Distinguishing strings

From lecture 3:

Definition

Let L be language over Σ, and let $x, y \in \Sigma^*$.
Then x, y are *distinguishable* wrt L (*L-distinguishable*),
if there exists $z \in \Sigma^*$ with

$$xz \in L \text{ and } yz \notin L \quad \text{or} \quad xz \notin L \text{ and } yz \in L$$

Such z *distinguishes* x and y wrt L.

Equivalent definition:
let $L/x = \{ z \in \Sigma^* \mid xz \in L \}$

x and y are L-*distinguishable* if $L/x \neq L/y$.
Otherwise, they are L-*indistinguishable*.

The strings in a set $S \subseteq \Sigma^*$ are *pairwise L-distinguishable*, if for every pair x, y of distinct strings in S, x and y are L-distinguishable.

Definition independent of FAs

[M] D 2.20

Strings with a in the nth symbol from the end

L_n the language of strings in $\{a, b\}^*$ with at least n symbols and an a in the nth position from the end

[M] E. 2.24

Strings with *a* in the *n*th symbol from the end

$$\begin{tikzcd}[nodes in empty cells]
 & \boxed{\boxed{aab}} \arrow[rrr, "b"] \arrow[dr, "a"] & & & \boxed{\boxed{abb}} \arrow[dd, "a"] \arrow[dr, "b"] & & {} \arrow[dl] \\
\boxed{\boxed{aaa}} \arrow[loop left, "a"] \arrow[ur, "b"] & & \boxed{\boxed{aba}} \arrow[dl, "a"] \arrow[r, bend right, "b"'] & \boxed{bab} \arrow[l, bend right, "a"'] \arrow[ur, "b"] & & \boxed{bbb} \arrow[loop right, "b"] \arrow[dl, "a"] & \\
 & \boxed{baa} \arrow[ul, "a"] \arrow[uu, "b"] & & & \boxed{bba} \arrow[lll, "a"] \arrow[ul, "b"] & &
\end{tikzcd}$$

[M] E. 2.24

From lecture 3:

Theorem

Suppose $M = (Q, \Sigma, q_0, A, \delta)$ is an FA accepting $L \subseteq \Sigma^*$.

If $x, y \in \Sigma^*$ are L-distinguishable, then $\delta^*(q_0, x) \neq \delta^*(q_0, y)$.

For every $n \geqslant 2$, if there is a set of n pairwise L-distinguishable strings in Σ^*, then Q must contain at least n states.

Hence, indeed: if $\delta^*(q_0, x) = \delta^*(q_0, y)$, then x and y are not L-distinguishable.

Proof. . .

[M] Thm 2.21

Theorem

For every language $L \subseteq \Sigma^*$,
if there is an infinite set S of pairwise L-distinguishable strings,
then L cannot be accepted by a finite automaton.

Proof. . .

[M] Thm 2.26

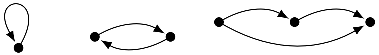
Pal

$Pal = \{x \in \{a, b\}^* \mid x = x^r\}$

[M] E. 2.27

Equivalence relation

R equivalence relation on A
– reflexive xRx for all . . .
– symmetric xRy then yRx
– transitive xRy and yRz then xRz



equivalence class $[x]_R = \{ y \in A \mid yRx \}$
short: $[x]$
partition A

[M] Sect. 1.3

Definition

For a language $L \subseteq \Sigma^*$, we define the relation $\equiv_L$ (an equivalence relation) on Σ^* as follows: for $x, y \in \Sigma^*$

$$x \equiv_L y \qquad \text{if and only if } x \text{ and } y \text{ are } L\text{-indistinguishable}$$

Equivalence relation...

right invariant $\quad x \equiv_L y$ implies $xz_1 \equiv_L yz_1$

Book uses I_L for $\equiv_L$

Example

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

L/x for $x = \Lambda, a, b, aa \ldots$
$L/\Lambda = \mathrm{L}$
$L/a = \{a\} \cup L$
$L/b = L$
$L/aa = \{\Lambda, a\} \cup L$

Equivalence classes / partitioning of
$\{a, b\}^* = \{\Lambda, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, \ldots\} \ldots$

Example

Equivalence classes of $\equiv_L$, where $L = AnBn = \{a^i b^i \mid i \geqslant 0\}$

[M] E 2.37

Example

Equivalence classes of $\equiv_L$, where $L = AnBn = \{a^i b^i \mid i \geqslant 0\}$

$\{\Lambda\}, \{a\}, \{a^2\}, \{a^3\}, \ldots$
$\{a^i b^i \mid i \geqslant 1\}$
$\{a^{i+1} b^i \mid i \geqslant 1\}, \{a^{i+2} b^i \mid i \geqslant 1\}, \{a^{i+3} b^i \mid i \geqslant 1\}, \ldots$
$\{x \in \{a, b\}^* \mid x$ is not a prefix of any element of $L\}$
$= \{b, ba, bb, aba, abb, baa, \ldots\}$

Infinitely many equivalence classes

quotients

- $L/a^i = \{ a^k b^{i+k} \mid k \geqslant 0 \}$
- $L/a^{i+k} b^i = \{ b^k \}$ $\quad i > 0, k \geqslant 0$
- $L/a^i b^j = L/xbay = \varnothing$ $\quad j > i$

[M] E 2.37

Example

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

L/x for $x = \Lambda, a, b, aa \ldots$

Equivalence classes / partitioning of
$\{a, b\}^* = \{\Lambda, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, \ldots\}$:

$\{\Lambda, b, ab, bb, aab, abb, \ldots\}$
$\{a, ba, aba, \ldots\}$
$\{aa, aaa, baa, \ldots\}$

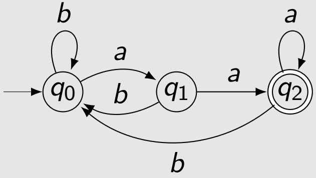
Finitely many equivalence classes

x ends with aa

From lecture 1:

Example

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


$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[loop above, "b"] \arrow[r, bend left, "a"] & q_1 \arrow[l, bend left, "b"] \arrow[r, "a"] & q_2 \arrow[loop above, "a"] \arrow[ll, bend left=40, "b"]
\end{tikzcd}$$

[M] E. 2.1

State q in FA $\approx$ $L_q = \{x \in \Sigma^* \mid \delta^*(q_0, x) = q\}$

From lecture 3:

Theorem

Suppose $M = (Q, \Sigma, q_0, A, \delta)$ is an FA accepting $L \subseteq \Sigma^*$.

If $x, y \in \Sigma^*$ are L-distinguishable, then $\delta^*(q_0, x) \neq \delta^*(q_0, y)$.

For every $n \geqslant 2$, if there is a set of n pairwise L-distinguishable strings in Σ^*, then Q must contain at least n states.

Proof. . .

[M] Thm 2.21

In other words: if $\delta^*(q_0, x) = \delta^*(q_0, y)$, then x, y are L-indistinguishable.
Each L_q is subset of equivalence class

Myhill-Nerode

Theorem

If $L \subseteq \Sigma^*$ can be accepted by a finite automaton, then the set Q_L of equivalence classes of the relation $\equiv_L$ is finite.

Conversely, if the set Q_L is finite,
the finite automaton $M_L = (Q_L, \Sigma, q_0, A, \delta)$ accepts L, where
$q_0 = \ldots$
$A = \ldots$
$\delta([x], \sigma) = \ldots$

[M] Thm 2.36

Myhill-Nerode

Theorem

If $L \subseteq \Sigma^*$ can be accepted by a finite automaton, then the set Q_L of equivalence classes of the relation $\equiv_L$ is finite.

Conversely, if the set Q_L is finite,
the finite automaton $M_L = (Q_L, \Sigma, q_0, A, \delta)$ accepts L, where
$q_0 = [\Lambda]$
$A = \{q \in Q_L \mid q \subseteq L\}$
$\delta([x], \sigma) = [x\sigma]$

Finally, M_L has the fewest states of any FA accepting L.

Note:
If $x \in L$, then $[x] \subseteq L$ (L is union of equivalence classes)
Right invariant $\quad x \equiv_L y$ implies $x\sigma \equiv_L y\sigma$

[M] Thm 2.36

Theorem

For every language $L \subseteq \Sigma^*$,
if there is an infinite set S of pairwise L-distinguishable strings,
then L cannot be accepted by a finite automaton.

Proof. . .

[M] Thm 2.26

Minimizing states

1. Remove unreachable states
2. Merge ‘equivalent’ states

Equivalent states

Equivalence relation $\equiv_L$ induces equivalence relation on states
Each L_q is subset of equivalence class
L_p and L_q may be subset of same equivalent class
$p \equiv q \iff L_p$ and L_q are subset of same equivalent class
$p \not\equiv q \iff$ for some $z \in \Sigma^*$ exactly one of $\delta^*(p, z)$ and $\delta^*(q, z)$ is in A

Definition

S_M: set of pairs (p, q) such that $p \not\equiv q$

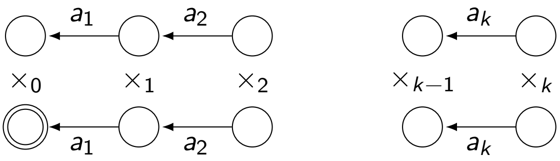
1. If exactly one of p and q is in A, then $(p, q) \in S_M$
2. If for some $\sigma \in \Sigma$, $(\delta(p, \sigma), \delta(q, \sigma)) \in S_M$, then $(p, q) \in S_M$

Minimizing states

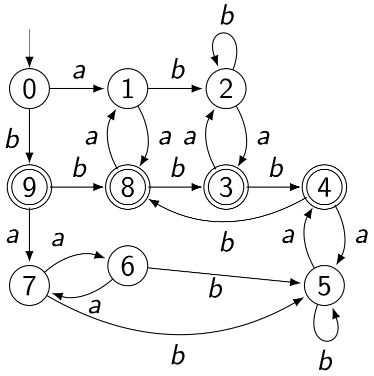
ALGORITHM mark pairs of non-equivalent states

```
start by marking pairs $(p, q)$ where exactly one $p, q$ in $A$

repeat
    for each unmarked pair $(p, q)$
        check whether there is a $\sigma$ such that $(\delta(p, \sigma), \delta(q, \sigma))$ is marked
        then mark $(p, q)$
until this pass does not mark new pairs
```

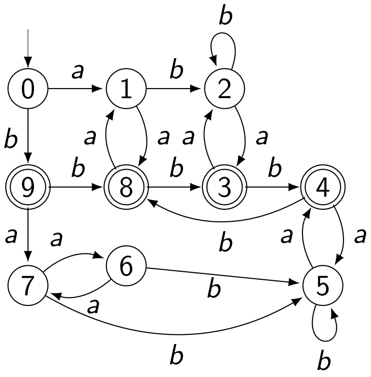


[M] Algo 2.40



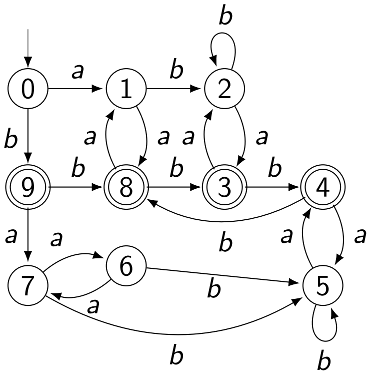
1	·								
2	·	·							
3	·	·	·						
4	·	·	·	·					
5	·	·	·	·	·				
6	·	·	·	·	·	·			
7	·	·	·	·	·	·	·		
8	·	·	·	·	·	·	·	·	
9	·	·	·	·	·	·	·	·	·
	0	1	2	3	4	5	6	7	8

[M] Fig 2.42



	0	1	2	3	4	5	6	7	8
1	·								
2	·	·							
3	1	1	1						
4	1	1	1	·					
5	·	·	·	1	1				
6	·	·	·	1	1	·			
7	·	·	·	1	1	·	·		
8	1	1	1	·	·	1	1	1	
9	1	1	1	·	·	1	1	1	·

[M] Fig 2.42



1	2								
2	2	·							
3	1	1	1						
4	1	1	1	·					
5	2	·	·	1	1				
6	2	2	2	1	1	2			
7	2	2	2	1	1	2	·		
8	1	1	1	·	·	1	1	1	
9	1	1	1	2	·	1	1	1	2
	0	1	2	3	4	5	6	7	8

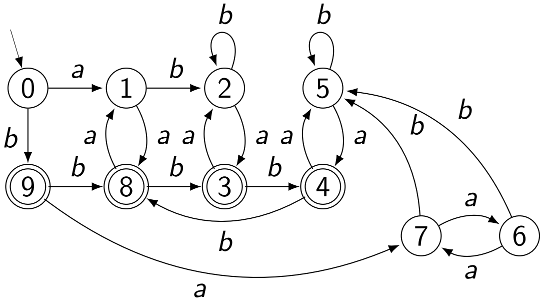
[M] Fig 2.42

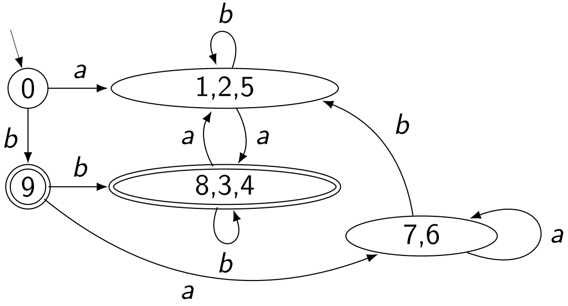
$$\begin{tikzcd}
\arrow[d] & & & \\
0 \arrow[r, "a"] \arrow[d, "b"'] & 1 \arrow[r, "b"] \arrow[d, bend left, "a"] & 2 \arrow[loop above, "b"] \arrow[d, bend left, "a"] & \\
9 \arrow[r, "b"] \arrow[d, "a"'] & 8 \arrow[u, bend left, "a"] \arrow[r, "b"] & 3 \arrow[u, bend left, "a"] \arrow[r, "b"] & 4 \arrow[ll, bend left, "b"] \arrow[d, bend left, "a"] \\
7 \arrow[r, bend left, "a"] \arrow[rrr, bend right, "b"'] & 6 \arrow[l, bend left, "a"] \arrow[rr, "b"] & & 5 \arrow[u, bend left, "a"] \arrow[loop below, "b"]
\end{tikzcd}$$

1	2								
2	2	·							
3	1	1	1						
4	1	1	1	·					
5	2	·	·	1	1				
6	2	2	2	1	1	2			
7	2	2	2	1	1	2	·		
8	1	1	1	·	·	1	1	1	
9	1	1	1	2	3	1	1	1	2
	0	1	2	3	4	5	6	7	8

Resulting (minimal) FA. . .

[M] Fig 2.42





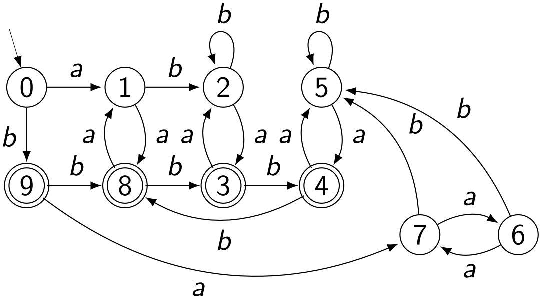
[M] Fig 2.42

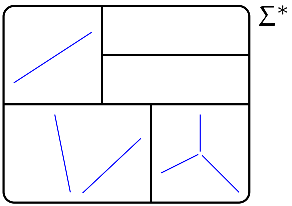
$L = L(M)$

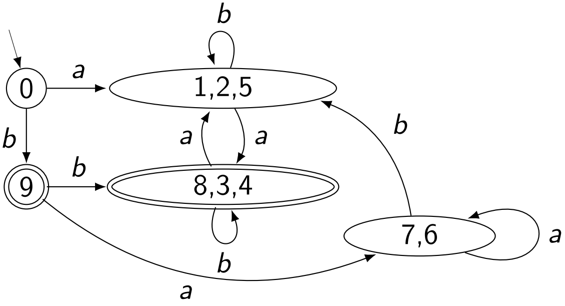
$\equiv_M$ state $\delta^*(q_0, x)$

$\equiv_L$ "future" L/x

$x \equiv_M y$, then $x \equiv_L y$.



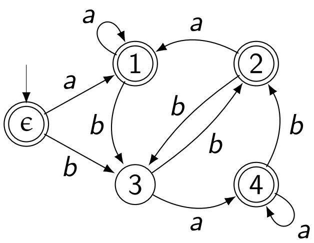


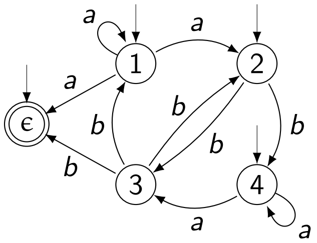


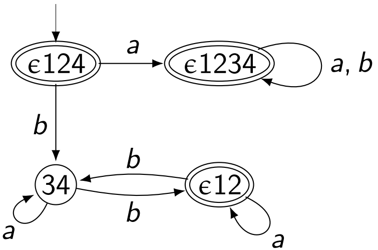
[M] Fig 2.42

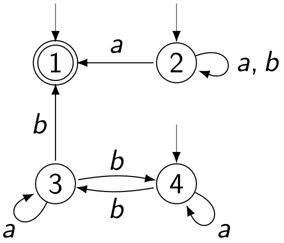
☒Example: Brzozowski minimization

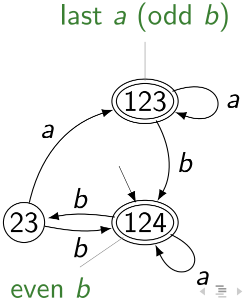
Self-study











ABOVE

Brzozowski observes that one can minimize an FA by performing the following operations twice: invert (mirror), then determinize, i.e., make deterministic.

It is rather magical that this indeed works.

The method is in theory rather unfavourable, because of the exponentiation when determinizing, but in practice seems not too slow.

Previous challenge

Can you find a language that satisfies the generalized version of the pumping lemma but is not accepted by a finite automaton?

According to this wiki page, the general version of the pumping lemma still does not characterize regular languages.

Next challenge?

Homework 1 is available!