Automata Theory

Dalia Papuc, Rudy van Vliet

Bachelor Informatica
Data Science and Artificial Intelligence
Universiteit Leiden

Fall 2023

Universiteit
Leiden
Leiden Institute of
Advanced Computer Science

Practical Information

- lectures: Mondays, 17.15–19.00, C1 Lecture Hall

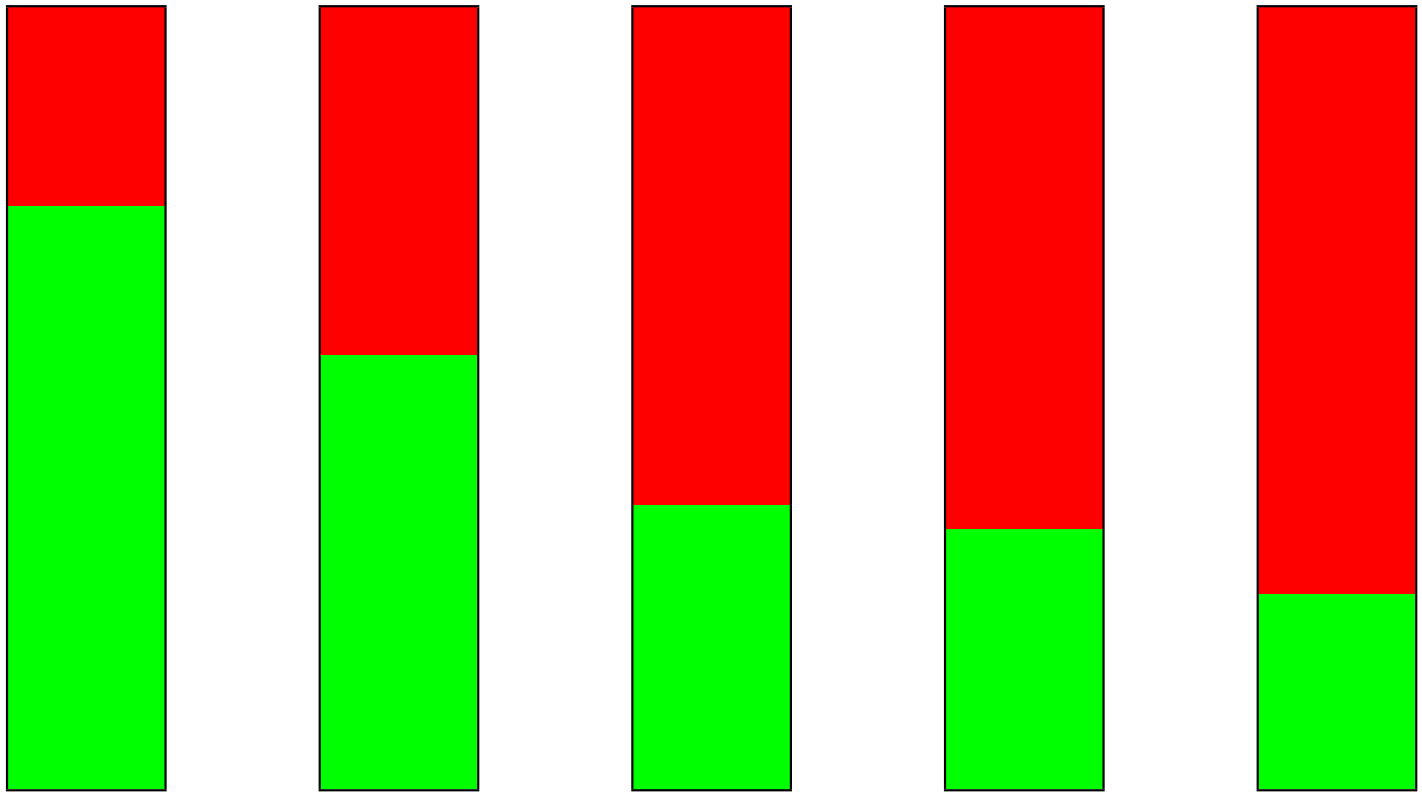
excercise classes (by assistants): Wednesdays, 09.00–10.45, varying rooms, varying buildings (check MyTimeTable)

from 5 September – 13 December 2023

- based on book: John C. Martin, Introduction to Languages and the Theory of Computation, 4th edition (available?)
- Chapters 1–6 (partly)
- recordings lecture, released after two weeks
- email to assistants: automata@liacs.leidenuniv.nl

- exams: Thursday, 21 December 2023, 09:00-12:00
Tuesday, 26 March 2024, 09:00-12:00
- Four homework assignments (individual) (assistants)
Not mandatory, but ...
finalgrade = 70% * gradeexam + 30% * gradehomeworkassignments
if gradeexam $\geq$ 5.5, then finalgrade $\geq$ 5.5
if gradeexam $<$ 5.5, then finalgrade = gradeexam

first exam, 19 December, 2022



4 HW	3 HW	2 HW	1 HW	0 HW
74.67%	55.56%	36.36%	33.33%	25.00%

Website

http://www.liacs.leidenuniv.nl/~vlietrvan1/automata/

- slides (thanks to HJH)
- overview of treated material
- solutions to selected exercises
- homework assignments
- errata

Brightspace

https://brightspace.universiteitleiden.nl

- submission of homework assignments
- grades
- email
- groups for exercise classes

Questions?

Something completely different

Voorbereiding Programmeerwedstrijden

Tuesdays, 13.15-17.00, six weeks

RANK	TEAM	SCORE	A	B	C	D	E	F	G	H	I	J	K	L
1	Delft University of Technology **Segfault go BRRRR** Delft University of Technology	12 1726	277 3 tries	109 1 try	61 1 try	149 2 tries	10 1 try	114 2 tries	193 1 try	77 2 tries	85 1 try	228 2 tries	169 2 tries	94 2 tries
2	Vrije Universiteit **std::string name = getTeamName();** Vrije Universiteit	10 1980		67 1 try	188 1 try	211 1 try	10 2 tries	31 1 try		279 5 tries	288 4 tries	111 2 tries	278 3 tries	297 1 try
3	Utrecht University **IT giet oan** Utrecht University	9 1295	2 tries	14 1 try	140 1 try	165 1 try	23 2 tries	56 1 try		218 3 tries	253 4 tries	185 2 tries	101 1 try	2 tries
4	**ADA Refactor** Delft University of Technology	9 1466	229 2 tries	83 1 try		272 4 tries	24 1 try	37 1 try		137 1 try	66 1 try	227 3 tries	251 2 tries	
5	**Chindia Targoviste** Delft University of Technology	9 1541	179 1 try	161 2 tries	70 1 try	215 2 tries	10 1 try	67 1 try			12 tries	274 4 tries	251 2 tries	194 1 try
6	**WA & Chill** Delft University of Technology	9 1780		75 4 tries	286 1 try	192 1 try	11 1 try	95 3 tries		205 1 try	164 4 tries	273 5 tries	239 1 try	4 tries
7	**Exponential Fenwick** Delft University of Technology	8 1212		141 1 try	291 2 tries	250 2 tries	21 3 tries	50 1 try		4 tries	40 2 tries	7 tries	99 1 try	220 1 try
8	Radboud University **Reckless Raccoons** Radboud University	8 1295		66 1 try	246 2 tries	169 2 tries	20 2 tries	91 1 try			140 2 tries	7 tries	201 1 try	222 4 tries
9	**PPAP** Vrije Universiteit	8 1648		44 1 try	159 1 try	276 1 try	25 4 tries	285 6 tries			77 3 tries	260 5 tries	222 2 tries	
10	Eindhoven University of Technology **Duck-Team\%** Eindhoven University of Technology	8 1839		179 1 try		297 1 try	71 4 tries	117 4 tries			152 5 tries	290 2 tries	258 1 try	255 1 try
11	**πece of** Eindhoven University of Technology	7 975		132 1 try	266 1 try	230 1 try	11 1 try	67 1 try			28 1 try	4 tries	241 1 try	
12	Rijksuniversiteit Groningen **Balloon Addicts, Passionate Co** Rijksuniversiteit Groningen	7 1003	4 tries	100 1 try	156 1 try	283 1 try	6 1 try	37 3 tries			118 1 try			243 2 tries
13	**Bidoof Aron Piplup Cyndaquil** Radboud University	6 532		34 1 try	205 1 try		28 1 try	61 1 try		3 tries	83 1 try	2 tries	121 1 try	
14	**Reduce all the things** Radboud University	6 610	2 tries	48 1 try		97 1 try	23 2 tries	78 2 tries	1 try	215 2 tries	89 1 try	2 tries	1 try	1 try
15	**Drive(2011)** Vrije Universiteit	6 911		49 1 try	5 tries	205 1 try	24 1 try	100 1 try			5 tries	163 4 tries	3 tries	290 2 tries
16	Universiteit van Amsterdam	6 1141		253	267		19	123			27	232		

Context

- Foundations of Computer Science / Fundamentele Informatica 1
- Computability / Fundamentele Informatica 3
- (Compiler Construction)

Contents

edit 2023-09-04

Section 1

Languages

1. Languages
   - Origins
   - Letter, alphabet, string, language
   - Chomsky hierarchy

Possibilities / limitations of computer / algorithms
Model
Computer receives input, performs 'computation', gives output

- Given instance of Nim. Who wins?
- Given sequence of numbers. Sort
- Given edge-weighted graph.
  Give shortest route from A to B

Dealing with languages / sets of instances

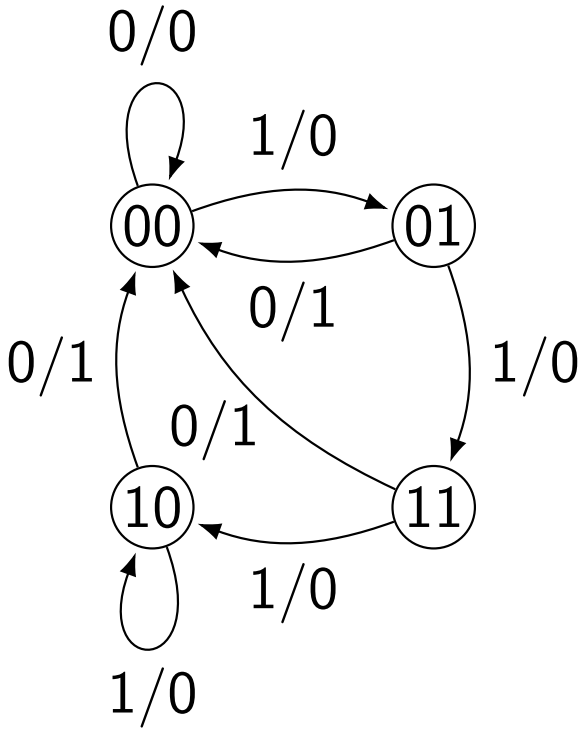
1. Abstract machines to accept or to recognize languages
2. Grammars to generate languages
3. Expressions to describe languages

Formal Languages: Origins

1. Logic and recursive-function theory Introduction to Logic
2. Switching circuit theory and logical design FDSD
3. Modeling of biological systems, particularly developmental systems and brain activity
4. Mathematical and computational linguistics
5. Computer programming and the design of ALGOL and other problem-oriented languages

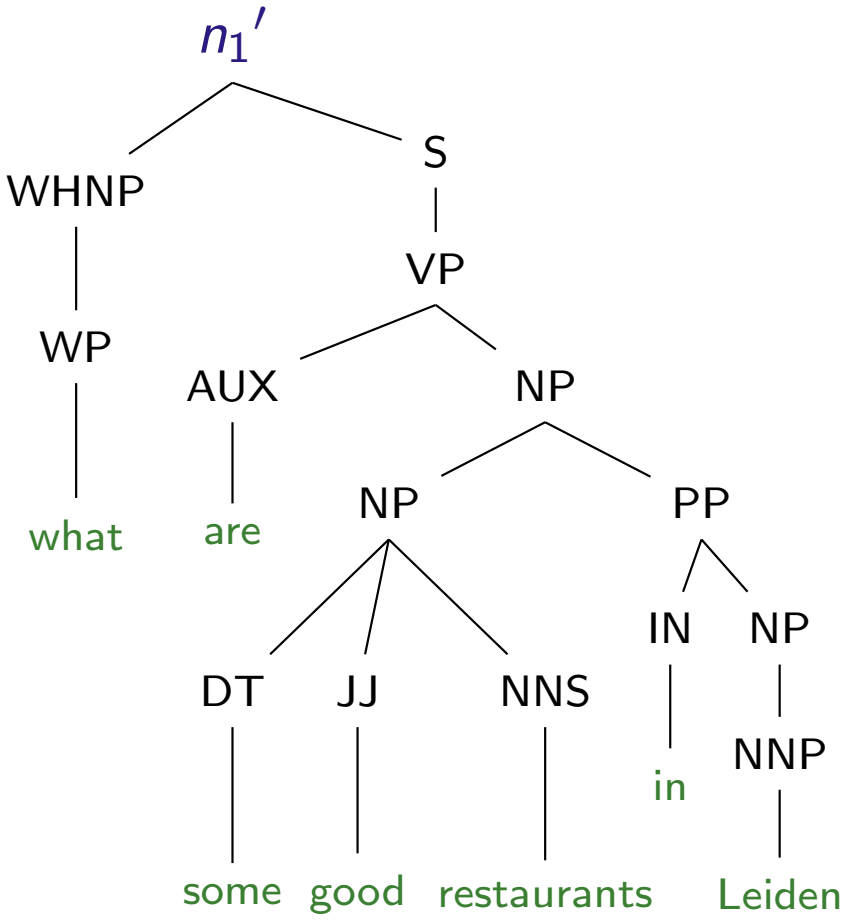
S.A. Greibach. Formal Languages: Origins and Directions.
Annals of the History of Computing (1981) doi:10.1109/MAHC.1981.10006

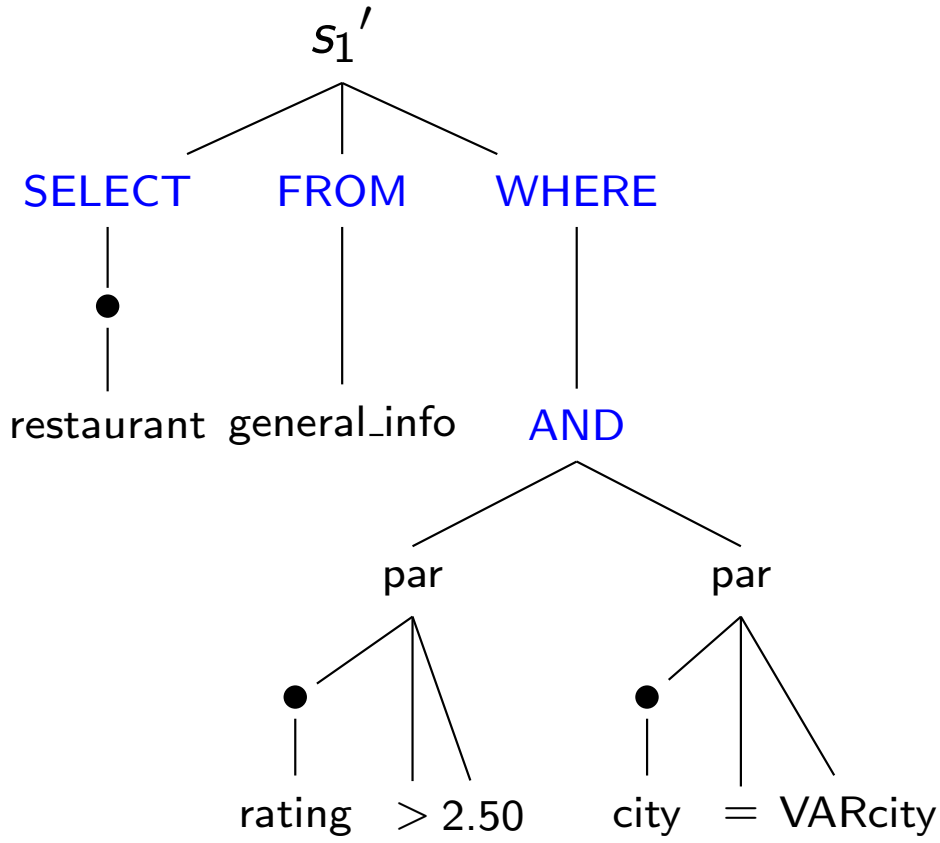
Mealy Machine



Fundamentals of Digital Systems Design by Todor Stefanov, Leiden University

Specifying languages





A. Giordani and A. Moschitti. Corpora for Automatically Learning to Map Natural Language Questions into SQL Queries (LREC 2010)

Inductive definition ⇒ grammar

inductive definition (of set of strings over { (,) })

Example

– $\Lambda \in$ *Balanced*	*basis*
– for every $x, y \in$ *Balanced*, also $xy \in$ *Balanced*	*induction:1*
– for every $x \in$ *Balanced*, also $(x) \in$ *Balanced*	*:2*
– no other strings in *Balanced*	*closure*

strings

basis Λ ind:2 $(\Lambda) = ()$ ind:1 $()()$ ind:2 $(())$

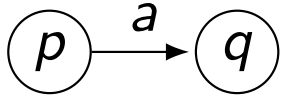
ind:1 $()()()$, $()(())$, $(())()$, ind:2 $(()())$, $((()))$

grammar

rules: $S \to \Lambda \mid SS \mid (S)$

rewriting: $S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow ()(S) \Rightarrow ()((S)) \Rightarrow ()(())$

[M] E 1.19 see Dyck language, Catalan numbers

TYPE	**grammar**	**automaton**
3	regular	
	regular	finite automaton
	$A \to aB$	
2	context-free	
	$A \to \alpha$	pushdown (+lifo stack)
1	context-sensitive	
	$(\beta_\ell, A, \beta_r) \to \alpha$	linear bounded
	$\alpha \to \beta \qquad \lvert\beta\rvert \geq \lvert\alpha\rvert$	
	monotone	
0	recursively enumerable	
	$\alpha \to \beta$	turing machine

[M] Table 8.21

letter, symbol σ $0, 1$ a, b, c

alphabet Σ $\{a, b, c\}$

(finite, nonempty)

string, word w finite

$w = a_1 a_2 \dots a_n$, $a_i \in \Sigma$ *abbabb*

empty string λ, Λ, ε

length $|x|$ $|\Lambda| = 0$ $|xy| = |x| + |y|$

concatenation $a_1 \dots a_m \cdot b_1 \dots b_n$ $ab \cdot babb$

$w\Lambda = \Lambda w = w$ $(xy)z = x(yz)$

string $w \in \Sigma^*$ $w \in \{a, b\}^*$

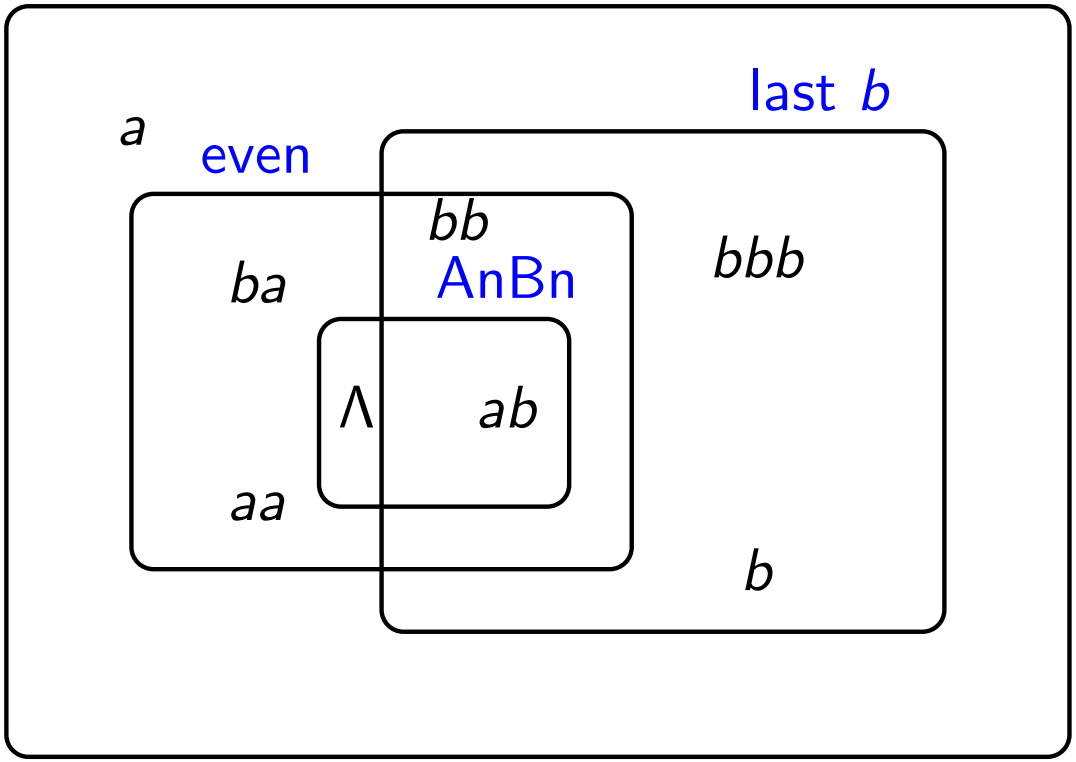
$\Sigma^* = \{\Lambda, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$ *canonical order*

infinite set of finite strings

language $L \subseteq \Sigma^*$

Example

- $\{a, b\}^*$ all strings over $\{a, b\}$ $\Lambda, baa, aaaaa$
- all strings of even length $\Lambda, babbba$
- all strings with last letter b $bbb, aabb$
- $AnBn = \{a^n b^n \mid n \in \mathbb{N}\}$ $\Lambda, aaabbb$ $(\mathbb{N} = \{0, 1, 2, 3, \ldots\})$



$\{a, b\}^*$

Confusion

Λ vs. $\{\Lambda\}$ vs. $\varnothing$

Boolean algebra $\cup, \cap, ^c$

commutativity	$A \cup B = B \cup A$	$\ldots$
associativity	$(A \cup B) \cup C = A \cup (B \cup C)$	
distributivity	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	
idempotency	$A \cup A = A$	$A \cap A = A$
De Morgan	$(A \cup B)^c = A^c \cap B^c$	
unit	$A \cup \varnothing = A$	$A \cap U = A$
	$A \cap \varnothing = \varnothing$	$A \cup U = U$
involution	$(A^c)^c = A$	
complement	$A \cap A^c = \varnothing$	
		duality

brackets

priority ^{c} before $\cup, \cap$

$K \cap L \cup M$??

[M] page 4 FDSD, FOCS

Definition

$K \cdot L = KL = \{\, xy \mid x \in K, y \in L \,\}$

$\{a, ab\}\{a, ba\} = \{aa, aba, abba\}$

one	$\{\Lambda\}L = L\{\Lambda\} = L$
zero	$\varnothing L = L\varnothing = \varnothing$
associative	$(KL)M = K(LM)$

$L^0 = \{\Lambda\}$, even if $L = \varnothing$, c.f. 0^0
$L^1 = L, \quad L^2 = LL, \ldots$
$L^{n+1} = L^n L.$

Definition

$L^* = \bigcup_{n \geq 0} L^n$

$$L^n = \underbrace{L \cdot L \cdot \ldots \cdot L}_{n \text{ times}}$$

$$L^n = \{\ w_1 w_2 \ldots w_n \mid w_1, w_2, \ldots, w_n \in L\ \} \quad \text{fixed } n$$

$$L^* = \{\ w_1 w_2 \ldots w_n \mid w_1, w_2, \ldots, w_n \in L, n \in \mathbb{N}\ \}$$

c.f. Σ^*

Example

$\{a\}^* \cdot \{b\} = \{\Lambda, a, aa, aaa, \ldots\} \cdot \{b\} = \{b, ab, aab, aaab, \ldots\}$

$(\ \{a\}^* \cdot \{b\}\ )^* = \{b, ab, aab, aaab, \ldots\}^* =$
$\{\Lambda, b, ab, bb, aab, abb, bab, bbb, aaab, \ldots\}$

$(\ \{a\}^* \cdot \{b\}\ )^* = \{a, b\}^* \{b\} \cup \{\Lambda\}$

family all languages that can be defined by
– type of automata
(deterministic) finite aut. FA, NFA, pushdown aut. PDA
– type of grammar
context-free grammar CFG, regular (aka right-linear)
– certain operations
regular REG

Boolean operations: $\cup$, $\cap$, ^{c}
Regular operations: $\cup$, $\cdot$, *

family F *closed under* operation ∇:
if $K, L \in F$, then $K \nabla L \in F$.

RECOGNIZING, algorithm

$L_2 = \{ x \in \{a, b\}^* \mid n_a(x) > n_b(x) \}$

count a and b

deterministic [finite] automaton

GENERATING, description

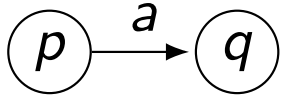
regular expression

$L_1 = (\{ab, bab\}^*\{b\})^*\{ab\} \cup \{b\}\{ba\}^*\{ab\}^*$

recursive definition

$\hookrightarrow$well-formed formulas

grammar

TYPE	**grammar**	**automaton**
3	regular	
	regular	finite automaton
	$A \to aB$	
2	context-free	
	$A \to \alpha$	pushdown (+lifo stack)
1	context-sensitive	
	$(\beta_\ell, A, \beta_r) \to \alpha$	linear bounded
	$\alpha \to \beta$ $\quad$ $\|\beta\| \geq \|\alpha\|$	
	monotone	
0	recursively enumerable	
	$\alpha \to \beta$	turing machine

[M] Table 8.21

Proofs

– clever idea, intuition

– formal construction, specification

– show it works, e.g., induction

once the idea is understood,
the other parts might be boring

but essential to test intuition

examples help to get the message

Quiz[1]

L_1, L_2, L_3 are languages over some alphabet Σ.

For each pair of languages below, what is their relationship?

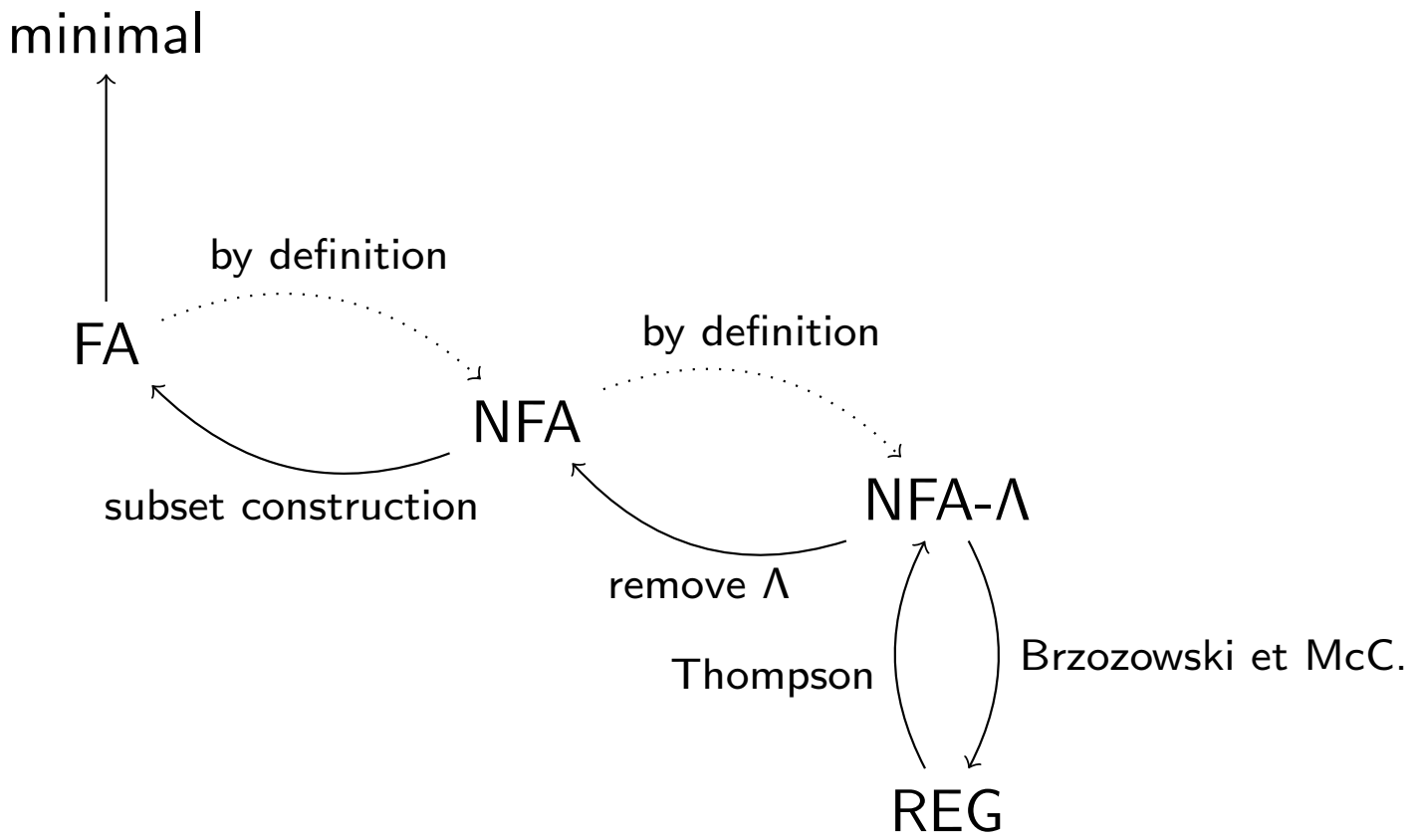
Are they always equal? If not, is one always a subset of the other?

1. $L_1(L_2 \cap L_3)$ vs. $L_1L_2 \cap L_1L_3$
2. $L_1^* \cap L_2^*$ vs. $(L_1 \cap L_2)^*$
3. $L_1^*L_2^*$ vs. $(L_1L_2)^*$

[M] Exercise 1.37

[1] A quiz is a brief assessment used in education to measure growth in knowledge, abilities, and/or skills. Wikipedia

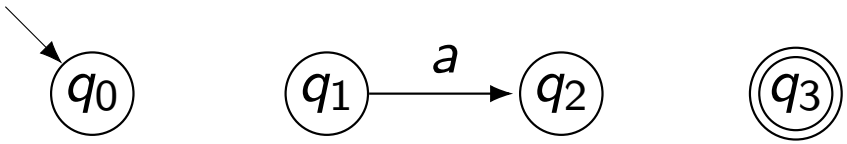
Overview Ch.2 & 3



Section 2

(Deterministic) Finite Automata

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

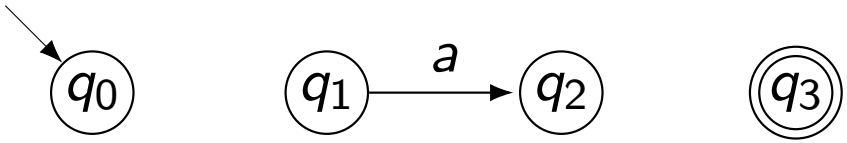


Example

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

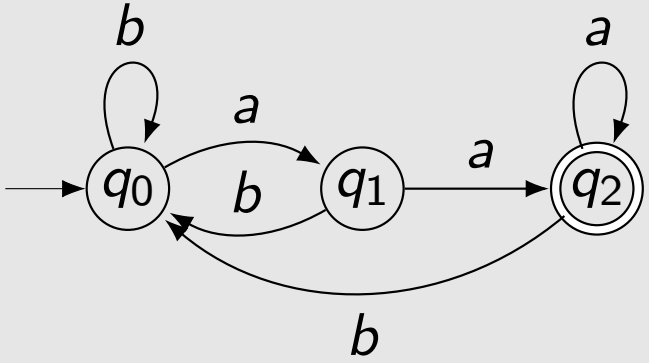
...

[M] E. 2.1



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

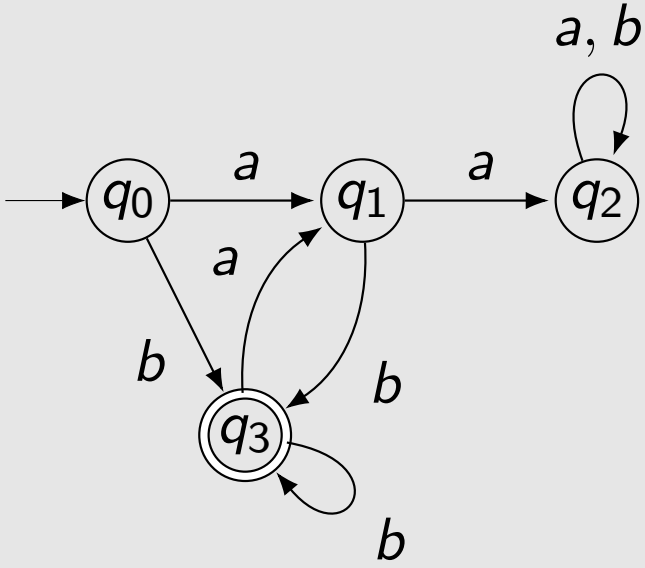
$L_2 = \{ x \in \{a, b\}^* \mid x \text{ ends with } b \text{ and does not contain } aa \}$

...

[M] E. 2.3

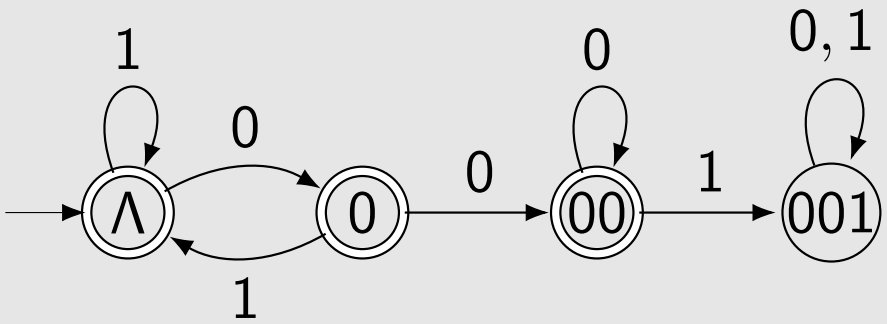
Example

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



[M] E. 2.3

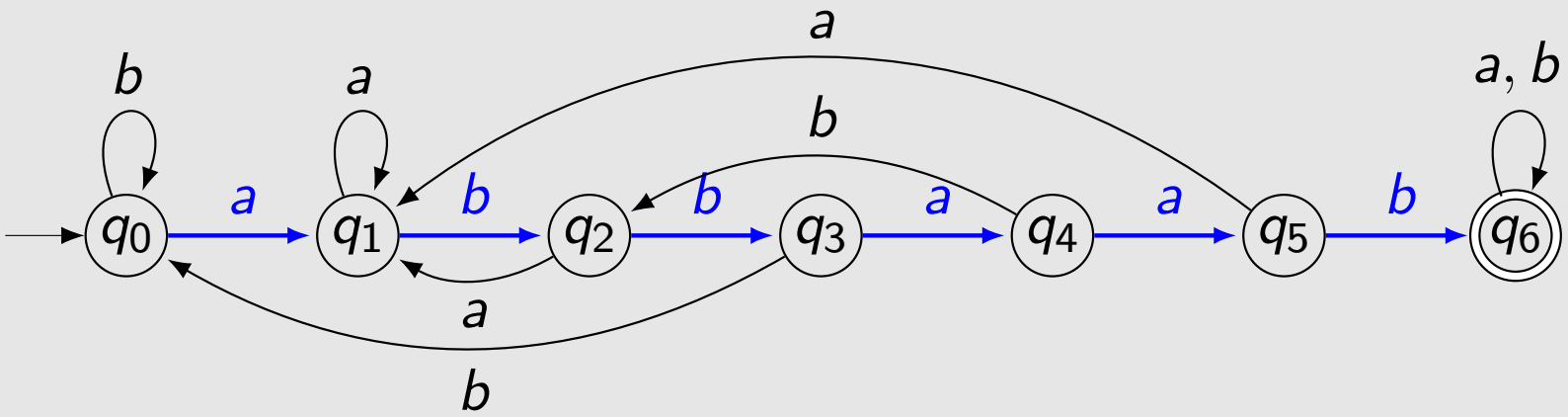
Example (Strings not containing 001)



[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

Binary integers divisible by 3

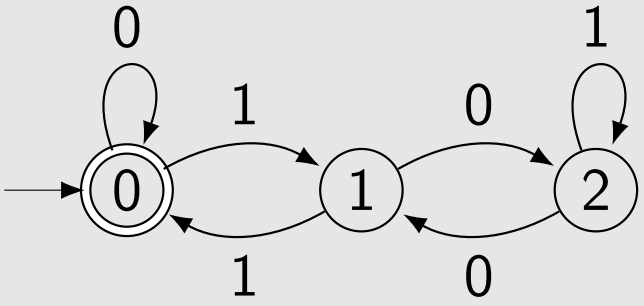
$w \in \{0,1\}^* \longrightarrow val(w) \in \mathbb{N}$

$val(w0) = \ldots$

$val(w1) = \ldots$

Binary integers divisible by 3

Example



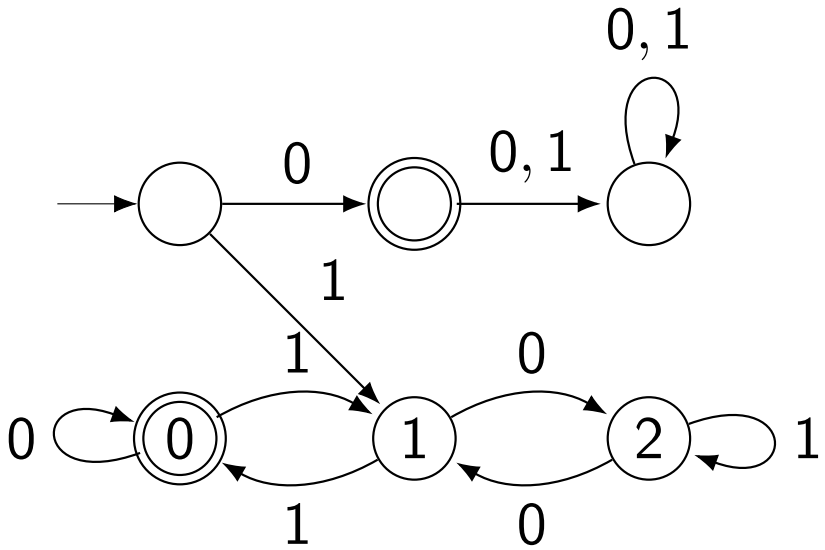
δ	0	1
x	$2x$	$2x+1$
0	0	1
1	2	0
2	1	2

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

$val(w0) = 2 \cdot val(w)$

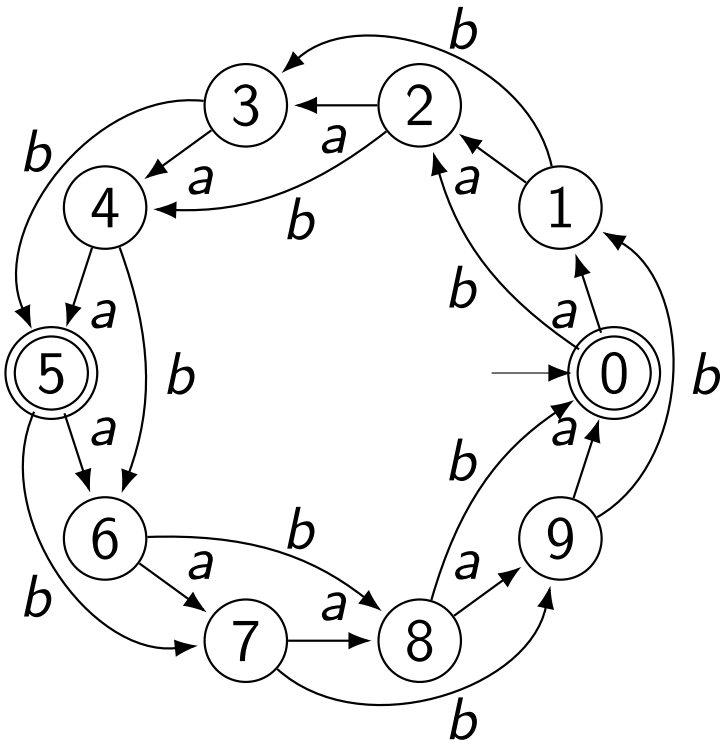
$val(w1) = 2 \cdot val(w) + 1$

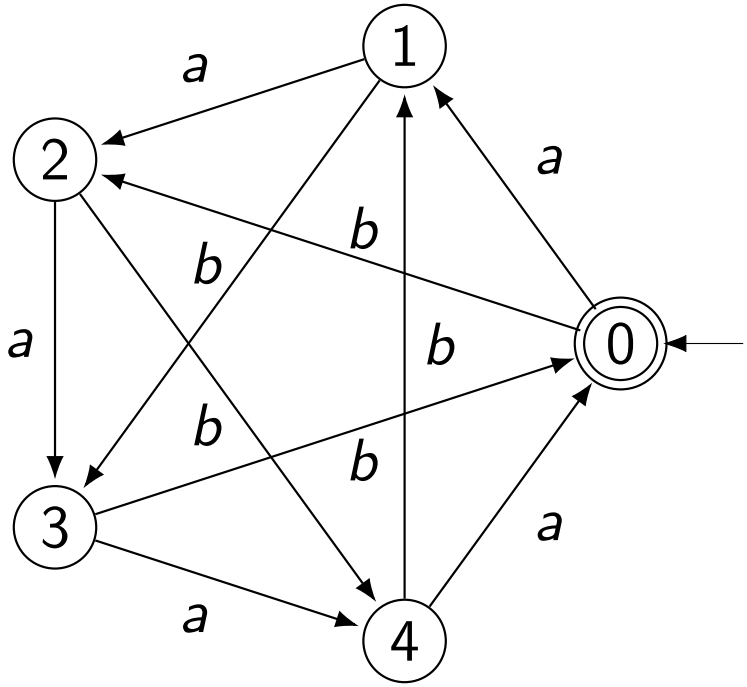
states represent $val(w)$ modulo 3



[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.