. If L is accepted by a finite automaton M, and if n is the number of states of M, then

$\forall$ *for every $x \in L$*
satisfying $|x| \geqslant n$

$\exists$ *there are three strings u, v, and w,*
such that $x = uvw$ and the following three conditions are true:
(1) $|uv| \leqslant n$,
(2) $|v| \geqslant 1$

$\forall$ *and (3) for all $m \geqslant 0$, $uv^m w$ belongs to L*

[M] Thm. 2.29

In other words:

Theorem

$\forall$ *For every regular language* L

$\exists$ *there exists a constant* $n \geqslant 1$

such that

$\forall$ *for every* $x \in L$

with $|x| \geqslant n$

$\exists$ *there exists a decomposition* $x = uvw$

with (1) $|uv| \leqslant n$,

and (2) $|v| \geqslant 1$

such that

$\forall$ *(3) for all* $m \geqslant 0$, $uv^m w \in L$

if $L = L(M)$ then $n = |Q|$.

[M] Thm. 2.29

In other words:

Theorem

If L *is a regular language, then*

$\exists$ *there exists a constant* $n \geqslant 1$

such that

$\forall$ *for every* $x \in L$

with $|x| \geqslant n$

$\exists$ *there exists a decomposition* $x = uvw$

with (1) $|uv| \leqslant n$,

and (2) $|v| \geqslant 1$

such that

$\forall$ *(3) for all* $m \geqslant 0$, $uv^m w \in L$

if $L = L(M)$ then $n = |Q|$.

Introduction to Logic: $p \to q \iff \neg q \to \neg p$

Theorem

If

$\forall$ *for every* $n \geqslant 1$

$\exists$ *there exists* $x \in L$

with $|x| \geqslant n$

such that

$\forall$ *for every decomposition* $x = uvw$

with (1) $|uv| \leqslant n$,

and (2) $|v| \geqslant 1$

$\exists$ *(3) there exists* $m \geqslant 0$,

such that

$uv^m w \notin L$

then L *is not a regular language.*

[M] Thm. 2.29

Example

$L = \{a^i b^i \mid i \geqslant 0\}$ is not accepted by FA.

[M] E 2.30

Proof: by contradiction

We prove that the language $L = \{a^i b^i \mid i \geqslant 0\}$ is not regular, by contradiction.

Assume that $L = \{a^i b^i \mid i \geqslant 0\}$ is accepted by FA with n states.
Take $x = a^n b^n$. Then $x \in L$, and $|x| = 2n \geqslant n$.
Thus there exists a decomposition $x = uvw$ such that $|uv| \leqslant n$ with v nonempty, and $uv^m w \in L$ for every m.
Whatever this decomposition is, v consists of a's only. Consider $m = 0$.
Deleting v from the string x will delete a number of a's. So $uv^0 w$ is of the form $a^{n'} b^n$ with $n' < n$.
This string is not in L; a contradiction. ($m \geqslant 2$ would also yield contradiction)

So, L is not regular.

Example

$L = \{a^i b^i \mid i \geqslant 0\}$ is not accepted by FA.

[M] E 2.30

$AeqB = \{x \in \{a, b\}^* \mid n_a(x) = n_b(x)\}$

Same argument, or closure properties

FA $M_i = (Q_i, \Sigma, q_i, A_i, \delta_i) \quad i = 1, 2$

Product construction

construct FA $M = (Q, \Sigma, q_0, A, \delta)$ such that

– $Q = Q_1 \times Q_2$

– $q_0 = (q_1, q_2)$

– $\delta(\ (p, q), \sigma) = (\ \delta_1(p, \sigma), \delta_2(q, \sigma)\ )$

– A as needed

Theorem (2.15 Parallel simulation)

– $A = \{(p, q) \mid p \in A_1$ *or* $q \in A_2\}$, *then* $L(M) = L(M_1) \cup L(M_2)$

– $A = \{(p, q) \mid p \in A_1$ *and* $q \in A_2\}$, *then* $L(M) = L(M_1) \cap L(M_2)$

– $A = \{(p, q) \mid p \in A_1$ *and* $q \notin A_2\}$, *then* $L(M) = L(M_1) - L(M_2)$

Proof. . .

[M] Sect 2.2

Exactly the same argument can be used (verbatim) to prove that $L =$ AeqB is not regular.

We can also apply closure properties of REG to see that AeqB is not regular, as follows.

Assume AeqB is regular. Then also AnBn $=$ AeqB $\cap\, a^* b^*$ is regular, as regular languages are closed under intersection.
This is a contradiction, as we just have argued that AnBn is not regular.
Thus, also AeqB is not regular.

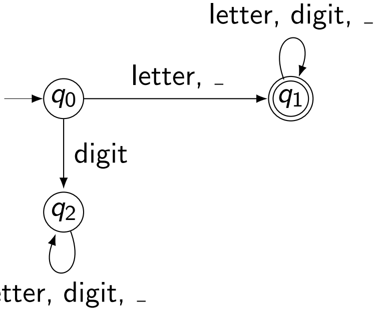
Issues:

- Which n? Can I just take $x = aababaabbab$?
- Which x? Some x may not yield a contradiction.
- Which decomposition uvw? Can I just take $u = a^{10}$, $v = a^{n-10}$, $w = b^n$?
- Which m? Some m may not yield a contradiction.

$L_1 = \{ w \mid w \text{ is a C-identifier} \}$

Legal C identifiers:

- sequence of letters, digits, underscores
- starts with a letter or an underscore



[L] E 1.16

$L_2 = \{ w \mid w \text{ is a C-style comment} \}$

L_2^c

C-identifiers (adjust to use the same alphabet: $\Sigma = \{ l, d, \_, *, / \}$) or C-style comments