Automata Theory

Fundamentele Informatica 2

Rudy van Vliet

Bachelor Informatica
Universiteit Leiden

Fall 2020

Universiteit Leiden
Leiden Institute of
Advanced Computer Science

- hoorcollege: dinsdag, 13.00–14.45 (weblecture)
werkcollege: dinsdag, 15.00–15.45 (weblecture)
vraag/antwoord: dinsdag, 16.00–16.45 (Kaltura)
van 1 september – 7 december 2020
- college gebaseerd op boek: John C. Martin, Introduction to Languages and the Theory of Computation, 4th edition (verkrijgbaar?)
- hoofdstuk 1–6 (deels)

- tentamens: maandagochtend 11 januari 2021
dinsdagochtend 23 maart 2021?
- Vier/vijf huiswerkopgaven (individueel) (assistent Femke Slangen)
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

Website

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

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

Brightspace

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

edit 2020-09-10

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

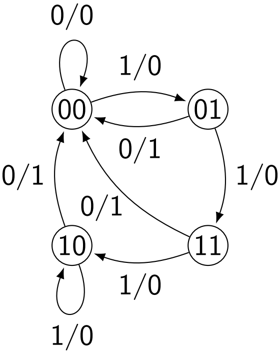
Dealing with languages / sets of instances

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

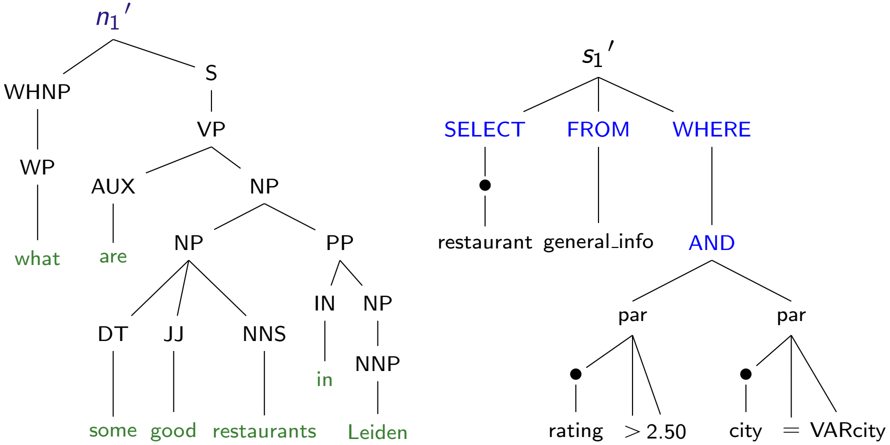
1. Logic and recursive-function theory   Logica
2. Switching circuit theory and logical design   DiTe
3. Modeling of biological systems, particularly developmental systems and brain activity
4. Mathematical and computational linguistics
5. Computer programming and the design of ALGOL and other problem-oriented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

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

Mealy Machine



Digital Technique by Todor Stefanov, Leiden University



A. Giordani and A. Moschitti. Corpora for Automatically Learning to Map Natural Language Questions into SQL Queries (LREC 2010) *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: $S \to \Lambda \mid SS \mid (S)$

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	
	right-linear		finite state
	$A \to aB$		$p \xrightarrow{a} q$
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

letter, symbol σ $0, 1$ a, b, c
alphabet Σ $\{a, b, c\}$
(finite, nonempty)

string, word w finite
$w = a_1 a_2 \ldots a_n$, $a_i \in \Sigma$ $abbabb$
empty string λ, Λ, ε

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

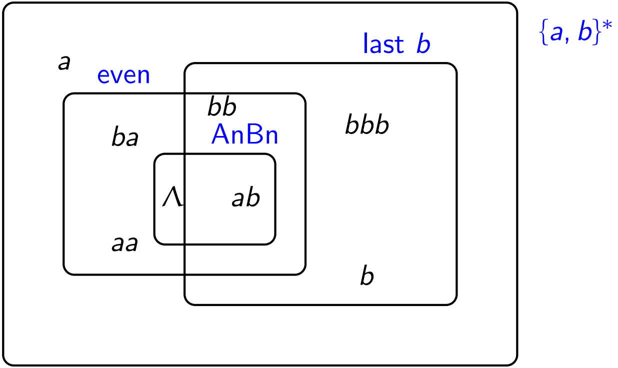
concatenation $a_1 \ldots a_m \cdot b_1 \ldots b_n$ $ab \cdot babb$
$w\Lambda = \Lambda w = w$ $(xy)z = x(yz)$

string $w \in \Sigma^*$
$\Sigma^* = \{\Lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$ *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 $bbb, aabb$
- $AnBn = \{a^n b^n \mid n \in \mathbb{N}\}$ $\Lambda, aaabbb$ $(\mathbb{N} = \{0, 1, 2, 3, \ldots\})$



Λ vs. $\{\Lambda\}$ vs. $\varnothing$

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 $\quad {}^c$ before $\cup, \cap$
$K \cap L \cup M$??

[M] page 4 DiTe, FI1

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	$(KL)M = K(LM)$

$L^0 = \{\Lambda\},\ L^1 = L,\ L^2 = LL,\ \ldots$
$L^{n+1} = L^n L.$

Definition

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

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

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

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

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

[M] Table 8.21

– 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

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

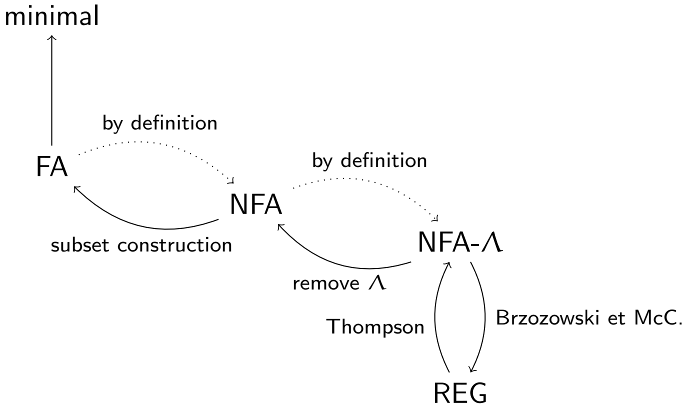
For each pair of languages below, what is their relationship?

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. $(L_1L_2)^*$

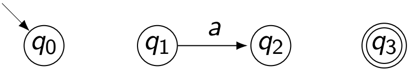
[M] Exercise 1.37

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


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

Section 2

(Deterministic) Finite Automata

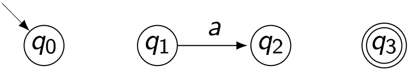


Example

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

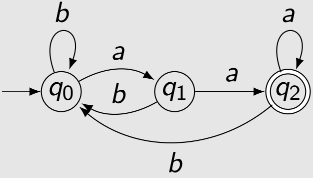
...

[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

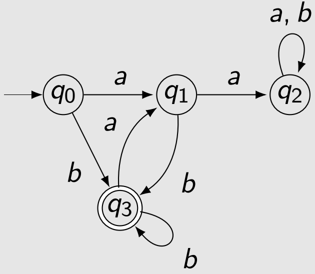
$L_2 = \{\, x \in \{a, b\}^* \mid x \text{ ends with } b \text{ and does not contain } aa \,\}$

...

[M] E. 2.3

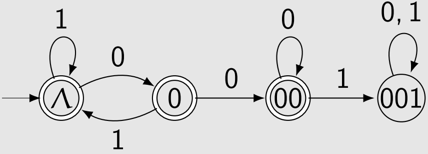
Example

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



[M] E. 2.3

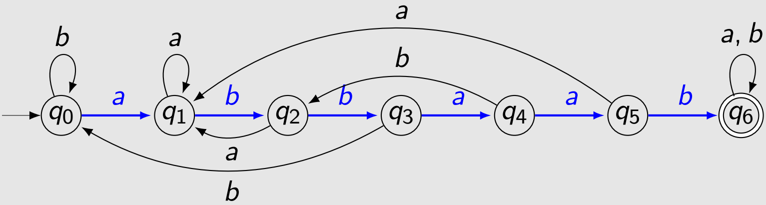
Example (Strings not containing 001)



[L] E 2.4

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

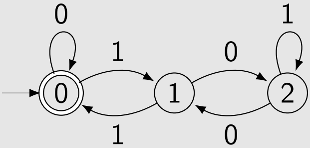
$L_3 = \{\ x \in \{a, b\}^* \mid x \text{ contains the substring } abbaab\ \}$



[M] E. 2.5

$w \in \{0,1\}^* \longrightarrow val(w) \in \mathbb{N}$
$val(w0) = \ldots$
$val(w1) = \ldots$

Example



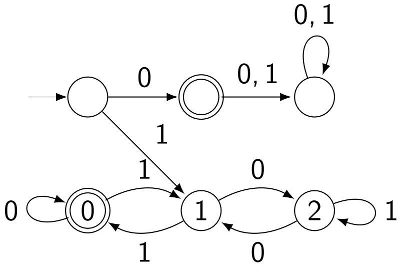
δ	0	1
x	$2x$	$2x+1$
0	0	1
1	2	0
2	1	2

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

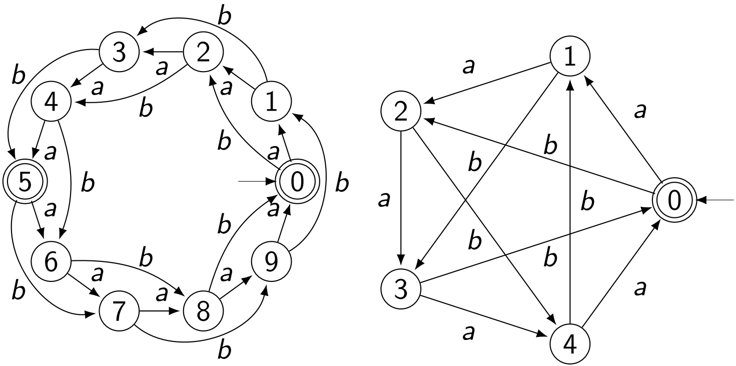
$val(w0) = 2 \cdot val(w)$

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

states represent $val(w)$ modulo 3



[M] E. 2.7



☒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.

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;

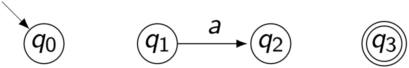
– $A \subseteq Q$ *accepting* states;

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

[M] D 2.11 Finite automaton

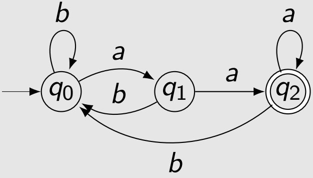
[L] D 2.1 Deterministic finite accepter, has ‘final’ states

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