Automata Theory

Fundamentele Informatica 2

Rudy van Vliet

Bachelor Informatica
Universiteit Leiden

Fall 2022

Universiteit Leiden
Leiden Institute of Advanced Computer Science

Praktische Informatie

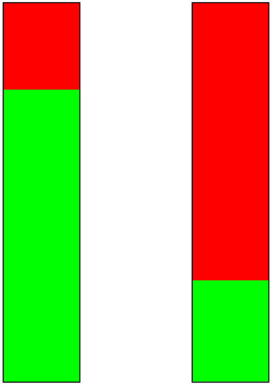
• hoorcollege: dinsdag, 13.15–15.00, Gorlaeus zaal 1
werkcollege: dinsdag, 15.15–17.00, Snellius (meestal) 401, 402, 403, 405, 408
van 6 september – 13 december 2021
• college gebaseerd op boek: John C. Martin, Introduction to Languages and the Theory of Computation, 4th edition (verkrijgbaar?)
• hoofdstuk 1–6 (deels)

Praktische Informatie

• tentamens: maandagochtend 19 december 2022, 09.00-12.00
woensdagochtend 1 februari 2023, 09.00-12.00
• Vier huiswerkopgaven (individueel) (assistenten Perri van den Berg, Amber van den Broek, Dirck van den Ende, Andy Tatman
Niet verplicht, maar . . .
eindcijfer $=$ 70% * tentamencijfer $+$ 30% * cijferhuiswerkopgave
als tentamencijfer $\geqslant 5.5$, dan eindcijfer $\geqslant 5.5$
als tentamencijfer < 5.5, dan eindcijfer $=$ tentamencijfer

Resultaten uit het Verleden

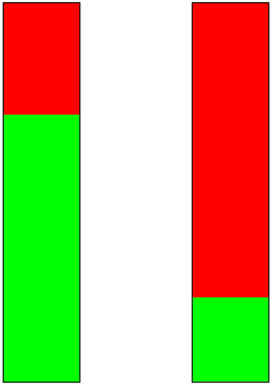
eerste tentamen
23 december 2021



3 of 4 hw-opgaven: 77.14%

0, 1 of 2 hw-opgaven 26.83%

hertentamen
3 februari 2022



3 of 4 hw-opgaven: 70.59%

0, 1 of 2 hw-opgaven 22.22%

Praktische Informatie

Website

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

- slides (dank HJH)
- overzicht van behandelde stof
- antwoorden van bepaalde opgaven
- huiswerkopgaven
- errata

Praktische Informatie

Brightspace

https://brightspace.universiteitleiden.nl

- inleveren huiswerkopgaven
- cijfers
- email
- (zo nodig) Kaltura

Context

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

Contents

1 Languages

2 (Deterministic) Finite Automata

3 Non-Determinism, Regular Expressions, and Kleene's Theorem

4 Context-Free Languages

5 Pushdown Automata

6 Context-Free and Non-Context-Free Languages

7 Course Computability

edit 2022-09-12

Section 1

Languages

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

Computer

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

Languages

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

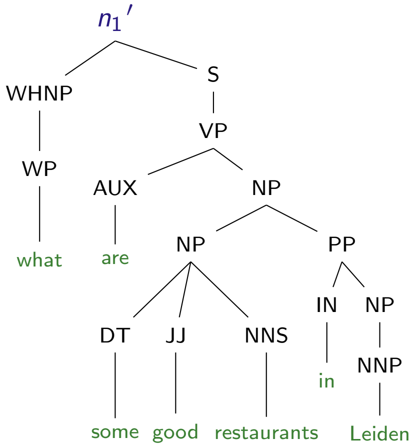
S.A. Greibach. Formal Languages: Origins and Directions.
Annals of the History of Computing (1981) doi:10.1109/MAHC.1981.10006

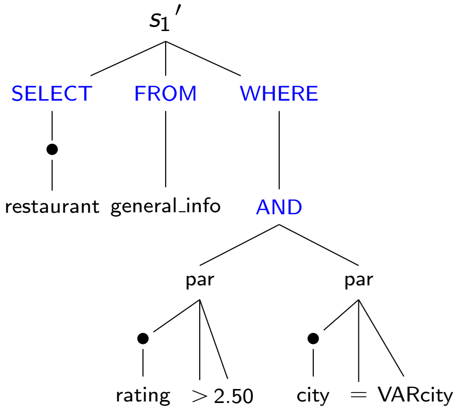
Mealy Machine

$$\begin{tikzcd}
00 \arrow[loop above, "0/0"] \arrow[r, bend left, "1/0"] & 01 \arrow[l, bend left, "0/1"] \arrow[d, bend left, "1/0"] \\
10 \arrow[u, bend left, "0/1"] \arrow[loop below, "1/0"] & 11 \arrow[ul, bend right, "0/1"'] \arrow[l, bend left, "1/0"]
\end{tikzcd}$$

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 $\Rightarrow$ grammar

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

Example

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

strings

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

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

grammar

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

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

[M] E 1.19 see Dyck language, Catalan numbers

Chomsky hierarchy

TYPE		grammar	automaton
3	regular	regular $A \to aB$	finite automaton $p \xrightarrow{a} q$
2	context-free	$A \to \alpha$	pushdown (+lifo stack)
1	context-sensitive	$(\beta_\ell, A, \beta_r) \to \alpha$ $\alpha \to \beta \qquad |\beta| \geqslant |\alpha|$ monotone	linear bounded
0	recursively enumerable	$\alpha \to \beta$	turing machine

[M] Table 8.21

Languages

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 $\lambda,\ \Lambda,\ \varepsilon$

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

concatenation $a_1 \dots a_m \cdot b_1 \dots b_n$ $ab \cdot babb$
$w\Lambda = \Lambda w = w$ $(xy)z = x(yz)$

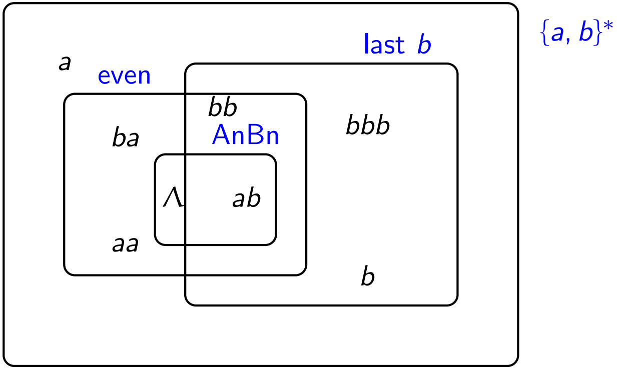
string $w \in \Sigma^*$ $w \in \{a, b\}^*$
$\Sigma^* = \{\Lambda, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$ *canonical order*
infinite set of finite strings

language $L \subseteq \Sigma^*$

Languages are sets

Example

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



Confusion

Λ vs. $\{\Lambda\}$ vs. $\varnothing$

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

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

Concatenation, power, star

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

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

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

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

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

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

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

Families of languages

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 $K \nabla L \in F$.

Specifying languages

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

Chomsky hierarchy

TYPE	**grammar**		**automaton**
3		regular	
	regular		finite automaton
	$A \to aB$		$\begin{tikzcd} p \arrow[r, "a"] & q \end{tikzcd}$
2		context-free	
	$A \to \alpha$		pushdown
			(+lifo stack)
1		context-sensitive	
	$(\beta_\ell, A, \beta_r) \to \alpha$		linear bounded
	$\alpha \to \beta$	$\lvert\beta\rvert \geqslant \lvert\alpha\rvert$	
	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. $L_1(L_2 \cap L_3) \quad \text{vs.} \quad L_1L_2 \cap L_1L_3$
2. $L_1^* \cap L_2^* \quad \text{vs.} \quad (L_1 \cap L_2)^*$
3. $L_1^*L_2^* \quad \text{vs.} \quad (L_1L_2)^*$

[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

$$\begin{tikzcd}
\text{minimal} & & \\
\text{FA} \arrow[u] \arrow[dr, dotted, bend left, "\text{by definition}"] & & \\
& \text{NFA} \arrow[ul, bend left, "\text{subset construction}"] \arrow[dr, dotted, bend left, "\text{by definition}"] & \\
& & \text{NFA-}\Lambda \arrow[ul, bend left, "\text{remove } \Lambda"] \arrow[d, bend left, "\text{Brzozowski et McC.}"] \\
& & \text{REG} \arrow[u, bend left, "\text{Thompson}"]
\end{tikzcd}$$

Section 2

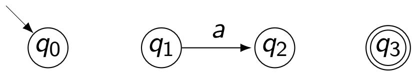
(Deterministic) Finite Automata

Chapter

2 (Deterministic) Finite Automata

- Examples
- FA definition
- Boolean operations
- Decision problems
- Distinguishing strings
- Equivalence classes
- Minimization
- Pumping lemma

Ingredients



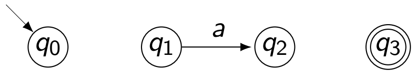
Example

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

. . .

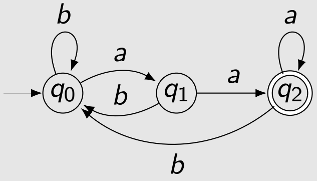
[M] E. 2.1

Ingredients



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

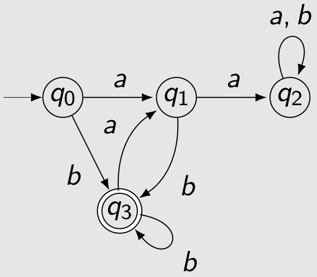
$L_2 = \{\, x \in \{a, b\}^* \mid x \text{ ends with } b \text{ and does not contain } aa \,\}$

...

[M] E. 2.3

Example

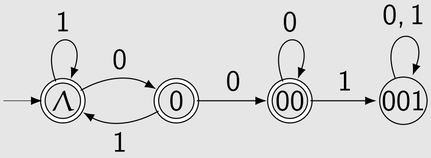
$L_2 = \{ x \in \{a, b\}^* \mid x \text{ ends with } b \text{ and does not contain } aa \}$



[M] E. 2.3

Avoiding pattern

Example (Strings not containing 001)

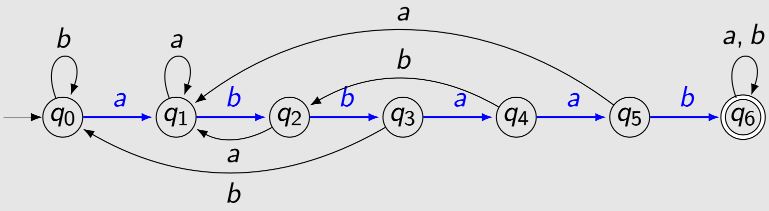

$$\begin{tikzcd}
\arrow[r] & \Lambda \arrow[loop above, "1"] \arrow[r, bend left, "0"] & 0 \arrow[l, bend left, "1"] \arrow[r, "0"] & 00 \arrow[loop above, "0"] \arrow[r, "1"] & 001 \arrow[loop above, "{0,1}"]
\end{tikzcd}$$

[L] E 2.4

Finding pattern

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

$$\begin{tikzcd}
{} \arrow[r] & 0 \arrow[loop above, "0"] \arrow[r, bend left, "1"] & 1 \arrow[l, bend left, "1"] \arrow[r, bend left, "0"] & 2 \arrow[l, bend left, "0"] \arrow[loop above, "1"]
\end{tikzcd}$$

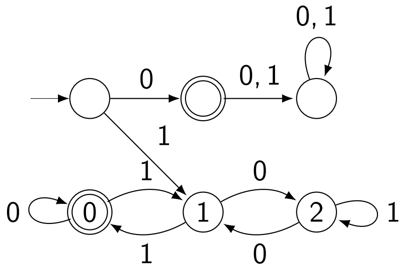
δ	0	1
x	$2x$	$2x+1$
0	0	1
1	2	0
2	1	2

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

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

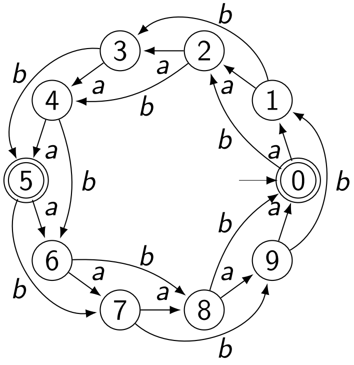
states represent $val(w)$ modulo 3

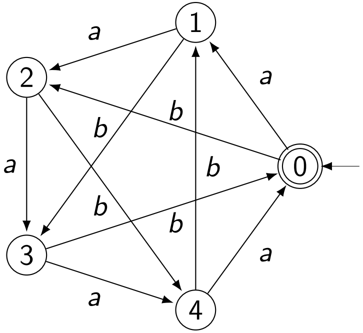
. . . divisible by 3   book-version



[M] E. 2.7

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





☒cs.SE Planar regular languages

Een student vroeg of alle automaten zonder kruisende takken getekend konden worden. De automaat rechts heeft de vorm van K_5 (de volledige graaf op vijf knopen) waarvan bekend is dat die niet planair is.

Dezelfde taal kan echter wel met een vlakke automaat verkregen worden (links). Er zijn talen zonder vlakke automaat.