Foundations of Computer Science

Fundamentele Informatica 1

Hendrik Jan Hoogeboom
Jeannette de Graaf

Bachelor Informatica (& specialisaties)
Universiteit Leiden

Najaar 2020

Universiteit
Leiden
Leiden Institute of
Advanced Computer Science

Hoofdstuk 9

Talen en Automaten

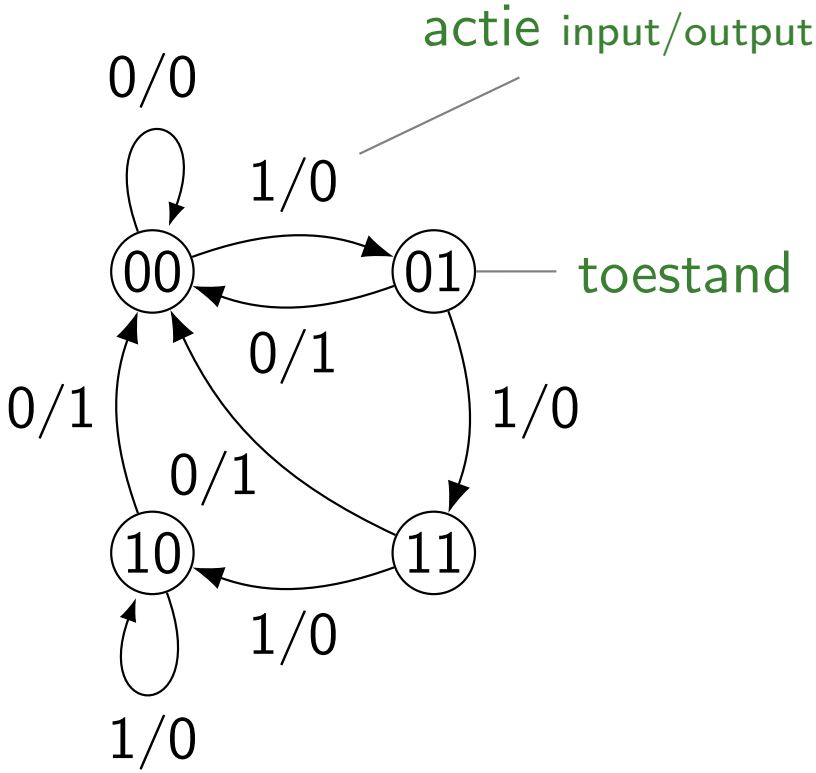
woord $\sim$ reeks acties

taal $\sim$ gedrag van systemen

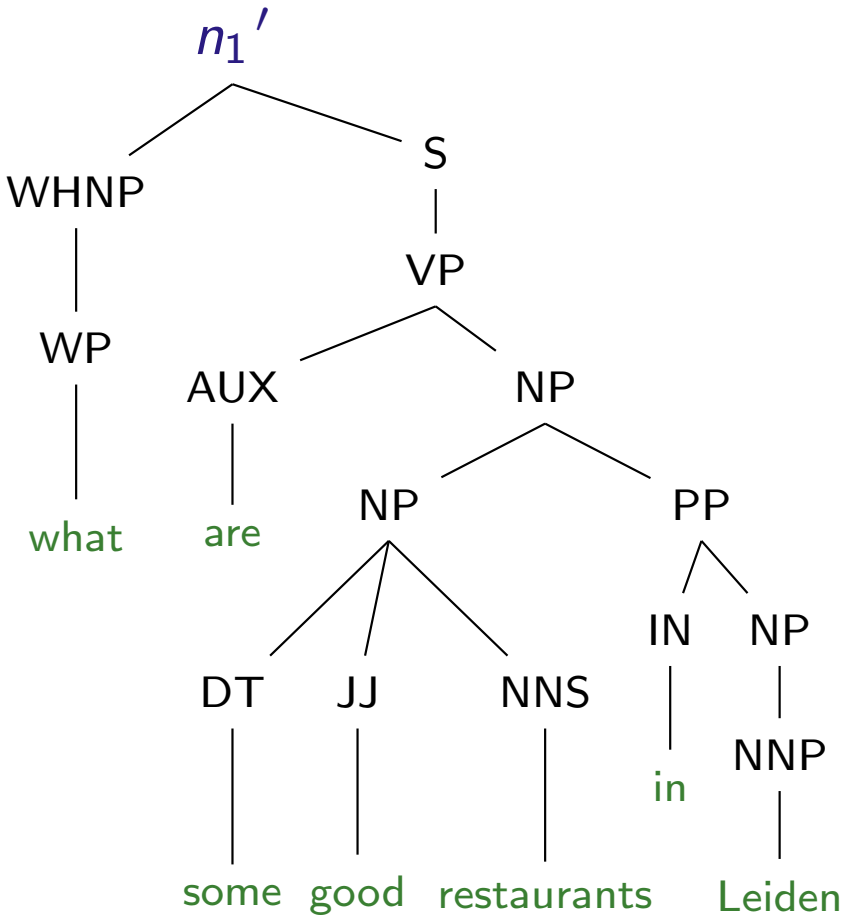
- specificeren $L = \{w_1, w_2, w_3, \ldots\}$
  $L = \{\ w \mid w$ heeft eigenschap P $\}$
  - taal operaties
    reguliere talen Ch. 12.4
  - grammatica
    context-vrije grammatica ⊠ Ch. 12.6
  - (niet-deterministische) automaat
    eindige automaat, Turing machine ⊠
- herkennen $w \in L$?
  - (deterministische) automaat
    eindige automaat Ch. 12.5
    Turing machine ⊠ Ch. 13.4

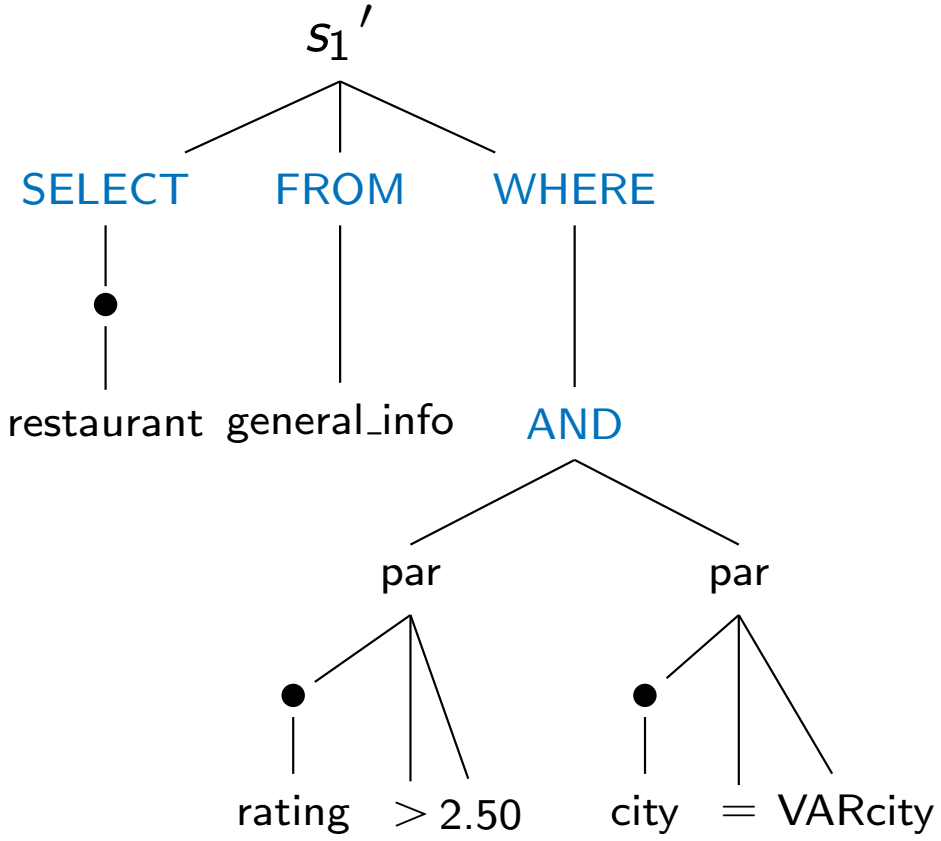
Ch. 12. Languages, Automata, Grammars

Ch. 13. Finite State Machines and Turing Machines



Digital Technique by Todor Stefanov, Leiden University





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

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

Chomsky hierarchy

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

letter, symbool σ $0, 1$ a, b, c

alfabet Σ, A $\{a, b, c\}$

(eindig, niet-leeg)

string, woord over Σ eindig rijtje

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

lege string λ Λ, ε, 1

Σ^* alle strings $\{ \lambda, a, b, c, aa, ab, ac, ba, bb, \ldots, aaa, aab, \ldots \}$

lengte $|w|$ aantal symbolen ~~$\ell(w)$~~

$|abbab| = 5$ $|\lambda| = 0$

taal over Σ $L \subseteq \Sigma^*$

eindige, niet-lege, verzameling

- $\{0, 1\}$ $\{a, b, c\}$ $\{0, 1, \uparrow, \boxplus\}$
- $\{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\} \times \{H, V, B, 10, 9, \ldots, 2, A\}$ $(\spadesuit, V)$
- $\Sigma_G = \{ (u, v) \mid (u, v) \in E \}$
  taal: paden in graaf
- $\{$ if, then, function, return, $\ldots \}$

concatenatie $a_1 \ldots a_m \cdot b_1 \ldots b_n$

$ab \cdot babb = abbabb$

wordt vaak weggelaten $u \cdot v = uv$

Thm. 12.1

$w\lambda = \lambda w = w$ eenheid

$(xy)z = x(yz)$ associatief

niet commutatief bei·aard $\neq$ aard·bei

$|xy| = |x| + |y|$

macht $w^n = \underbrace{w \cdot \ldots \cdot w}_{n \text{ keer}}$

$w^0 = \lambda$ (!)

$w^1 = w$, $w^2 = ww$, ...

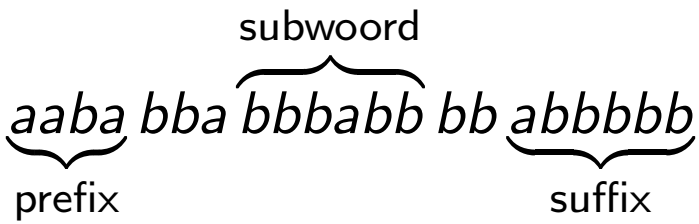
$w^{n+1} = w^n \cdot w$

$(ab)^3 bab^4 a = abababbabbbba$

$(\text{ker})^5 = \text{kerk}\cdot\text{erker}\cdot\text{kerker}$

$x^{m+n} = x^m \cdot x^n$

$|w^n| = n \cdot |w|$



$x, y \in \Sigma^*$

x is een *deelwoord* van y als $y = u{\cdot}x{\cdot}v$ voor $u, v \in \Sigma^*$

prefix als $y = x{\cdot}v$ voor $v \in \Sigma^*$

suffix als $y = u{\cdot}x$ voor $u \in \Sigma^*$

$=$ *subwoord*

lekkerkerker deelwoord kerk

twee *voorkomens*

aarzelaar prefix én suffix

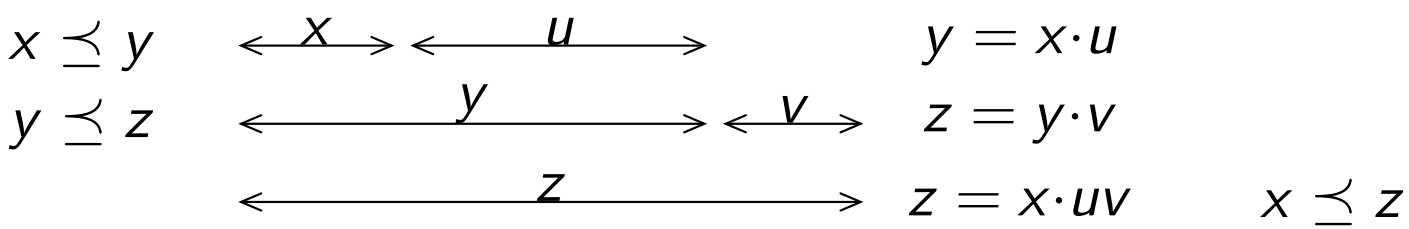
$x \preceq y$ als x prefix van y

partiële ordening

– reflexief $x \preceq x$ voor alle $x \in \Sigma^*$

– anti-symmetrisch als $x \preceq y$ en $y \preceq x$ dan $x = y$ voor alle . . .

– transitief als $x \preceq y$ en $y \preceq z$ dan $x \preceq z$



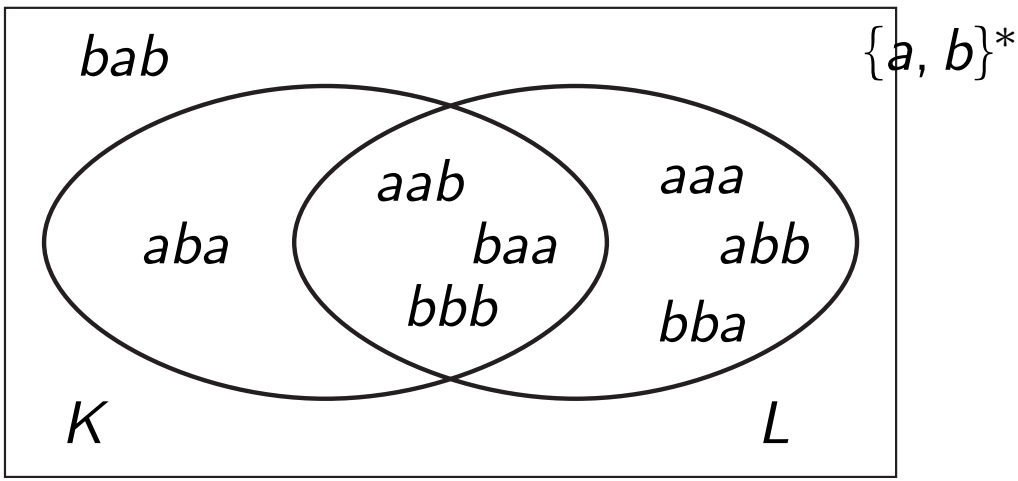
ook: deelwoord en suffix

opsommen, eigenschap

- $\varnothing$ nul woorden
- $\{ \lambda \}$ één woord
- $\{ a^m b^n \mid m, n \in \mathbb{N} \} = \{ \lambda, a, b, aa, ab, bb, aaa, aab, abb, bbb, \ldots \}$
- $\{ a^n b^n \mid n \in \mathbb{N} \} = \{ \lambda, ab, aabb, aaabbb, a^4 b^4, \ldots \}$
- $\{ a^m b a^n \mid m, n \in \mathbb{N} \}$ precies één b
- $L = \{ x \in \{a, b, c\}^* \mid x$ heeft subwoord $aab \}$

$K = \{\, x \in \{a, b\}^* \mid x \text{ heeft een even aantal } a\text{'s} \,\}$
$= \{\, \lambda, b, aa, bb, aab, aba, baa, bbb, \ldots \,\}$

$L = \{\, x \in \{a, b\}^* \mid x \text{ heeft (ergens) twee gelijke letters achter elkaar} \}$
$= \{\, aa, bb, aaa, aab, abb, baa, bba, bbb, \ldots \,\}$



$aab, baa, bbb \in K \cap L$ $\quad aba \in K \setminus L$ $\quad aaa, abb, bba \in L \setminus K$ $\quad bab \in (K \cup L)^c$

doorsnede, vereniging, complement

$K = \{\, x \in \{a, b\}^* \mid x \text{ heeft een even aantal } a\text{'s} \,\}$
$= \{\, \lambda, b, aa, bb, aab, aba, baa, bbb, \ldots \,\}$

$L = \{\, x \in \{a, b\}^* \mid x \text{ heeft (ergens) twee gelijke letters achter elkaar} \}$
$= \{\, aa, bb, aaa, aab, abb, baa, bba, bbb, \ldots \,\}$

- $K \cap L = \{\, aa, bb, aab, baa, bbb, aaaa, aabb, baab, \ldots \,\}$
- $\{a, b\}^* - L = \{\, \lambda, a, b, ab, ba, aba, baba, ababa, \ldots \,\}$
- $K - L = \{\, \lambda, b, aba, abab, baba, babab, abababa, \ldots \,\}$
- $L - K = \{\, aaa, abb, bba, aaab, aaba, baaa, baab, \ldots \,\}$

spiegelbeeld w^R, $\mathrm{mir}(w)$

$$\mathrm{mir}(a_1 a_2 \ldots a_n) = a_n \ldots a_2 a_1$$

voor talen $\mathrm{mir}(L) = \{\, \mathrm{mir}(w) \mid w \in L \,\}$

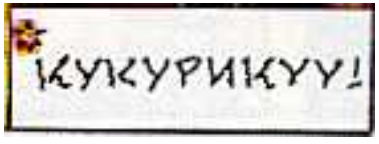
inductief

- $\mathrm{mir}(\lambda) = \lambda$
- $\mathrm{mir}(w{\cdot}a) = a{\cdot}\mathrm{mir}(w)$ voor $a \in \Sigma$, $w \in \Sigma^*$

palindroom $w = \mathrm{mir}(w)$

- snorfrons, netebeten
- *aabaa*, *abba·abba*

$\mathrm{PAL} = \{\, w \in \Sigma^* \mid w = \mathrm{mir}(w) \,\}$

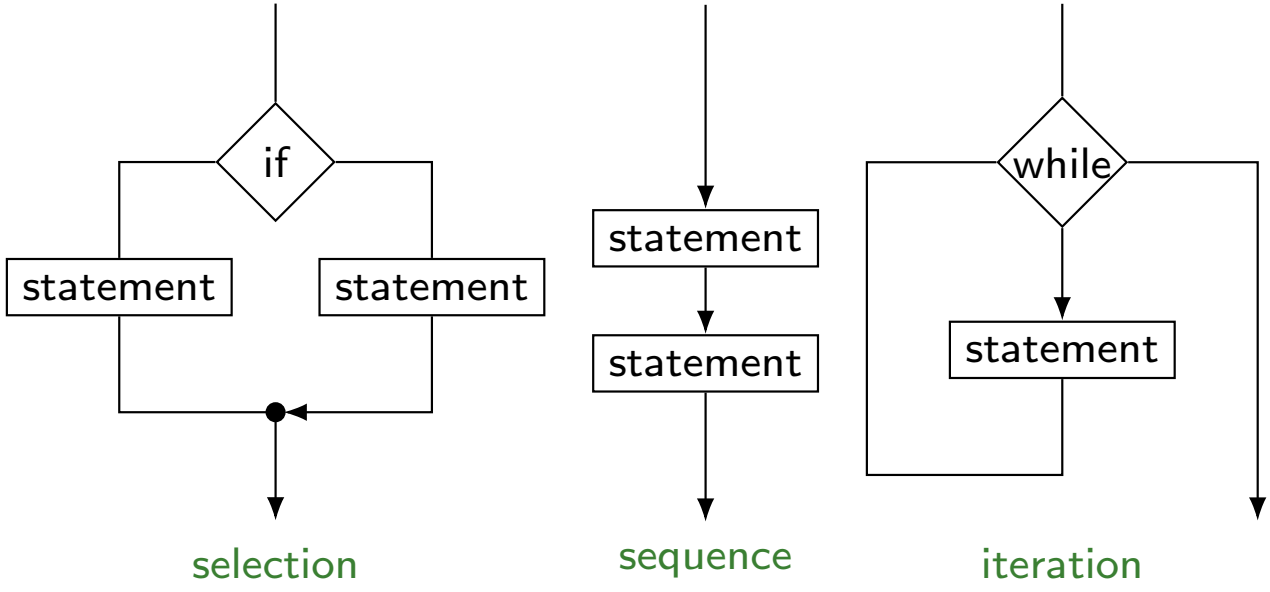


Boolese operaties

$\cup$	vereniging	$\vee$
$\cap$	doorsnede	$\wedge$
^{c}	complement	$\neg$

reguliere operaties

$\cup$	vereniging
$\cdot$	concatenatie
*	ster



concatenatie

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

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

een	$\{\lambda\}{\cdot}L = L{\cdot}\{\lambda\} = L$
nul	$\varnothing{\cdot}L = L{\cdot}\varnothing = \varnothing$
associatief	$(KL)M = K(LM)$

- $\{\, \lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots \,\} \cdot \{\, b \,\} =$
  $\{\, \lambda b, ab, bb, aab, abb, bab, bbb, aaab, aabb, \ldots \,\}$
- $\{\, a, ab, abb, abbb, abbbb, \ldots \,\} \cdot \{\, a, ba, aba, baba, ababa, \ldots \,\} =$
  $\{\, aa, aba, aaba, abba, ababa, abbba, aababa, abbaba, abbbba, \ldots \,\}$

  $abb \cdot aba = ab \cdot baba$

$$K^n = \underbrace{K \cdot K \cdot \ldots \cdot K}_{n \text{ keer}}$$

$$K^n = \{\, w_1 w_2 \ldots w_n \mid w_1, w_2, \ldots, w_n \in K \,\} \qquad \text{vaste } n \text{ 'buiten'}$$

- $\{\, a, bb \,\}^3 = \{\, aaa, aabb, abba, ab^4, bbaa, bbabb, b^4a, b^6 \,\}$
- $\{\, \lambda, a, ab \,\}^3 = \{\, \lambda, a, aa, ab, aaa, aab, ab{\cdot}a, aaab, aaba, abaa,$
  $abab, a{\cdot}ab{\cdot}ab, abaab, ababa, ababab \,\}$
- $L = \{\, a^n b a^n \mid n \in \mathbb{N}\} = \{b, aba, aabaa, a^3ba^3, \ldots\}$
  $L^2 = \{\, a^m b a^{m+n} b a^n \mid m, n \in \mathbb{N}\}$
- $(\, \{a\}^*\{b\}\{a\}^* \,)^3 = \{\, w \in \{a, b\}^* \mid w \text{ bevat 3 voorkomens van } b \,\}$
- $\{\, \lambda, a, a^4, a^9, a^{16}, \ldots\}^4 = \{a\}^*$ Lagrange's four-square theorem

(Kleene) ster

$$K^* = \underbrace{K \cdot K \cdot \ldots \cdot K}_{\text{willekeurig}}$$

$$K^* = \{\, w_1 w_2 \ldots w_n \mid w_1, w_2, \ldots, w_n \in K, n \in \mathbb{N} \,\} \qquad n \text{ varieert 'binnen'}$$

- $\{a, b\}^*$ alle woorden over $\{a, b\}$
  $\{a, b\}^n$ alle woorden over $\{a, b\}$ van lengte n
- $\{a\}^* \cdot \{b\} = \{\lambda, a, aa, aaa, \ldots\} \cdot \{b\} = \{b, ab, aab, aaab, \ldots\}$
- $(\, \{a\}^* \cdot \{b\} \,)^* =$
  $\{b, ab, aab, aaab, \ldots\}^* = \{\lambda, b, ab, bb, aab, abb, bab, bbb, aaab, \ldots\}$
  $= \{a, b\}^* \cdot \{b\} \cup \{\lambda\}$

korte notatie $\{a\} \leftrightarrow a$ $a^* b$, $(a^* b)^*$ zie ook straks

$K^0 = \{\lambda\}$
$K^{n+1} = K^n K \qquad n \geqslant 0$

$\varnothing^0 = \{\lambda\}$ afspraak!

$K^1 = K^0 \cdot K = \{\lambda\} \cdot K = K$ (gelukkig)

$K^m K^n = K^{m+n}$

Kleene ster

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

Kleene plus

$K^+ = \bigcup_{n \geqslant 1} K^n$

$K^* = K^+ \cup \{\lambda\}$

alfabet Σ

reguliere talen

- $\varnothing$, $\{\lambda\}$ en $\{a\}$ zijn regulier $\quad a \in \Sigma$
- als K en L regulier, dan ook $K \cup L$, $K \cdot L$ en K^*

- $\{a\}^*\{b\}\{a\}^*$ één b $\quad a^*ba^*$
- $\{b\}^*\{a\}\{b\}^*\{a\}\{b\}^*$ precies twee a's $\quad b^*ab^*ab^*$
- $(\{a\} \cup \{b\})^*\{b\} = \{a, b\}^*\{b\}$ eindigt op b
- $\{a\}\{a\}^*\{b\}\{b\}^*$ één of meer a's gevolgd door één of meer b's $\quad aa^*bb^*$
- $(\ \{a\}^*\{b\}\,\{a\}^*\{b\}\ )^*\ \{a\}^*$ even aantal b $\quad (\ a^*ba^*b\ )^*\ a^*$

regulier $\varnothing$, $\{\lambda\}$, $\{a\}$ $K \cup L$, $K \cdot L$, K^*

elke *eindige* taal is regulier

met inductie twee keer

– enkel woord inductie naar lengte
  basis $\{\lambda\}$, $\{a\}$
  inductie $\{w\} \cdot \{\sigma\}$

– eindige verzameling inductie op aantal
  basis $\varnothing$
  inductie $K \cup \{w\}$

voortaan: regulier $\sim$ eindige talen en operaties $\cup$, $\cdot$ en *

$\{ab, bab\}^* \{\lambda, bb\}$

ipv $((\{a\} \cdot \{b\}) \cup (\{b\} \cdot \{a\} \cdot \{b\}))^* \cdot (\{\lambda\} \cup (\{b\} \cdot \{b\}))$

notatie

syntax regex

- $\varnothing$, λ en a zijn regex $\quad a \in \Sigma$
- als r_1 en r_2 regex, dan ook $(r_1 + r_2)$, $(r_1 r_2)$ en (r_1^*)

betekenis expressie $\quad r \mapsto L(r) \subseteq \Sigma^*$ $\quad$ taal

taal van r

semantiek regex

- $L(\varnothing) = \varnothing$, $L(\lambda) = \{\lambda\}$, en $L(a) = \{a\}$
- als r_1 en r_2 regex, dan $L(r_1 + r_2) = L(r_1) \cup L(r_2)$, $L(r_1 r_2) = L(r_1) \cdot L(r_2)$ en $L(r_1^*) = L(r_1)^*$

haakjes weglaten

– associativiteit $+$, $\cdot$

– voorrangsregels

* voor $\cdot$ voor $+$

$aab + ab^*a + (ab)^*a$

$\{aab\} \cup \{aa, aba, abba, abbba, \ldots\} \cup \{a, aba, ababa, abababa, \ldots\}$

verschil verzamelingsnotatie en reguliere expressie

$\{ab, bab\}^* \{\lambda, bb\}$ vs $(ab + bab)^*(\lambda + bb)$

trivial	$E + 0 = 0 + E = E$	where $0 = \varnothing$
	$E \cdot 0 = 0 \cdot E = 0$	
	$E \cdot 1 = 1 \cdot E = E$	where $1 = \Lambda$
associative	$(E_1 + E_2) + E_3 = E_1 + (E_2 + E_3)$	
	$(E_1 \cdot E_2) \cdot E_3 = E_1 \cdot (E_2 \cdot E_3)$	
distributive	$E(E_1 + E_2) = EE_1 + EE_2$	
	$(E_1 + E_2)E = E_1E + E_2E$	
commutative	$E_1 + E_2 = E_2 + E_1$	
unrolling	$E^* = 1 + EE^* = 1 + E^*E$	$*$-rules
denesting	$(E_1 + E_2)^* = E_1^* \cdot (E_2 \cdot E_1^*)^*$	
sliding	$(E_1 \cdot E_2)^* \cdot E_1 = E_1 \cdot (E_2 \cdot E_1)^*$	
cyclic Zn	$E^* = (1 + E + E^2 + \ldots E^{n-1}) \cdot (E^n)^*$	
idempotency	$E + E = E$	
	$(E^*)^* = E^*$	

based on Jacques Sakarovitch: Automata and expressions

$\mathrm{mir}(x) = x^R$ spiegelbeeld

$\mathrm{mir}(L) = \{\, \mathrm{mir}(x) \mid x \in L \,\}$

als L regulier dan ook $\mathrm{mir}(L)$ *regulier*

(structurele) inductie

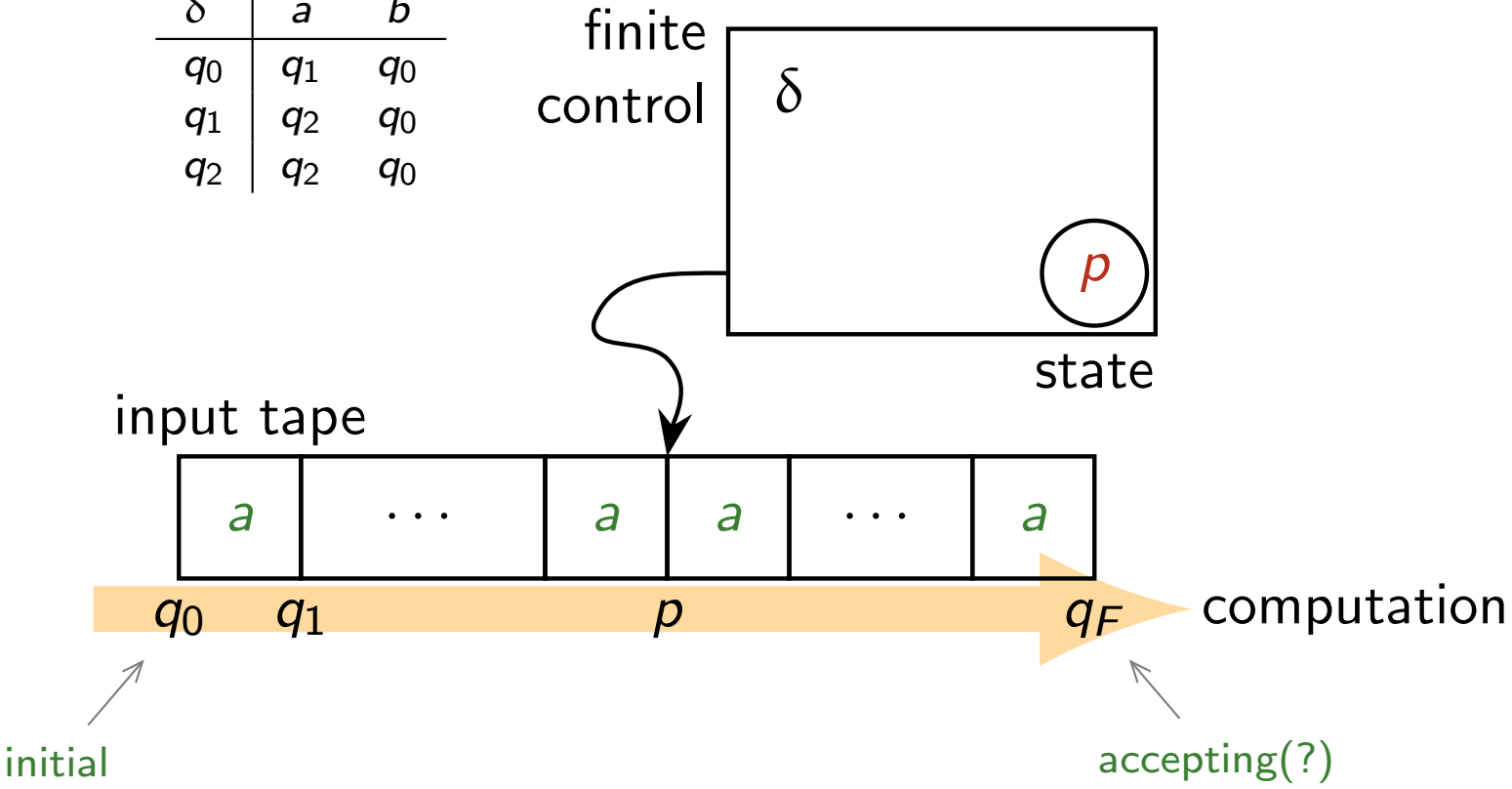
basis: eindige talen ok

$$\mathrm{mir}(L_1 + L_2) = \mathrm{mir}(L_1) \cup \mathrm{mir}(L_2)$$

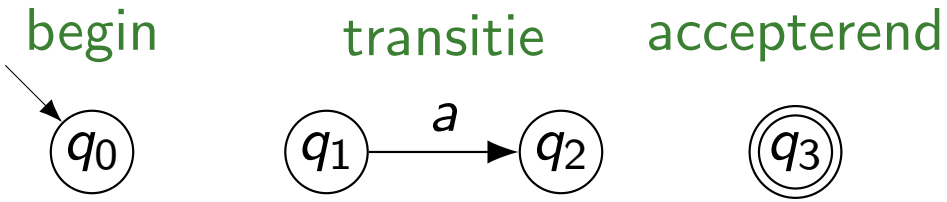
$$\mathrm{mir}(L_1 L_2) = \mathrm{mir}(L_2) \cdot \mathrm{mir}(L_1) \quad (!)$$

$$\mathrm{mir}(L^*) = \mathrm{mir}(L)^*$$

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

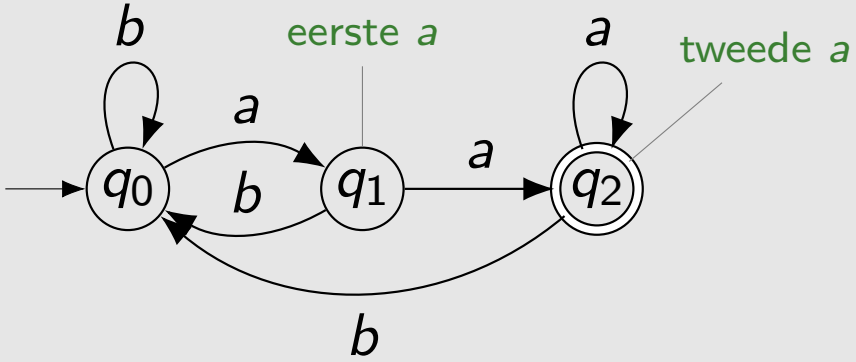


finite control $\rightsquigarrow$ gerichte graaf



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

gerichte graaf, tak-labels, begintoestand, accepterend (eindtoestand)

DFA

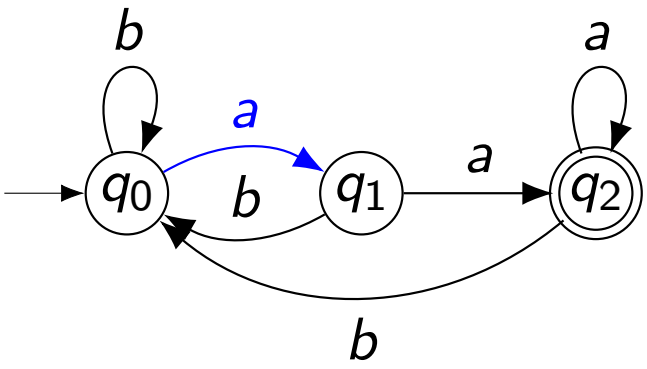
[deterministische] *eindige automaat* 5-tuple $M = (Q, \Sigma, \delta, q_{\text{in}}, A)$,

- Q eindige verzameling toestanden;
- Σ (eindig) input alfabet;
- $\delta : Q \times \Sigma \to Q$ transitiefunctie; *next state*
- $q_{\text{in}} \in Q$ begintoestand;
- $A \subseteq Q$ accepterende toestanden

functie $\rightsquigarrow$ *elke* toestand heeft voor *elke* letter een uitgaande tak

Schaum $M = (A, S, Y, s_0, F)$ (niet erg standaard)

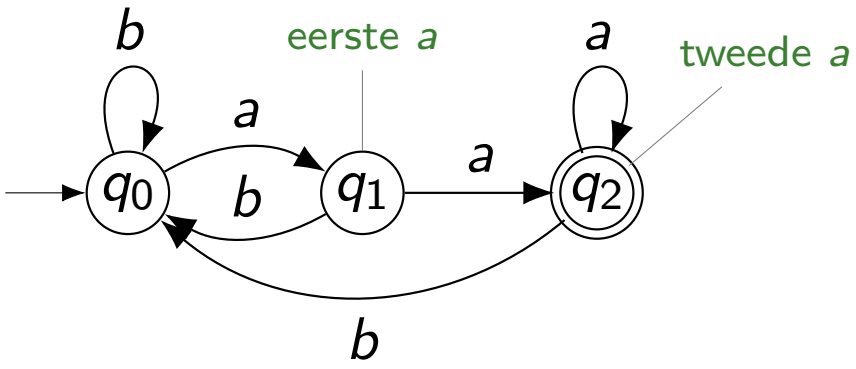
toestandsdiagram



transitiefunctie

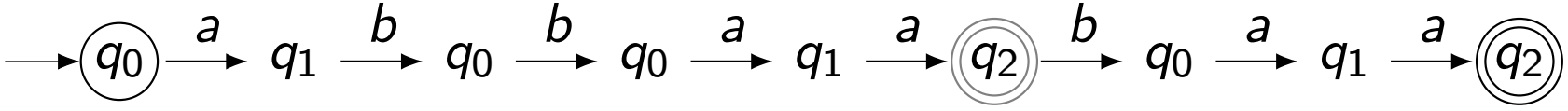
δ	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- δ als in matrix, rechts
- $q_{\text{in}} = q_0$
- $A = \{q_2\}$

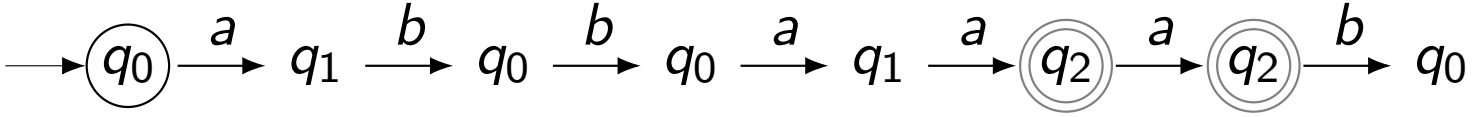


δ	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

accepterend

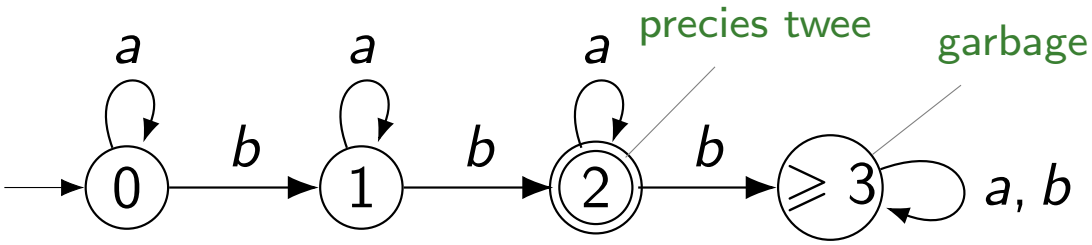


niet accepterend



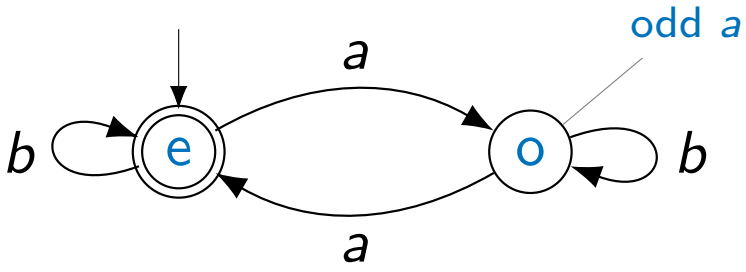
$L(M)$ *taal van* M labels paden van begin naar accepterende toestand

– precies twee b's



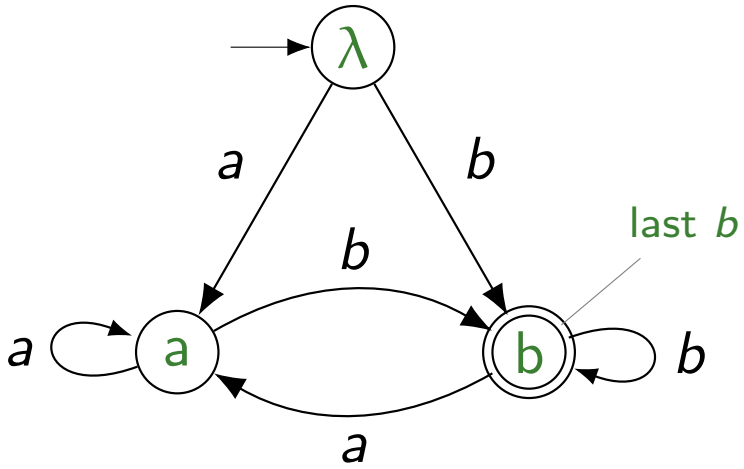
$a^* b a^* b a^*$

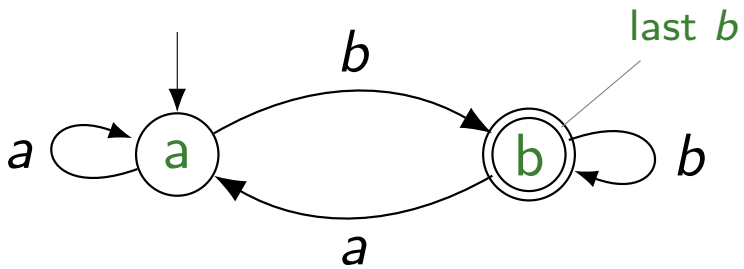
– even aantal a



$(\; b^* a b^* a \;)^* \; b^*$

– laatste letter b $\{a, b\}^* b$





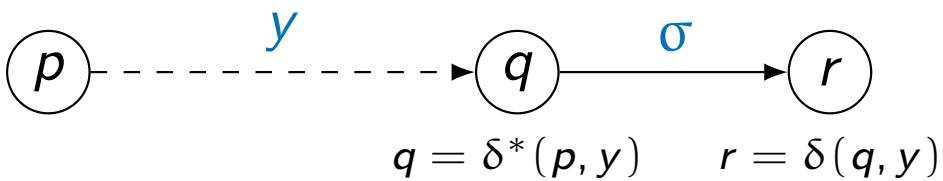
DFA $M = (Q, \Sigma, \delta, q_{\text{in}}, A)$ $\quad \delta : Q \times \Sigma \to Q$

extended transition function

$\delta^* : Q \times \Sigma^* \to Q$, such that

– $\delta^*(p, \lambda) = p$ for $p \in Q$

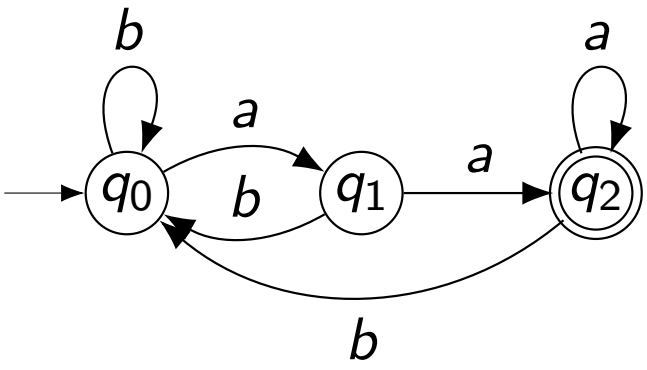
– $\delta^*(p, y{\cdot}\sigma) = \delta(\ \delta^*(p, y), \sigma\ )$ for $p \in Q, y \in \Sigma^*, \sigma \in \Sigma$

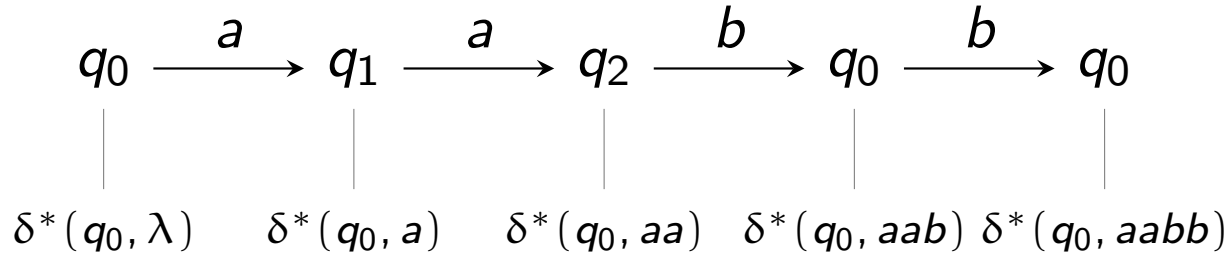


$q = \delta^*(p, w)$ iff
there is a path in [the transition graph of] M from p to q with label w.

alternatief

the *language accepted* by $M = (Q, \Sigma, \delta, q_{\text{in}}, A)$ is the set
$L(M) = \{\ x \in \Sigma^* \mid \delta^*(q_{\text{in}}, x) \in A\ \}$





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

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

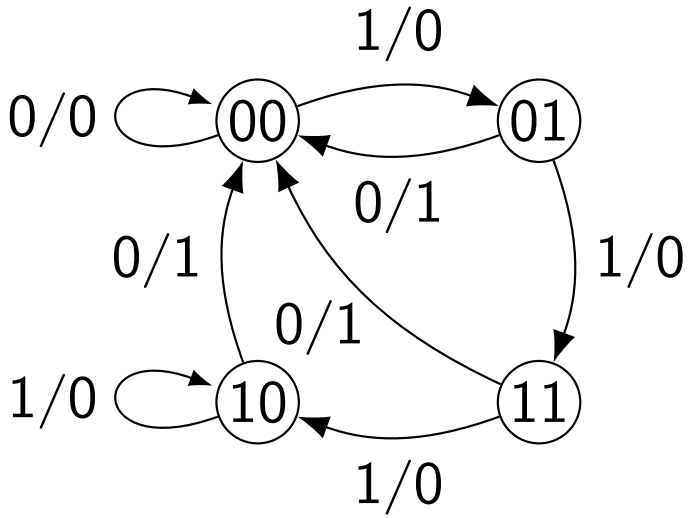
$$\delta^*(q_0, a) = \delta^*(q_0, \lambda{\cdot}a) = \delta(\delta^*(q_0, \lambda), a) = \delta(q_0, a) = q_1$$

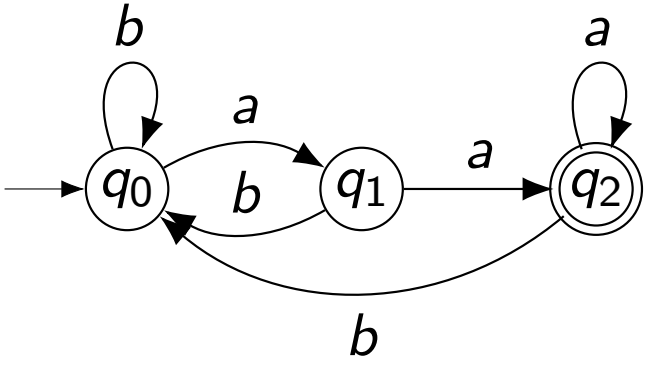
$$\delta^*(q_0, a{\cdot}a) = \delta(\delta^*(q_0, a), a) = \delta(q_1, a) = q_2$$

$$\delta^*(q_0, aa{\cdot}b) = \delta(\delta^*(q_0, aa), b) = \delta(q_2, b) = q_0$$

$$\delta^*(q_0, aab{\cdot}b) = \delta(\delta^*(q_0, aab), b) = \delta(q_0, b) = q_0$$

finite state machines digitale technieken vs. formele talen theorie





technische verschillen

– concrete implementatie: bits
– begin- en accepterende toestanden
– invoer/uitvoer gedrag vs. taal
  Moore / Mealy toestand / transitie
– (niet-)determinisme

taalfamilies

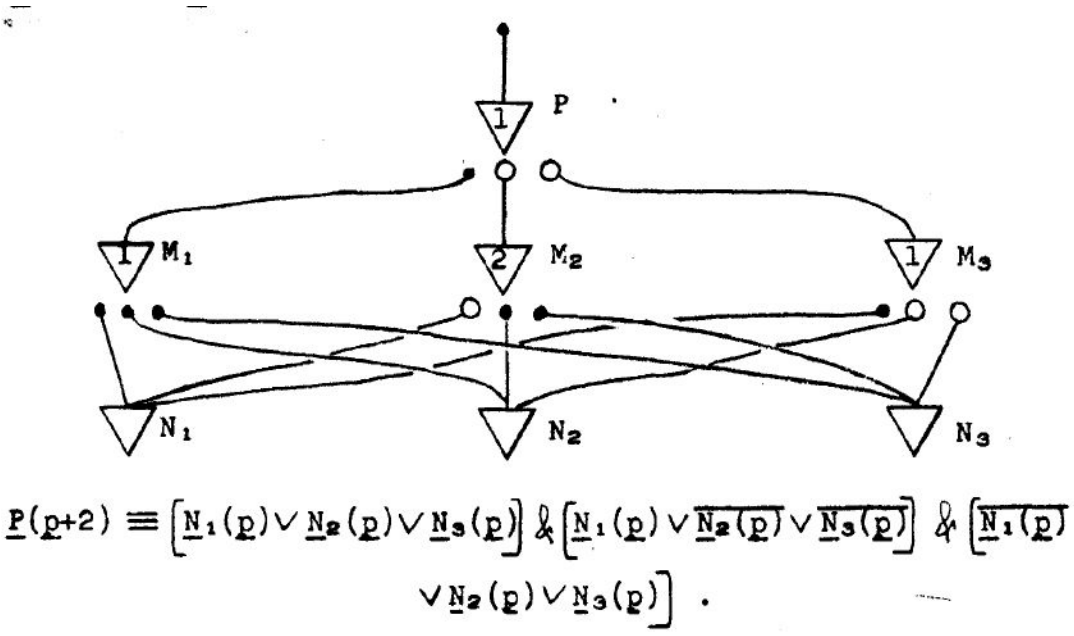
- regulier operaties $\cup$, $\cdot$, *
- eindige automaat

Thm. 12.2 Kleene

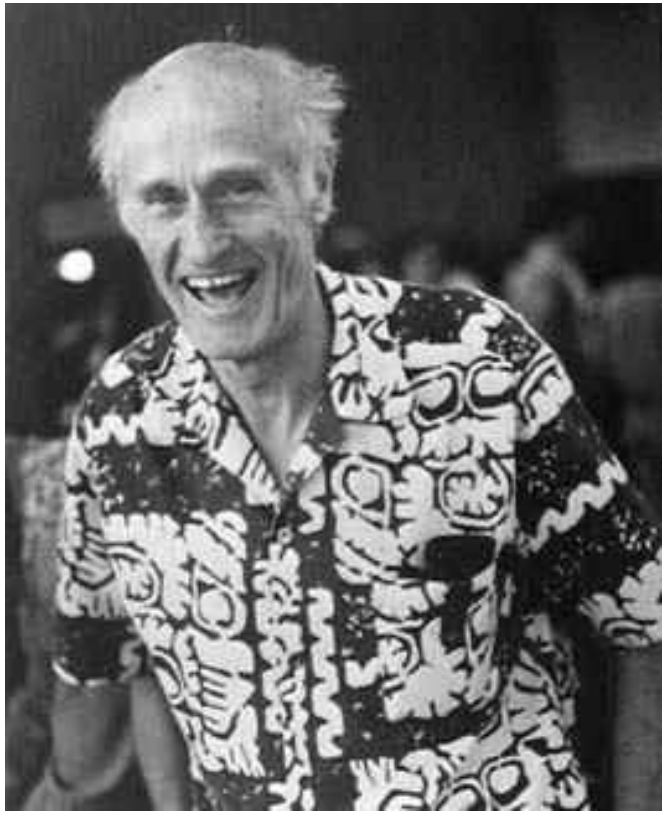
een taal L is regulier desda $L = L(M)$ voor een DFA M

zonder bewijs – zie Automata Theory

Stephen Cole Kleene



$$\underline{P}(p+2) \equiv \left[\underline{N}_1(p) \vee \underline{N}_2(p) \vee \underline{N}_3(p)\right] \,\&\, \left[\underline{N}_1(p) \vee \overline{\underline{N}_2(p)} \vee \overline{\underline{N}_3(p)}\right] \,\&\, \left[\overline{\underline{N}_1(p)} \vee \underline{N}_2(p) \vee \underline{N}_3(p)\right] .$$

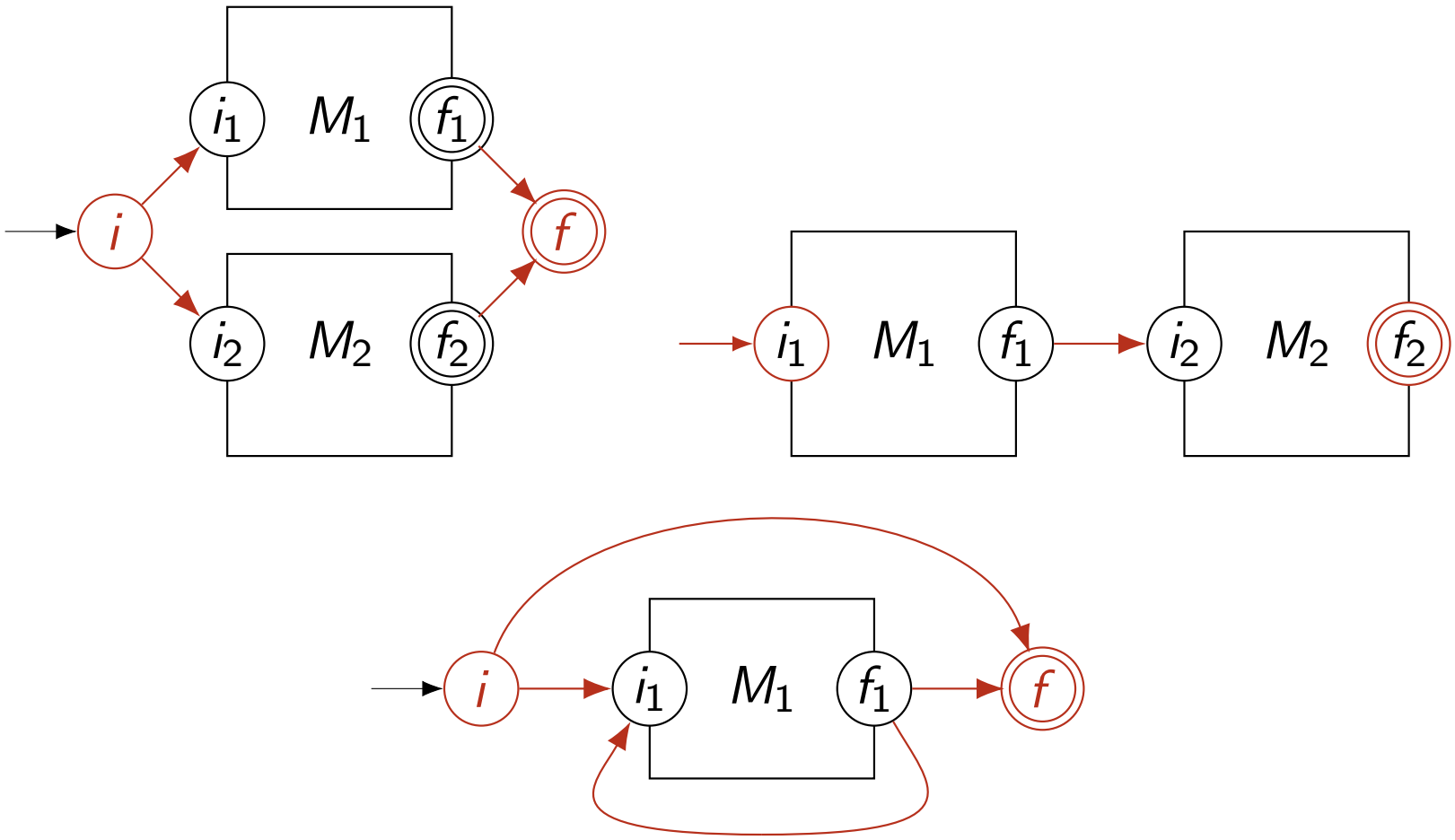


1909 Hartford CT, USA –
1994 Madison WI, USA

Kleene, S.C. (1956). Representation of Events in Nerve Nets and Finite Automata. Annals of Mathematics Studies vol. 34. Princeton University Press. pp. 3–41.

wikipedia MacTutor

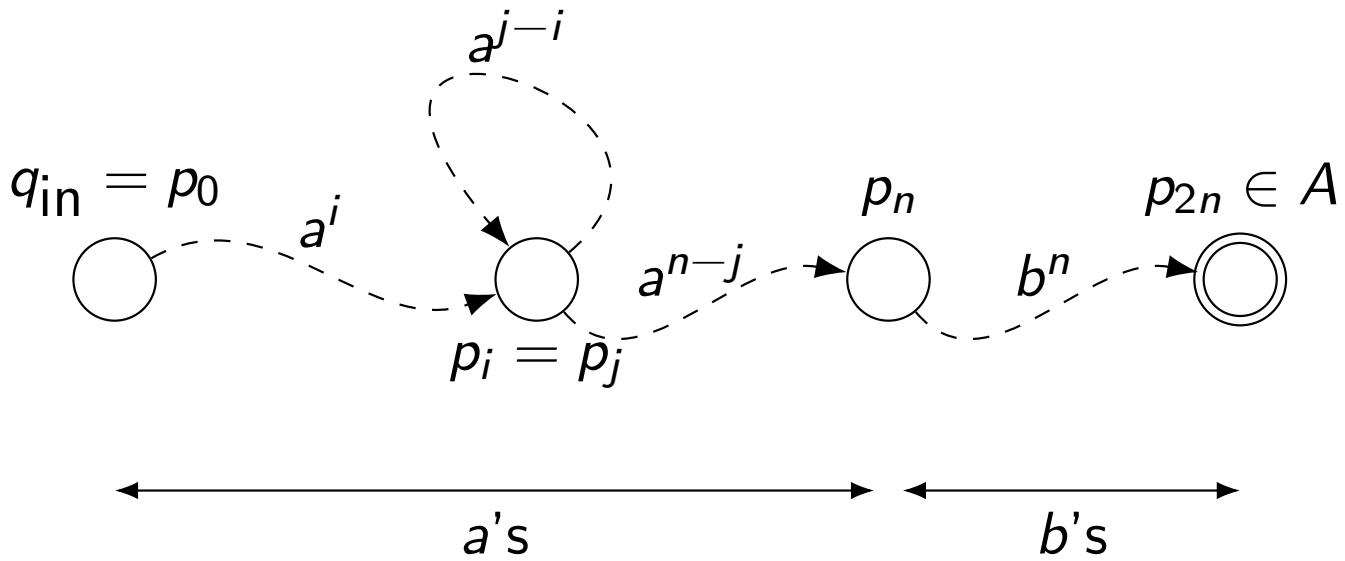
reguliere operaties $\Longrightarrow$ automaten



☒Thm 12.3 (Pumping Lemma)

$\{ a^n b^n \mid n \in \mathbb{N} \}$ is niet regulier

bewijs. vergelijk Thm. 8.2 over simpele paden



Door middel van tegenspraak. Stel $L = \{ a^m b^m \mid n \in \mathbb{N} \}$ is wél regulier. Reguliere talen worden geaccepteerd door eindige automaten.

Er is dus een DFA $M = (Q, \Sigma, \delta, q_{\text{in}}, A)$ die L accepteert, zeg met n toestanden.

Kijk naar een accepterend pad (berekening) voor de string $w = a^n b^n$. De begintoestand is q_{in}. Verder noem $p_k = \delta^*(q_{\text{in}}, a^k)$ voor $k = 0, 1, \ldots, n$. Na toestand p_n worden b's gelezen. Op het pad van a^n van $p_0 = q_{\text{in}}$ tot p_n liggen in totaal $n + 1$ toestanden. Die kunnen niet allemaal verschillend zijn, want M heeft maar n toestanden.

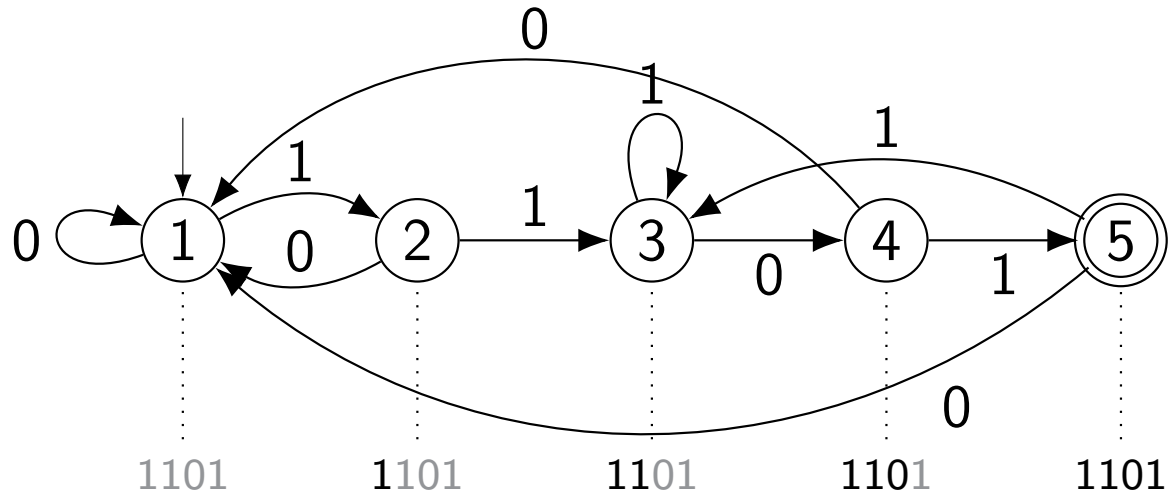
Er is dus een paar $p_i = p_j$ dat gelijk is. Het pad tussen p_i en p_j kan verwijderd worden, en wat overblijft is nog steeds een pad van q_{in} naar een toestand in A.

Omdat er a's zijn weggehaald, terwijl er nog steeds n letters b staan, wordt nu een woord $w' = a^m b^n$ geaccepteerd met $m < n$, dus $w' \notin L$.

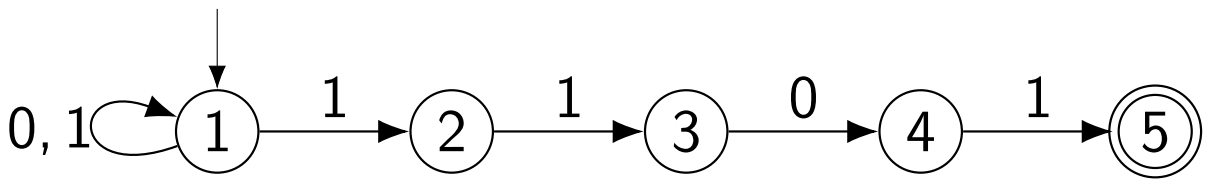
M accepteert dus niet L. Tegenspraak, L is niet regulier.

suffix [= eindigt op] 1101

algoritme deterministisch



beschrijving niet-deterministisch



er verandert weinig

NFA

[niet-deterministische] *eindige automaat* 5-tuple $M = (Q, \Sigma, \delta, q_{\text{in}}, A)$,

. . .

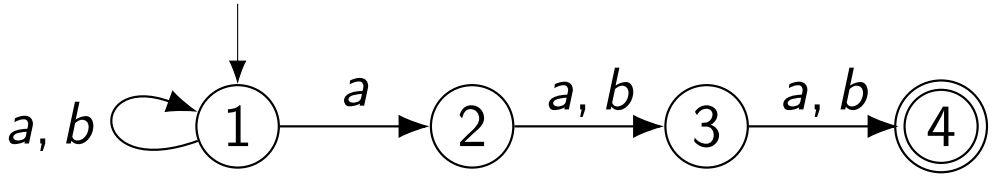
– $\delta \subseteq Q \times \Sigma \times Q$ transitie relatie

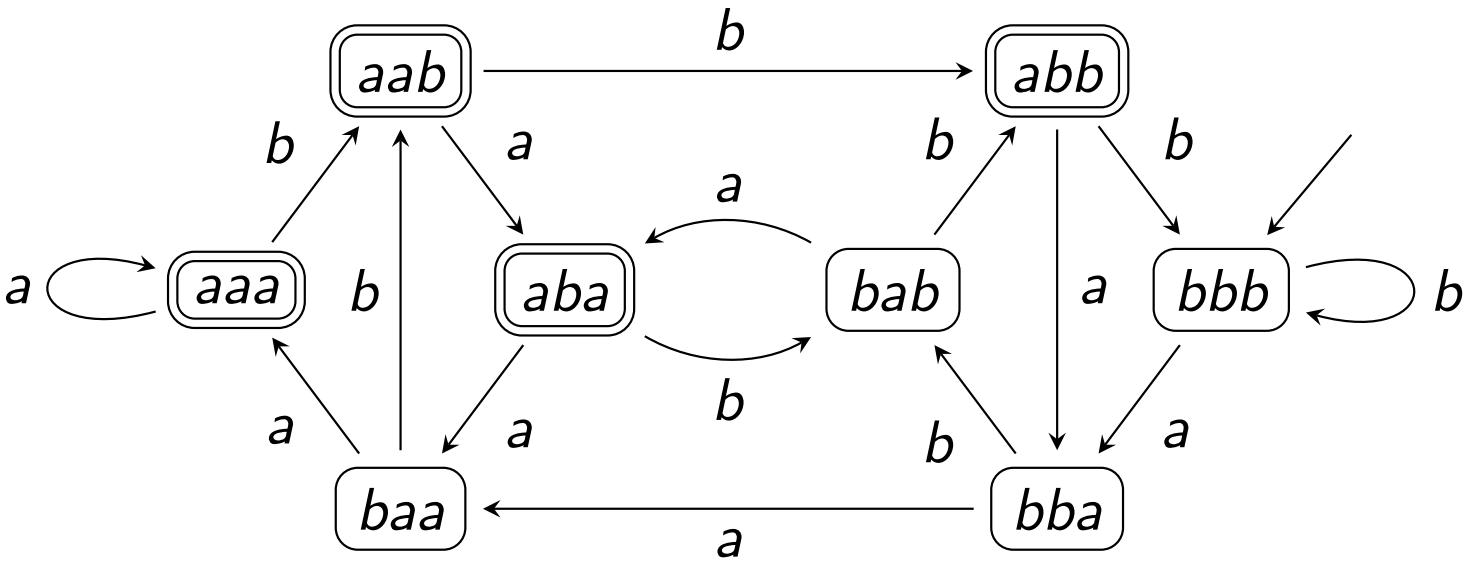
. . .

$L(M)$ *taal van* M labels paden van begin naar accepterende toestand

– soms meerdere paden (wel / niet accepterend)
– soms geen enkel pad ‘automaat blokkeert’
– vaak compacter, of eenvoudiger te construeren

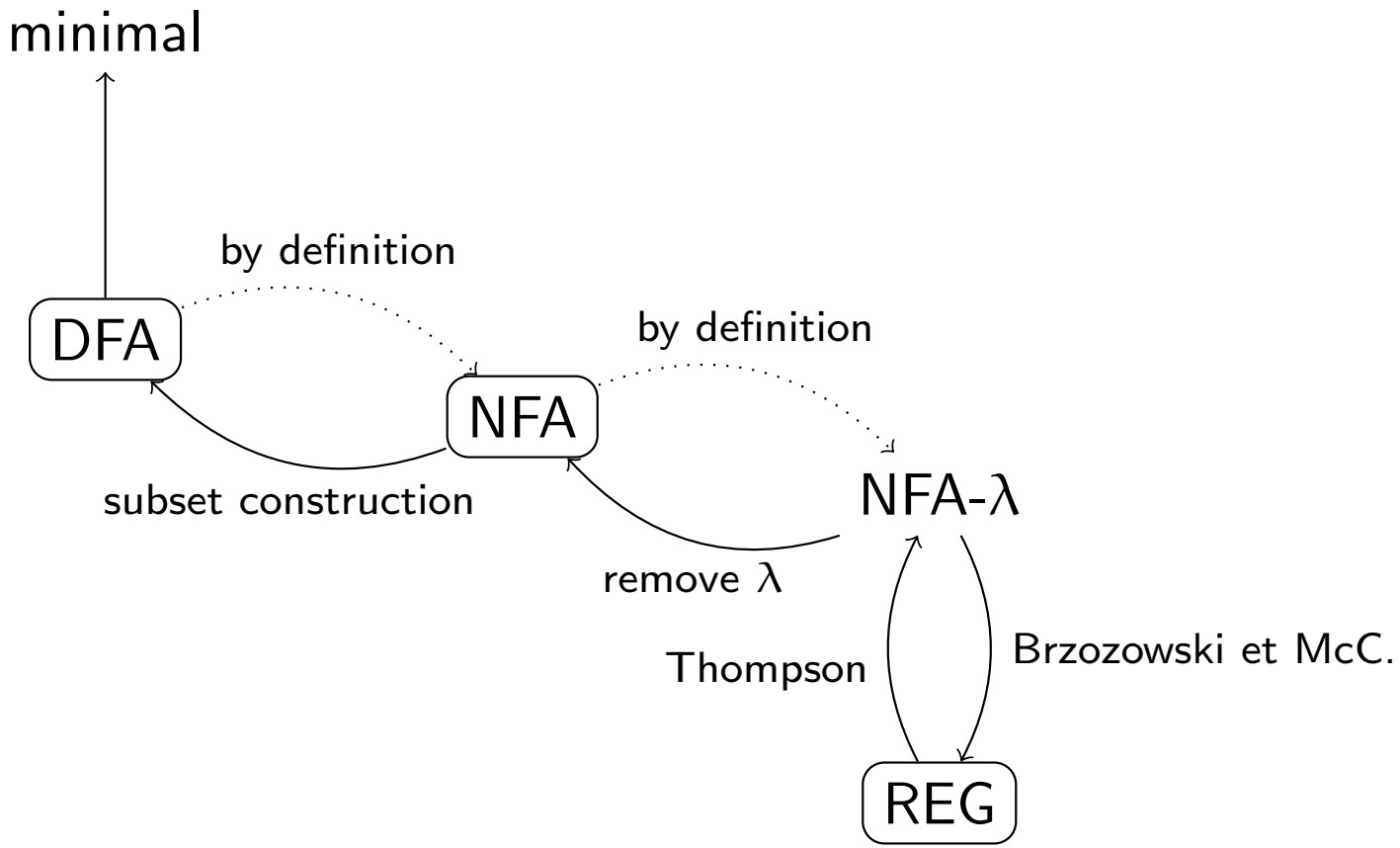
$\{a, b\}^*\ a\ \{a, b\}^2$





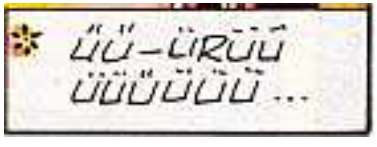
$n + 1$ versus 2^n states

☒equivalente modellen



talen over $\{a, b\}$ of $\{0, 1\}$

- even aantal a en eindigt op b
- start en eindigt met dezelfde letter
- aantal 1 is drievoud
- op elke oneven positie staat b
- als eindigt op b dan begint met b
- $\{\, 01, 011, 0111 \,\}^*$
- prefix / subwoord / suffix 110
- binair getal deelbaar door 3 $\quad 10101 \sim 16{+}4{+}1 = 21$



programmeren van eindige automaten

- toestand $\sim$ eigenschap
- garbage toestand (!)
- (niet-)deterministisch

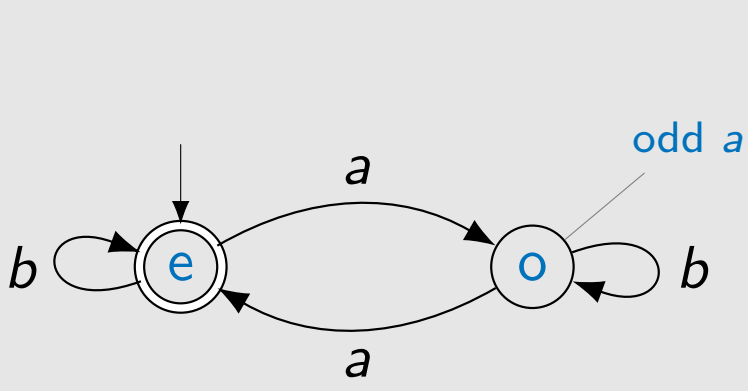
concepten

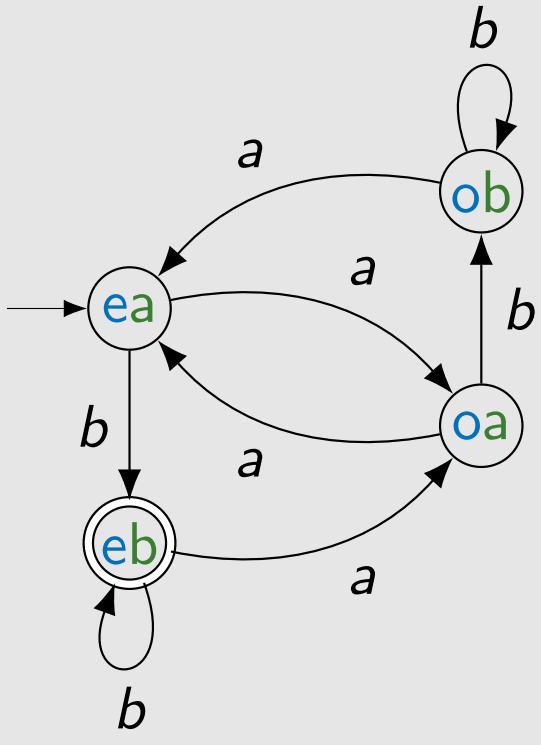
- even / oneven modulo m
  lengte, aantal letters a
- 'patroon'
  prefix, subwoord, suffix
- aantal voorkomens overlap $aba{\cdot}ba = ab{\cdot}aba$
- Boolese combinaties
  en, of, niet als. . . dan
- volgorde

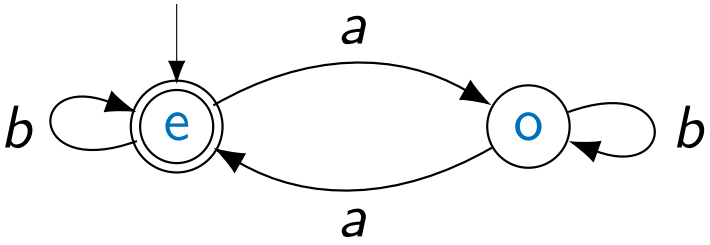
– even aantal a

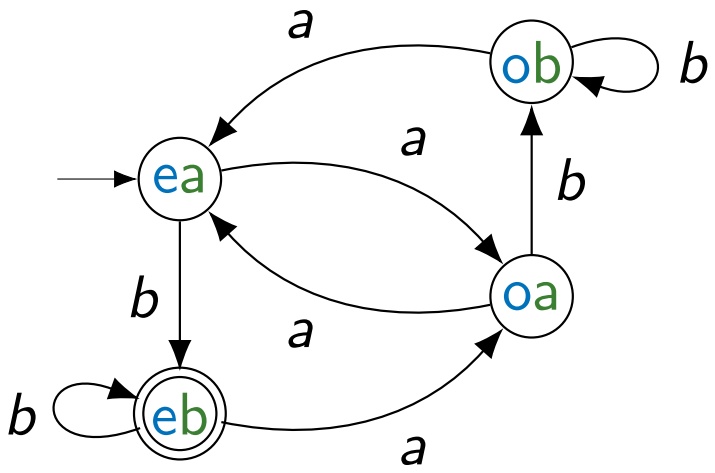
– laatste letter b

Example (even aantal a én eindigt met b)

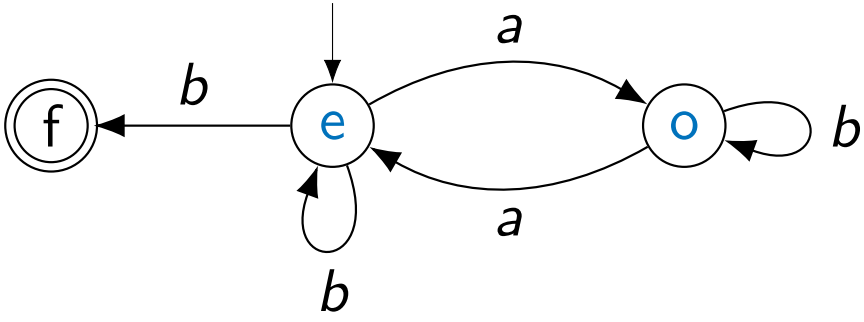


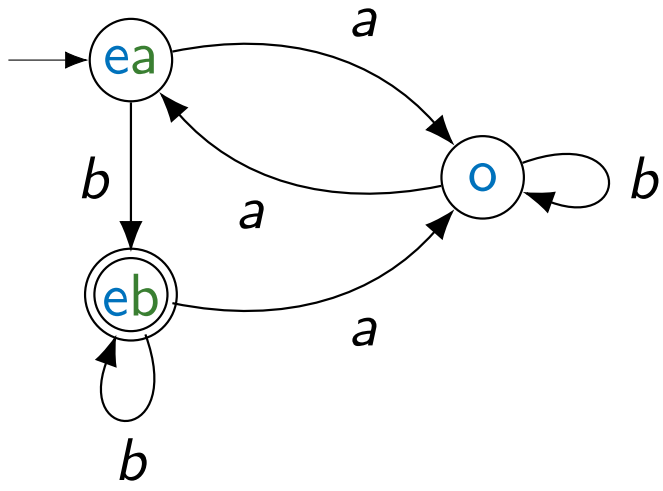




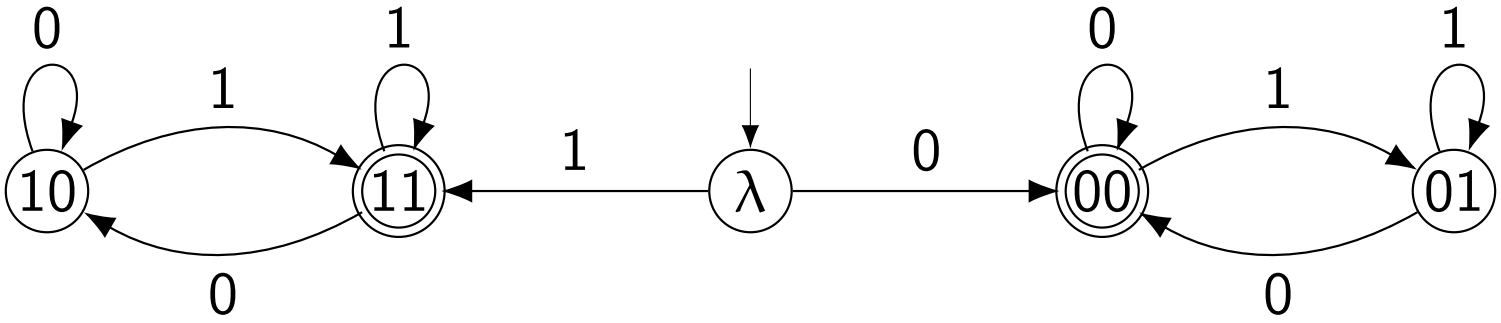


even aantal a én eindigt met b

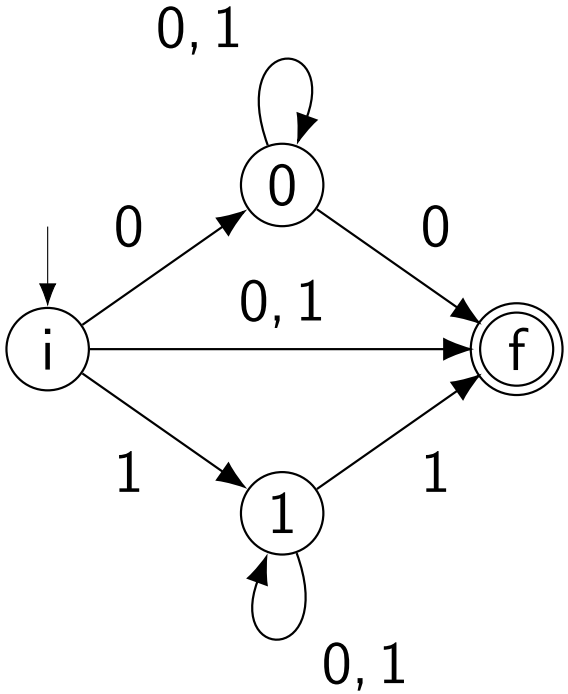




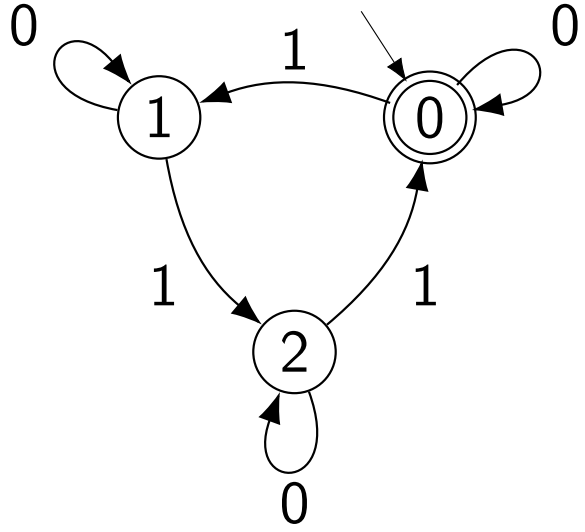
– starts and ends with the same symbol



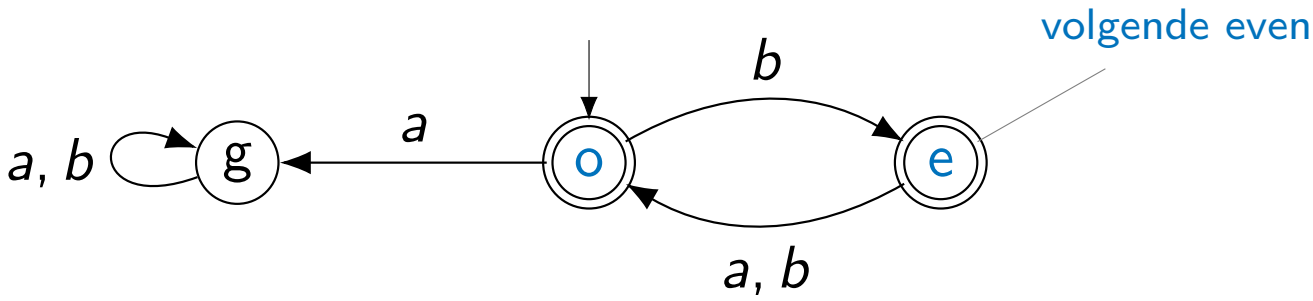
niet-deterministisch



aantal 1-en is drievoud



op elke oneven positie staat een b de eerste letter is de eerste positie

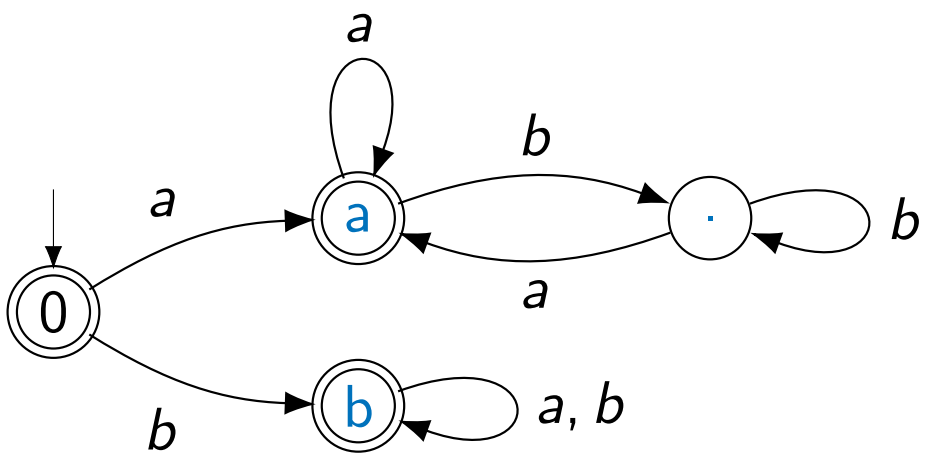


als w eindigt op een b dan begint w met een b

$a \cdots a$✓ $a \cdots b$× $b \cdots a$✓ $b \cdots b$✓ λ✓ (!)

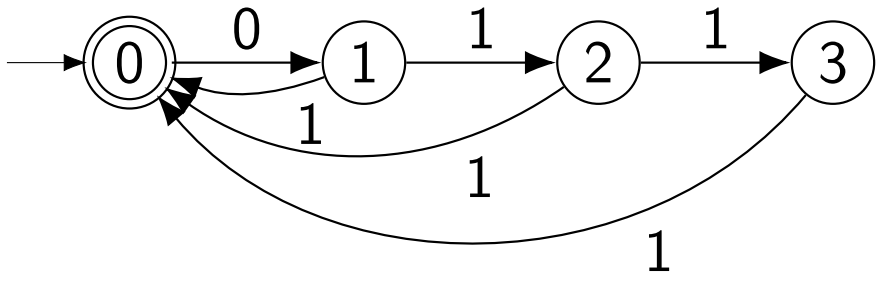
of: w begint met een b

of: w begint met een a en eindigt met een a

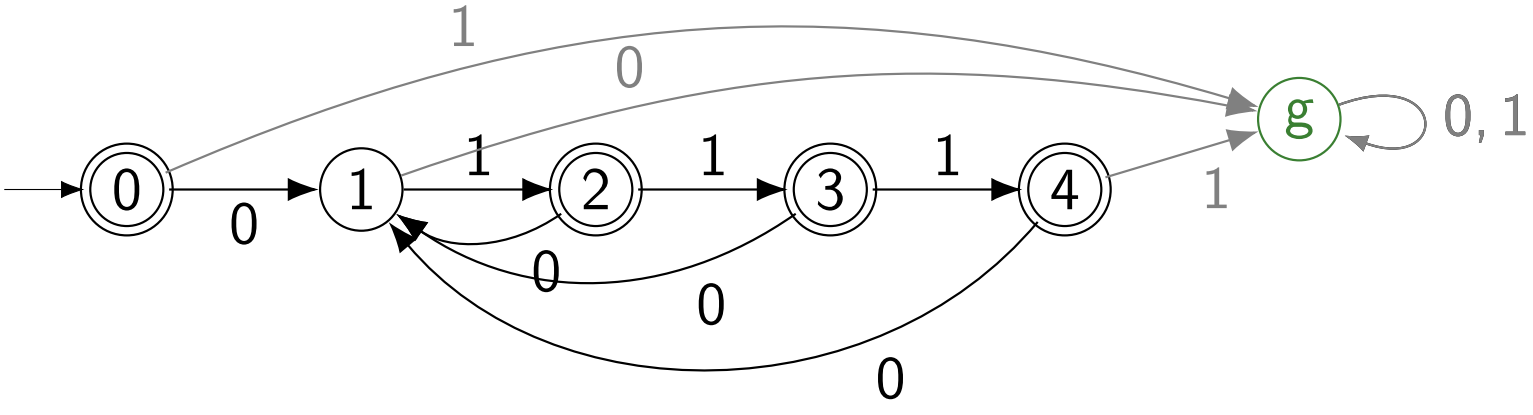


$\{ 01, 011, 0111 \}^*$

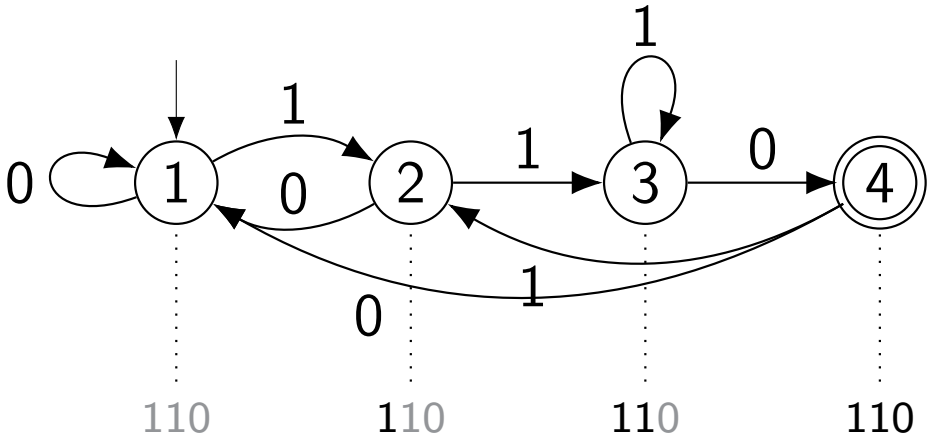
niet-deterministisch



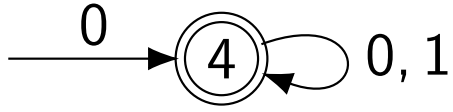
deterministisch ‘uitrollen’ + garbage



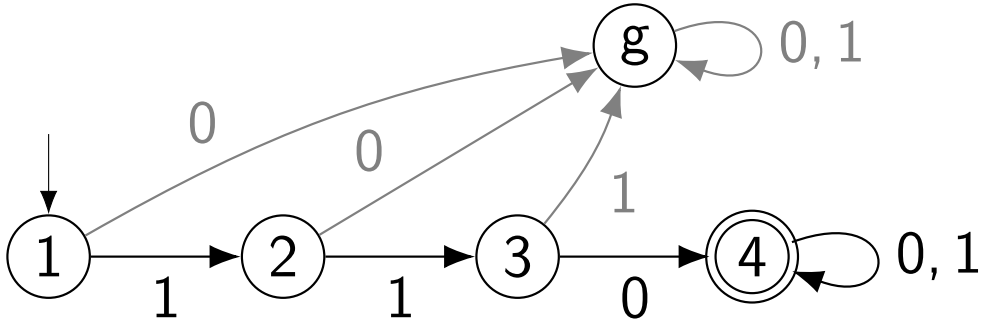
suffix [= eindigt op] 110



subwoord [= bevat] 110



prefix [= begint met] 110



binair woord, deelbaar door drie

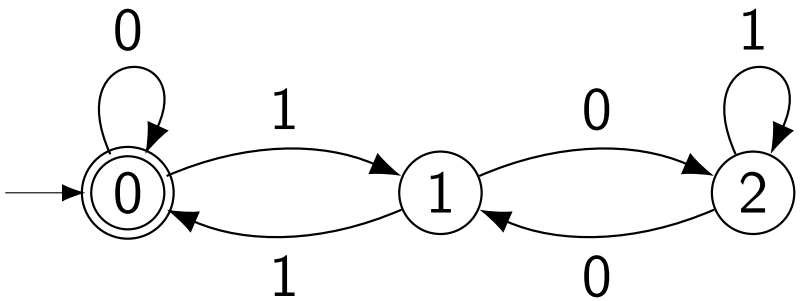
$$
\begin{array}{lll}
10101 & 2^4+2^2+2^0 & = 16+4+1 = 21 \\
101010 & 2^5+2^3+2^1 & = 32+8+2 = 42 = 2{\cdot}21 \\
101011 & 2^5+2^3+2^1+2^0 & = 32+8+2+1 = 43 = 2{\cdot}21+1
\end{array}
$$

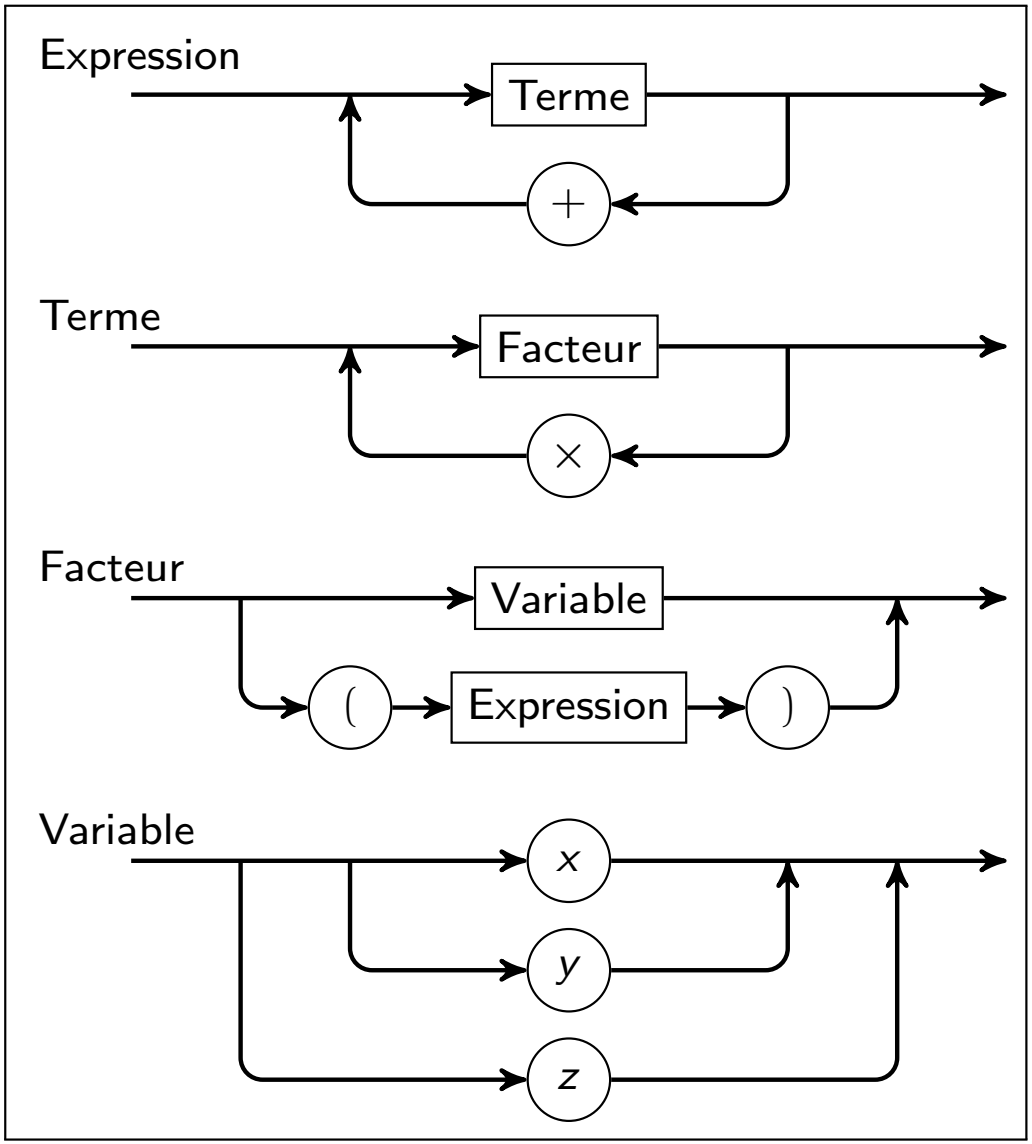
$val : \{0,1\}^* \longrightarrow \mathbb{N}$

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

states represent $val(w)$ modulo 3

δ	0	1
x	$2x$	$2x+1$
0	0	1
1	2	0
2	1	2





$$\Sigma = \{ x, y, z, (,), +, \times \}$$

$$x + x \times y \times (y + z) + x$$

$$E \to T + E \mid T$$
$$T \to F \times T \mid F$$
$$F \to V \mid (E)$$
$$V \to x \mid y \mid z$$

http://math.et.info.free.fr/TikZ/index.html

Chapitre 7

The formulas φ of propositional logic are given by the following *context free grammar*, in which p ranges over the propositional variables PVar.

$$\varphi ::= p \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid (\varphi)$$

1. *Basis.* Formules zijn $\bot$ (*false*), en de propositievariabelen p in PVar.
2. *Inductie.* φ_1 en φ_2 formules, dan ook $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \to \varphi_2$ en (φ_1).

$\Sigma = \text{Pvar} \cup \{ \bot, \wedge, \vee, \to, (,) \}$

$$S \to p \mid \bot \mid S \wedge S \mid S \vee S \mid S \to S \mid (S)$$

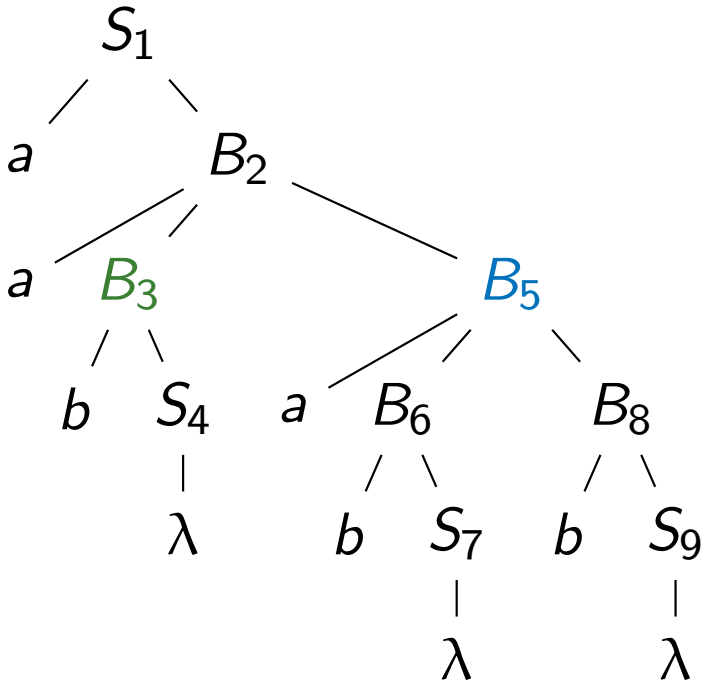
$AeqB = \{\, x \in \{a, b\}^* \mid n_a(x) = n_b(x) \,\}$ $\quad n_a(x) =$ aantal letters a in x

$aaabbb, ababab, aababb, \ldots$

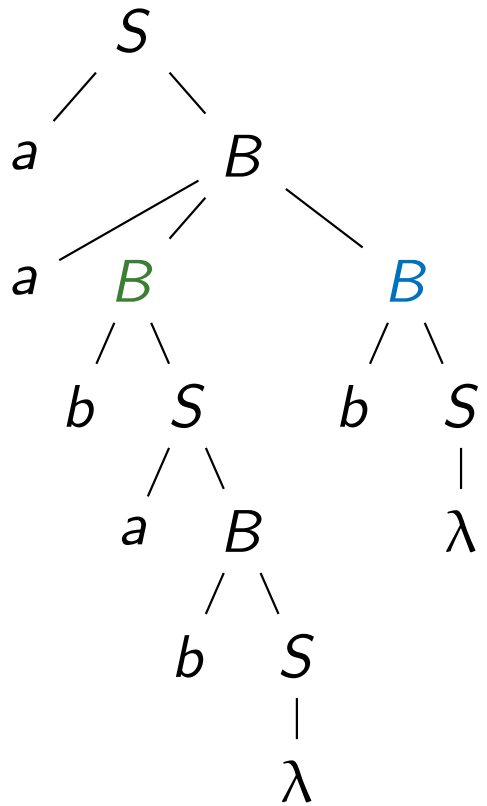
$$S \to \lambda \mid aB \mid bA$$
$$A \to aS \mid bAA \qquad A \text{ generates } n_a(x) = n_b(x) + 1$$
$$B \to bS \mid aBB \qquad B \text{ generates } n_a(x) + 1 = n_b(x)$$

$S \Rightarrow a\underline{B} \Rightarrow aa\underline{B}B \Rightarrow aab\underline{S}B \Rightarrow aab\underline{B} \Rightarrow aaba\underline{B}B \Rightarrow aabab\underline{S}B \Rightarrow aababb\underline{B} \Rightarrow aababbb\underline{S} \Rightarrow aababb$

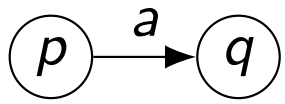
afleiding $\quad S \Rightarrow^* w \in \{a, b\}^*$



$S_1 \Rightarrow aB_2 \Rightarrow aaB_3B_5 \Rightarrow aabS_4B_5$
$\Rightarrow aabB_5 \Rightarrow aabaB_6B_8 \Rightarrow$
$aababS_7B_8 \Rightarrow aababB_8 \Rightarrow$
$aababbS_9 \Rightarrow aababb$



$S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabSB \Rightarrow$
$aababB \Rightarrow aababSB \Rightarrow aababB \Rightarrow$
$aababbS \Rightarrow aababb$

TYPE	grammar	automaton
3	regular	
	right-linear	finite state
	$A \to aB$	
2	context-free	
	$A \to \alpha$	pushdown
		(+lifo stack)
1	context-sensitive	
		linear bounded
	$\lvert\alpha\rvert \leqslant \lvert\beta\rvert$	
	monotone	
0	recursively enumerable	
	$\alpha \to \beta$	turing machine

Chomsky hierarchy*

type 3 Klasse van talen heet regulier, naar de operaties.

Eindige automaten. De rechtslineaire grammatica's doen de automaat precies na. Productie $A \to aB$ (transitie) en $A \to \lambda$ (eindtoestand).

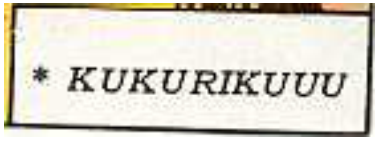
type 2 Klasse van talen heet context-vrij, naar de grammatica. Producties $A \to \alpha$, oftwel één symbool wordt herschreven, onafhankelijk van de buren, vandaar de naam.

De grammatica's definieren een recursief proces. De bijbehorende automaten hebben een stapel, waarmee dat kan worden nagedaan.

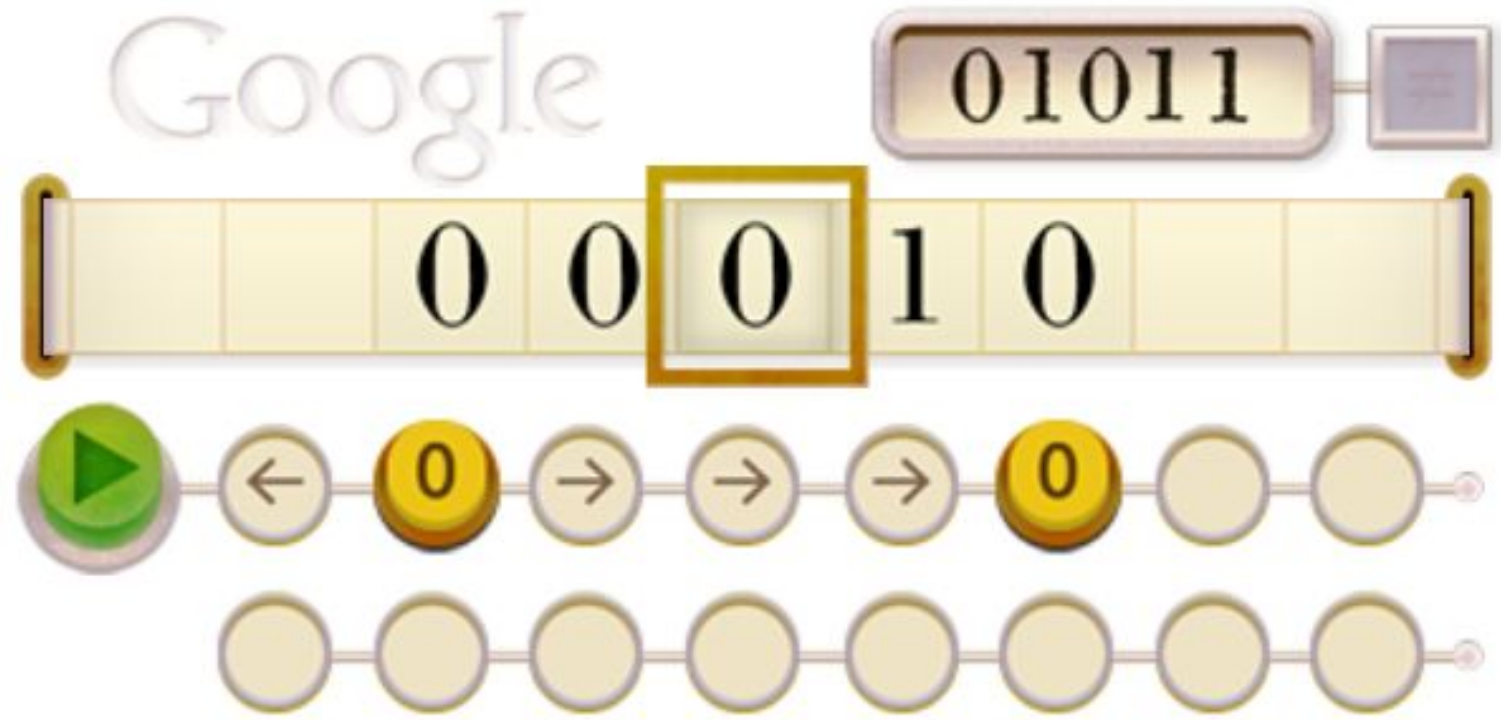
type 0 De talen heten recursief opsombaar: er is een algoritme dat de strings opsomt (genereert), en dat is dan de Turing machine. De grammatica's hebben geen beperking.

type 1 Talen heten context gevoelig, naar een grammatica type waar de $A \to \alpha$ regels alleen gebruikt kunnen worden tussen bepaalde buren. Een equivalent type grammatica heet monotoon, dat is de beperking dat nooit een kortere string mag worden verkregen met de regels.

De automaat is lineair begrensd, dat wil zeggen een Turing machine die slechts evenveel tape mag gebruiken als waar zijn invoer staat.



23 juni 2012 100e geboortedag van Alan Turing



doodle: Alan Turings 100th birthday

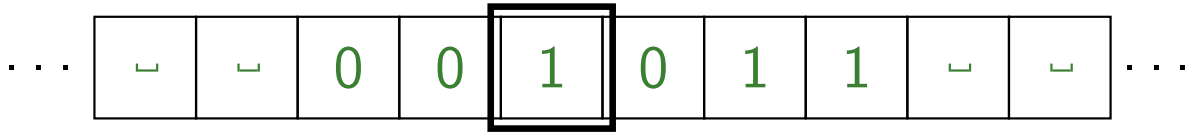
- berekenbaarheid *computability*

  wat kan er berekend worden?
  onbeslisbaar

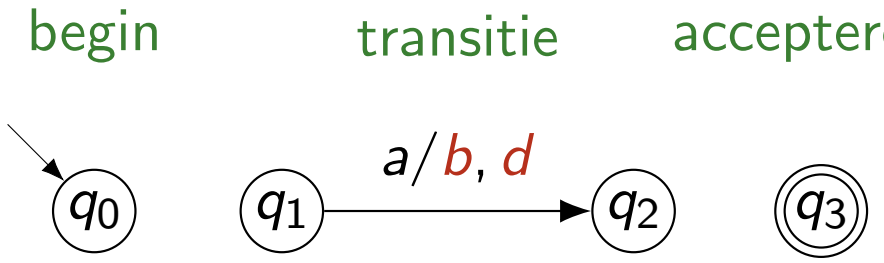
- complexiteit

  hoe *efficient* kan iets berekend worden?

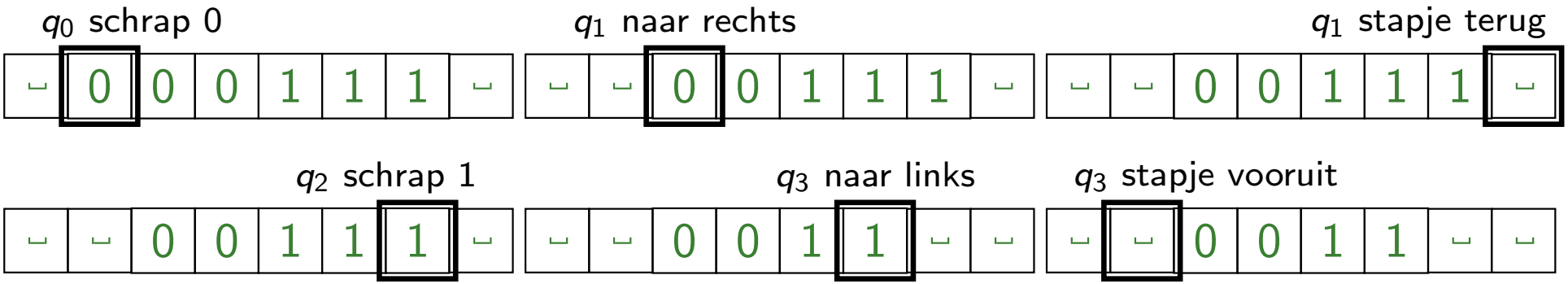
  tijd- en ruimtecomplexiteit
  P versus NP
  NP-compleet



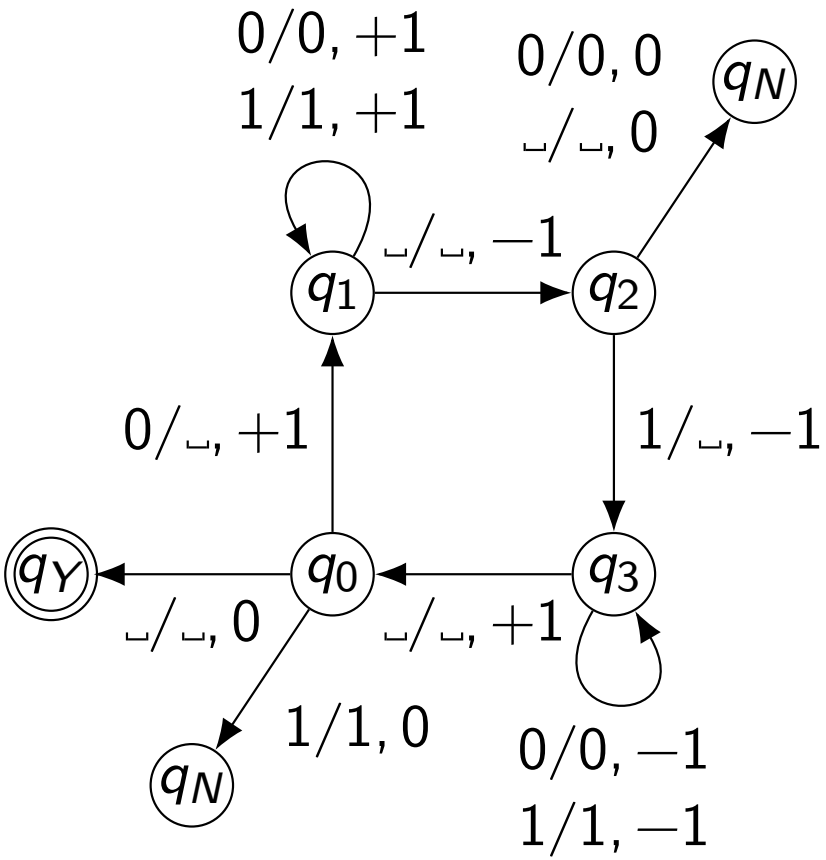
– lezen en schrijven a/b
– twee richtingen $d \in \{0, \pm 1\}$
– onbegrensde tape



- $\{ 0^n 1^n \mid n \in \mathbb{N} \}$



Turing machine $\{0^n 1^n \mid n \geqslant 0\}$



q_0 op eerste symbool links, schrap 0

q_1 ga naar rechts, tot ␣

q_2 op laatste symbool rechts, schrap 1

q_3 ga naar links, tot ␣

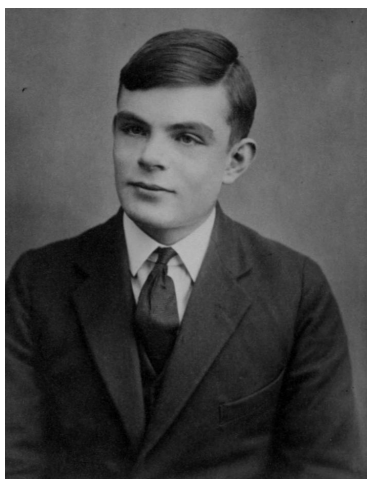
	0	1	␣
q_0	$q_1, ␣, +1$	$q_N, 1, 0$	$q_Y, ␣, 0$
q_1	$q_1, 0, +1$	$q_1, 1, +1$	$q_2, ␣, -1$
q_2	$q_N, 0, 0$	$q_3, ␣, -1$	$q_N, ␣, 0$
q_3	$q_3, 0, -1$	$q_3, 1, -1$	$q_0, ␣, +1$

deterministisch: functie

$$\delta : \underbrace{Q \times \{0, 1, ␣\}}_{\text{in toestand, lees symbool}} \mapsto \underbrace{Q \times \{0, 1, ␣\} \times \{-1, 0, +1\}}_{\text{nieuwe toestand, schrijf symbool, richting}}$$

searching for simple and powerful models

- Alan Turing (1912–1954)
  "On Computable Numbers, with an Application to the Entscheidungsproblem" (1937)
- Emil Post (1897–1954)
  "Finite Combinatory Processes - Formulation 1" (1936)
- Marvin Minsky (1927–2016)
  counter machine / register machine (1961)



pictures from turingarchive.org, Wikipedia, Bcjordan CC BY 3.0

Er bestaat *geen* algoritme voor

Entscheidungsproblem 1935/6 Alonzo Church + Stephen Kleene

is a given statement φ true ?

Halting problem 1936 Alan Turing

does a given program P halt on input x ?

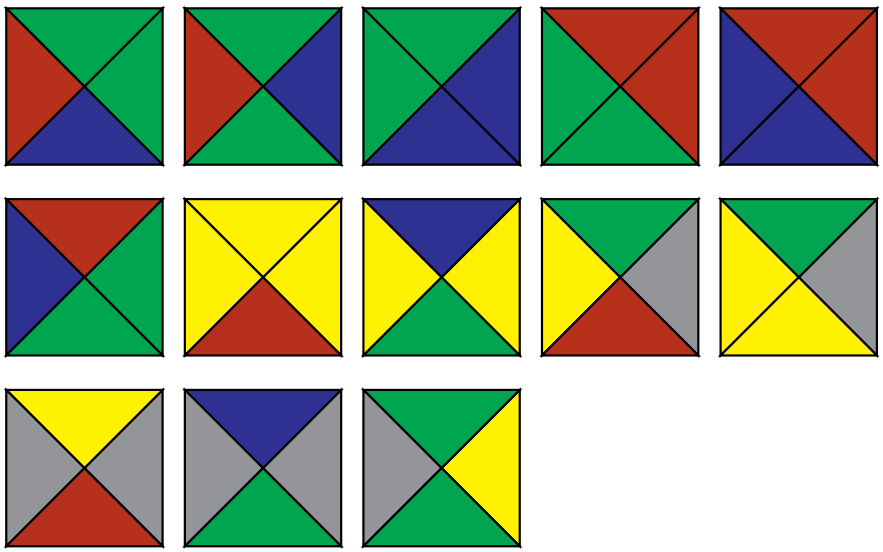
Hilbert's tenth problem 1970 Matiyasevich + Davis, Putnam, Robinson

has a given Diophantine equation D a solution ?

$$61x^2 + 1 = y^2$$

Wang tiles, 1961

niet draaien



patroon zonder regelmaat (lastig)

Karel Culik II, 1996

invoer: verzameling tegels

gevraagd: bestaat er een passende betegeling van het vlak?

er is géén algoritme dat dit probleem oplost

Berger 1966

probleem $\sim$ taal

graaf G $\rightsquigarrow$ $\langle G \rangle$ string

Ham alle Hamilton grafen $\rightsquigarrow$ $\langle \text{Ham} \rangle$ taal

G is Hamilton graaf $\iff$ $\langle G \rangle \in \langle \text{Ham} \rangle$

klasse van talen / problemen

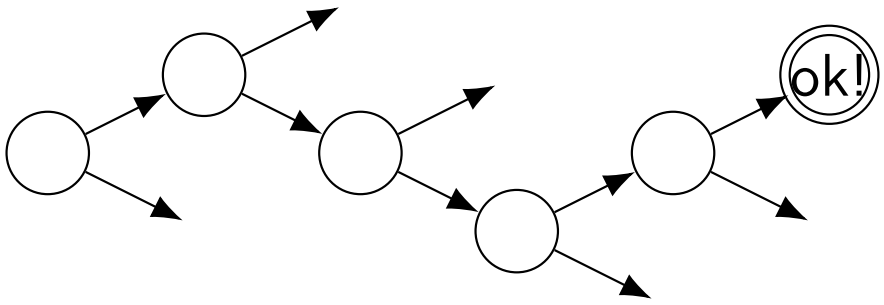
P geaccepteerd door *deterministische* TM in *polynomiale* tijd

Ham circuit berekenen in polynomiale tijd

NP geaccepteerd door *niet-deterministische* TM in *polynomiale* tijd

Ham circuit gokken en verifieren in polynomiale tijd

$\simeq$ gegeven Ham circuit verifieren in polynomiale tijd


ok!

P vs NP

P geaccepteerd door *deterministische* TM in *polynomiale* tijd

oplossing berekenen in polynomiale tijd

NP geaccepteerd door *niet-deterministische* TM in *polynomiale* tijd

oplossing (gokken en) verifieren in polynomiale tijd

The P versus NP problem is a major unsolved problem in computer science. It asks whether every problem whose solution can be quickly verified can also be solved quickly.

wikipedia

3-SAT satisfiability

gegeven: Boolese formule φ in 3CNF
vraag: kan φ waargemaakt worden?

clause literal

$$\varphi = (p \vee q \vee r) \wedge (p \vee \neg q \vee s) \wedge (\neg p \vee \neg r \vee \neg s) \wedge (\neg p \vee r \vee s)$$

3-COL 3-colouring

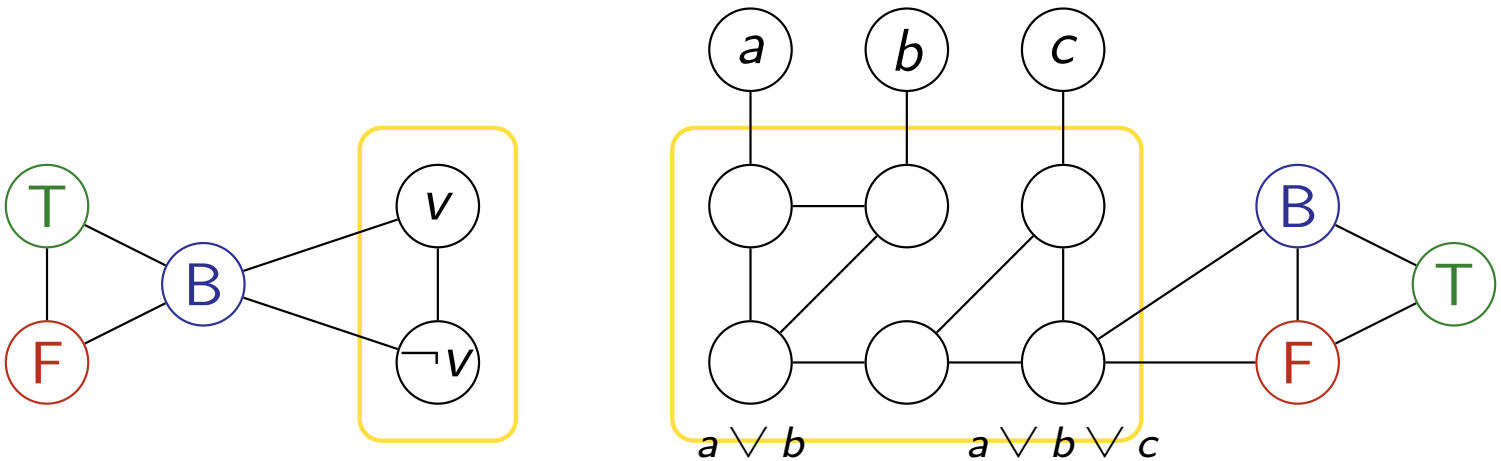
gegeven: ongerichte graaf G
vraag: kan G gekleurd worden met 3 kleuren?

reductie algoritme vertalen

$$3\text{-SAT} \leqslant_p 3\text{-COL}$$

formule φ $\xrightarrow{R}$ G_φ ongerichte graaf

φ waargemaakt desda G_φ 3-kleurbaar



elke variabele v elke clause $a \vee b \vee c$

David Epstein wikipedia:gadget

SAT is één van de moeilijkste problemen in **NP**

Stelling (Stephen Cook, 1971; Leonid Levin, 1973)

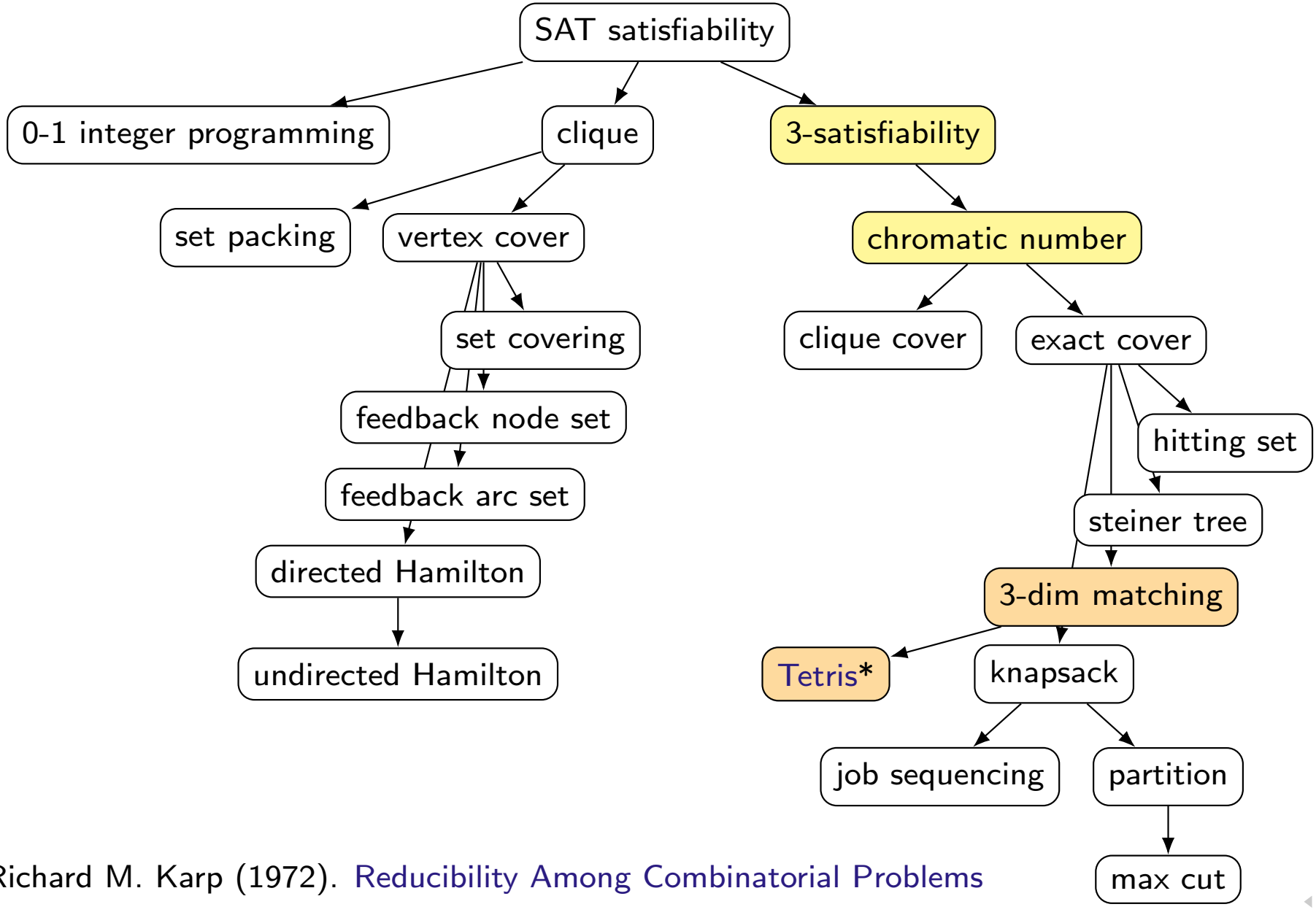
SAT $\in$ **NPC** $\quad$ P $\leqslant_p$ SAT voor alle P in **NP**

Jiří Janíček wikipedia

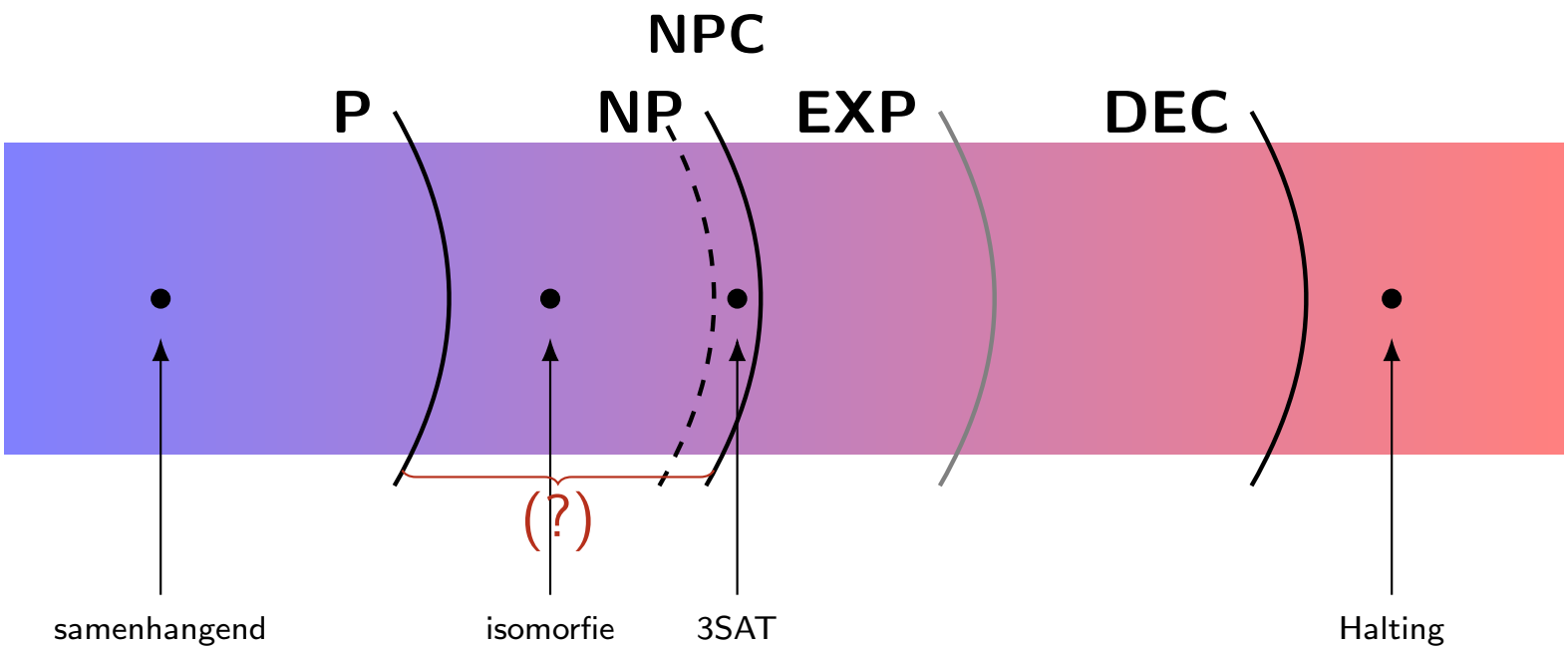
Sergio01 wikipedia

Stephen Cook, The complexity of theorem proving procedures. Proceedings of the Third Annual ACM Symposium on Theory of Computing. pp. 151–158, 1971. doi:10.1145/800157.805047

Karp's 21 NP-complete problems



Richard M. Karp (1972). Reducibility Among Combinatorial Problems


NPC
P
NP
EXP
DEC
(?)
samenhangend
isomorfie
3SAT
Halting

END.

edit 6 december 2020