Formele Talen en Berekenbaarheid

Rudy van Vliet

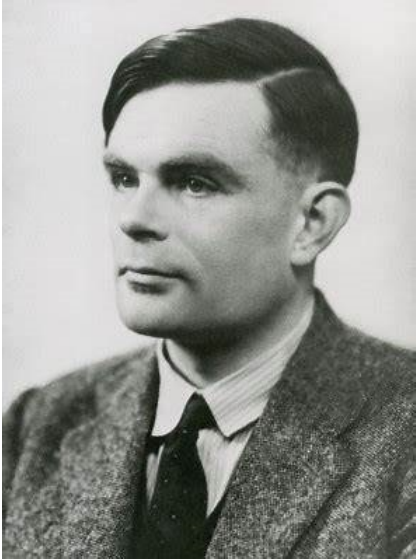
Bachelor Informatica
Universiteit Leiden

najaar 2025

Universiteit Leiden
Leiden Institute of
Advanced Computer Science

- Automata Theory (6 EC), Computability (3 EC)
- FTB this year (6 EC)
- Automaten (3 EC)
- FTB next year (6 EC)
- DSAI: Languages and Computation
- students from earlier years...

Why this course



Website and Brightspace

Practical information on website:
https://liacs.leidenuniv.nl/~vlietrvan1/ftb/

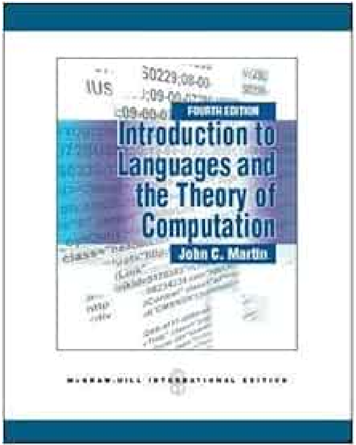
Grades and email via Brightspace

Lectures: Monday 11:00–12:45 in BW.0.40 (except today, obviously)
Lecturer: Rudy van Vliet

Tutorials: Thursday 11:00–12:45 in various classrooms (MyTimeTable)

e-mail: ...(at)liacs(dot)leidenuniv(dot)nl

Introduction to Languages and the Theory of Computation, John C. Martin, 4th edition



- Four homework assignments (30%), average grade H
- Written exam (70%), grade W

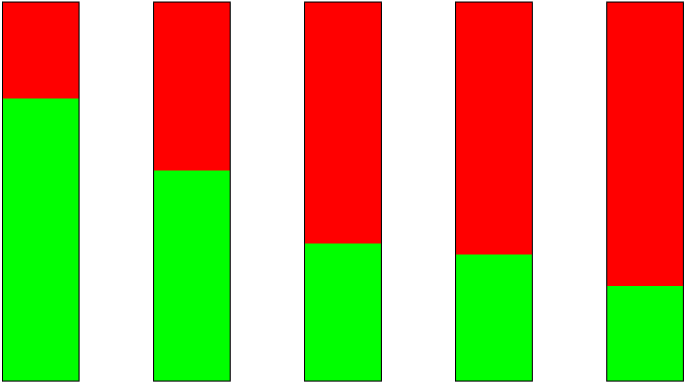
Let $F = 0.3H + 0.7W$.

if $W < 5.5$ then grade $= W$
else if $F \geq 5.5$ then grade $= F$
else grade $= 6.0$

Homework

- not mandatory, but ...
- individual

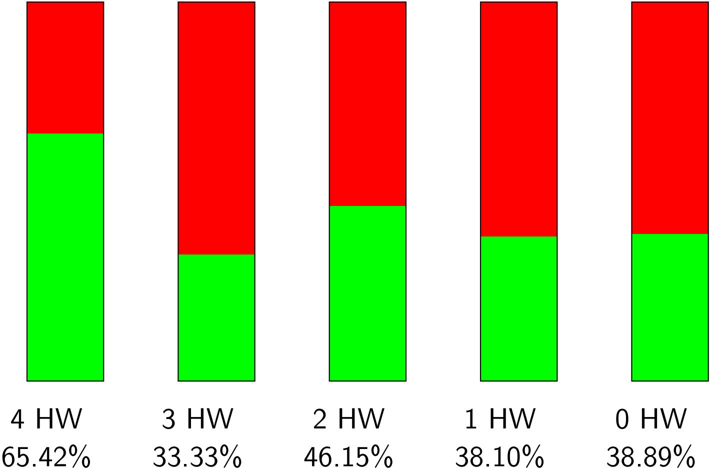
first exam, 19 December, 2022



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

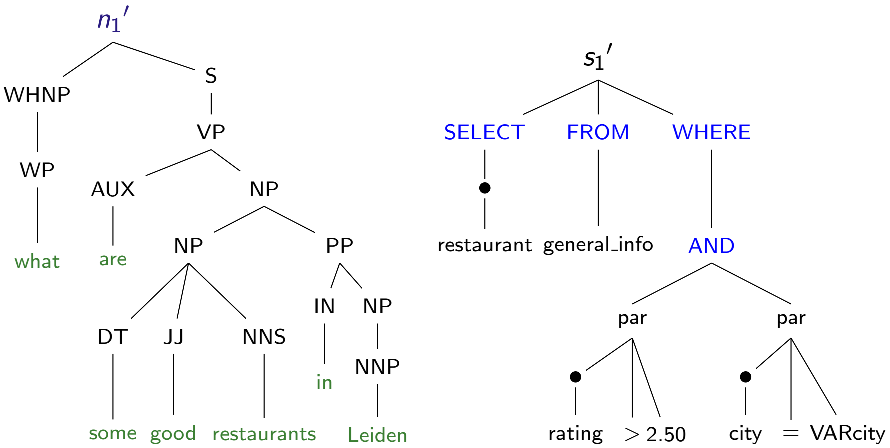
Results from the Past (Automata Theory)

first exam, 21 December, 2023



Dealing with languages / sets of strings

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

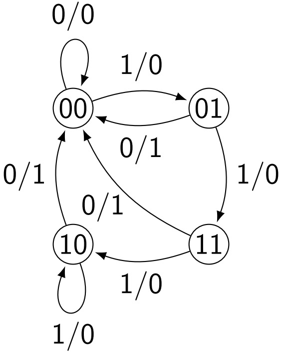


A. Giordani and A. Moschitti. Corpora for Automatically Learning to Map Natural Language Questions into SQL Queries (LREC 2010)

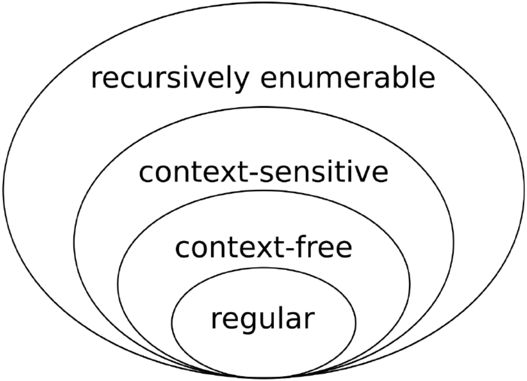
1. Logic and recursive-function theory   Introduction to Logic
2. Switching circuit theory and logical design   FDSD
3. Modeling of biological systems, particularly developmental systems and brain activity
4. Mathematical and computational linguistics
5. Computer programming and the design of ALGOL and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



Turing machines

Pushdown automata

Finite automata

[M] Table 8.21

commutativity	$A \cup B = B \cup A$	$\dots$
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

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

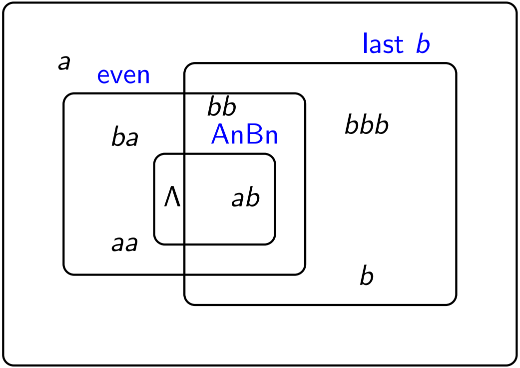
language $L \subseteq \Sigma^*$

Confusion

Λ vs. $\{\Lambda\}$ vs. $\varnothing$

Example

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



$\{a, b\}^*$

Definition

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

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

$$\begin{array}{rl} \text{one} & \{\Lambda\}L = L\{\Lambda\} = L \\ \text{zero} & \varnothing L = L\varnothing = \varnothing \\ \text{associative} & (KL)M = K(LM) \end{array}$$

$L^0 = \{\Lambda\}$, 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

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

count a and b

deterministic [finite] automaton

GENERATING, description

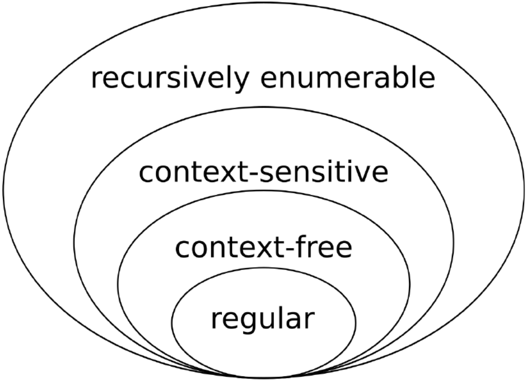
regular expression

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

recursive definition

$\hookrightarrow$well-formed formulas

grammar



Turing machines

Pushdown automata

Finite automata

[M] Table 8.21

Proofs

During lectures: sometimes complete proofs, sometimes only sketch

In exercises, validity of a statement requires *proof*

To prove non-validity, provide *counterexample*

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

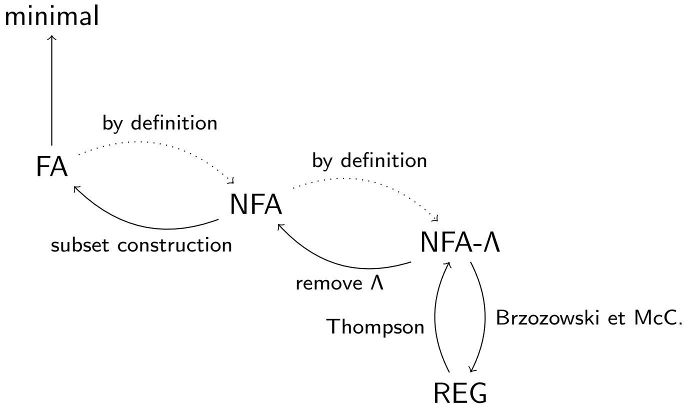
For each pair of languages below, what is their relationship?

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

1. $L_1(L_2 \cap L_3) \quad \text{vs.} \quad L_1 L_2 \cap L_1 L_3$
2. $L_1^* \cap L_2^* \quad \text{vs.} \quad (L_1 \cap L_2)^*$
3. $L_1^* L_2^* \quad \text{vs.} \quad (L_1 L_2)^*$

[M] Exercise 1.37

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


minimal
by definition
FA
by definition
NFA
subset construction
NFA-Λ
remove Λ
Thompson
Brzozowski et McC.
REG

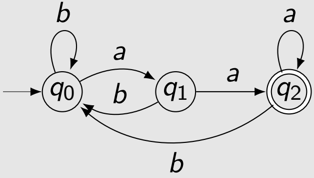
Section 2

(Deterministic) Finite Automata

1. (Deterministic) Finite Automata
   - Examples
   - FA definition
   - Boolean operations
   - Pumping lemma for regular languages
   - Decision problems
   - Distinguishing strings (not this year)
   - Equivalence classes (not this year)
   - Minimization (not this year)

Example

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

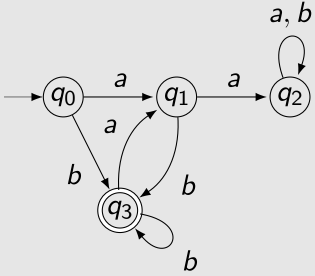


δ	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

[M] E. 2.1

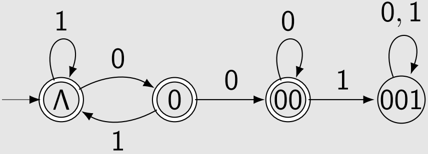
Example

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



[M] E. 2.3

Example (Strings not containing 001)



[L] E 2.4

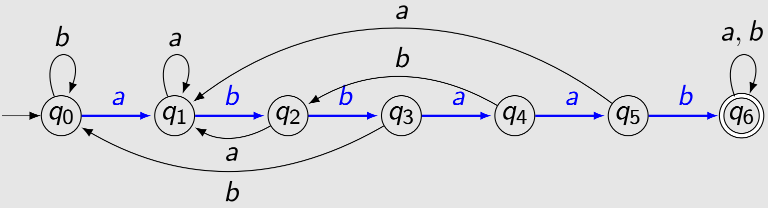
Example (Similar to Knuth-Morris-Pratt string search)

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

[M] E. 2.5

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

Example (Numerical constants in C++)

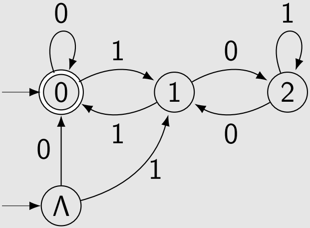
42, 3.1415926, .1415926, 3.
-42, 3.1415926E-42

Note: x is divisible by 3 $\Leftrightarrow$ $x \bmod 3 = 0$.

Note: $x0 = 2 \cdot x$, and $x1 = 2 \cdot x + 1$.

Therefore:
if $x \bmod 3 = 0$, then $x0 \bmod 3 = 0$ and $x1 \bmod 3 = 1$;
if $x \bmod 3 = 1$, then $x0 \bmod 3 = 2$ and $x1 \bmod 3 = 0$;
if $x \bmod 3 = 2$, then $x0 \bmod 3 = 1$ and $x1 \bmod 3 = 2$.

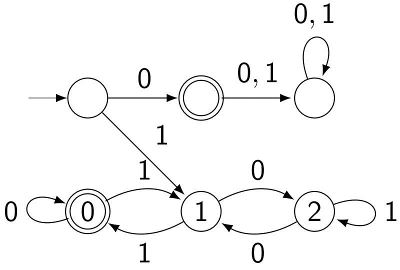
Example



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

States represent x mod 3.

Disallows leading 0's in binary representations, e.g., 0001.



[M] E. 2.7

Definition (FA)

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

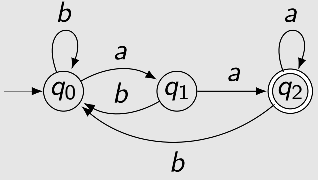
- Q finite set *states*;
- Σ finite *input alphabet*;
- $q_0 \in Q$ *initial* state;
- $A \subseteq Q$ *accepting* states;
- $\delta : Q \times \Sigma \to Q$ *transition function*.

[M] D 2.11 Finite automaton

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

Example

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

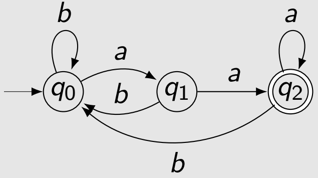


δ	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

[M] E. 2.1

Example

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



$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b\}$
q_0 initial state
$A = \{q_2\}$

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

E.g., $\delta(q_0, a) = q_1$

[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 $q \in Q, y \in \Sigma^*, \sigma \in \Sigma$

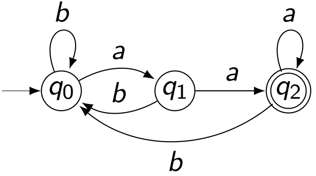
[M] D 2.12 [L] p.40/1

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

Extended transition function



$$\begin{array}{ccccccccc} q_0 & \xrightarrow{a} & q_1 & \xrightarrow{a} & q_2 & \xrightarrow{b} & q_0 & \xrightarrow{b} & q_0 \\ | & & | & & | & & | & & | \\ \delta^*(q_0,\Lambda) & & \delta^*(q_0,a) & & \delta^*(q_0,aa) & & \delta^*(q_0,aab) & & \delta^*(q_0,aabb) \end{array}$$

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

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

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

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

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

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