Inhoudsopgave

1. Patronen

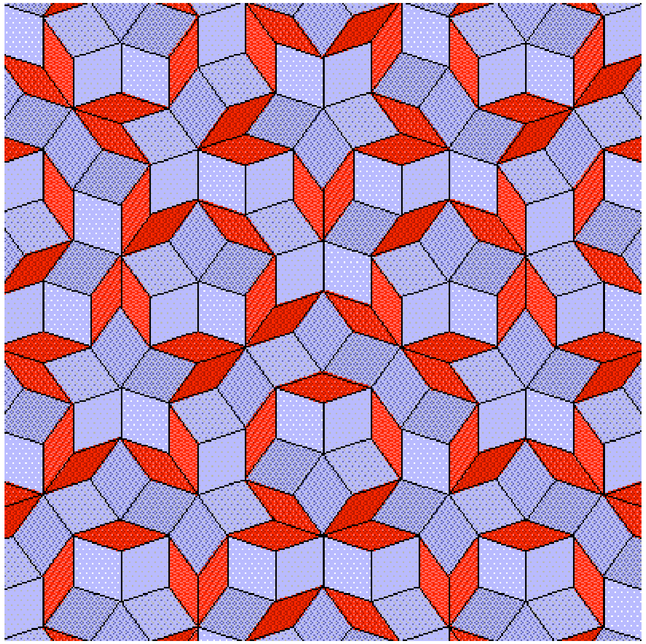
Natuurlijke talen vertonen een rijke verscheidenheid aan vormpatronen, op verschillende niveau's van analyse (lettergreep- en woordstructuur, woordgroepen, ...).

- Welke instrumenten staan ons ter beschikking om met eindige middelen oneindige verzamelingen van uitdrukkingen te karakteriseren die aan bepaalde patroonkenmerken voldoen?
- Zijn er verschillen aan te wijzen in de complexiteit van de patronen die we aantreffen, en hoe kunnen we die complexiteit meten?
- Welke (mentale) rekenvermogens zijn er nodig om taalpatronen te verwerken?

FORMELE GRAMMATICA'S $\rightsquigarrow$ patroonrecepten

AUTOMATEN $\rightsquigarrow$ de bijhorende rekenmodellen

2. Vergelijk: tegelpatronen



- Simpele bouwstenen, combinatieregels; oneindig aantal realiseringen
- Links: ‘tumbling blocks’, periodisch (cf behangpapier)
- Rechts: Penrose tegels, a-periodisch

3. Regulier versus context-vrij

We vergelijken twee soorten patronen:

- Lettergreepstructuur
- Spiegelwoorden (palindromen)

Elk van beide heeft een eigen grammatica en verwerkingsmodel:

- Lettergreepstructuur: reguliere expressies, eindige automaten
- Spiegelwoorden: contextvrije grammatica's, stapelautomaten

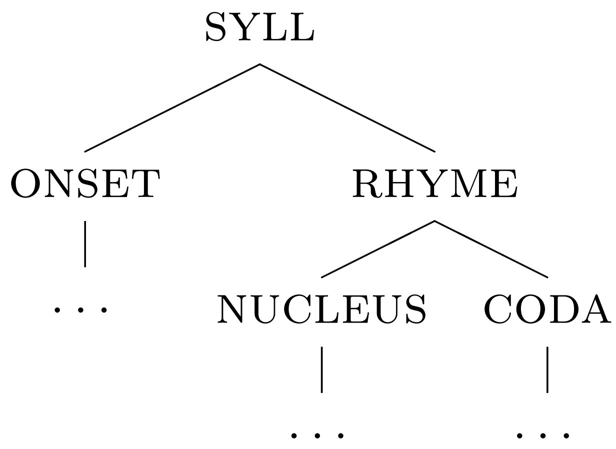
4. Lettergrepen: taal met één hand

Whales for the Welsh is een vreemd boek. Er komt niet één lang woord in voor. Elk woord in dat boek is kort. Als men het leest weet men eerst niet wat het is, en dan moet men vaak ha! ha! doen, want het staat erg raar, een heel boek met elk woord zo kort. Vaak is het net of een kind het zegt en het geeft ook een soort toon van spot; het viel mij op dat het soms lijkt op de stijl van Piet Grijs. Ik weet niet goed hoe dat komt. ...

RUDY KOUSBROEK, *De logologische ruimte*, pp. 118 e.v.

- 1-lettergrepige bespreking van 1-lettergrepig boek (*Whales for the Welsh. A Tale of War and Peace, with Notes for those who Teach or Preach*).
- Wat is het bouwschema voor een (Nederlandse) lettergreep?

5. Bouwplan voor lettergrepen



- onset: nul of meer medeklinkers (maar: **ls*)
- nucleus: klinkerkern (maar: **iaa*)
- coda: nul of meer medeklinkers (maar: **sl*)

6. Taalspel met lettergreepstructuur

Taalspel: venster op de mentale realiteit van onze kennis van het bouwplan:

- Alliteratie (*fliereﬂuiter*), rijm (*holderdebolder*)
- p-taal: vervang onset-nucleus door onset-nucleus-p-nucleus

  Kupunnepen jupullipie dipit veperstapaan?

- le Verlan (Frans): volgorde van lettergrepen omkeren

  bagnole (auto) $\rightsquigarrow$ *gnolba*,
  parents (ouders) $\rightsquigarrow$ *rempa*,
  baraque (huis) $\rightsquigarrow$ *raqueba*,
  flic $\rightsquigarrow$ *keuf* $\rightsquigarrow$ *feuk* (!)

7. Spiegelwoorden

m e e t s y s t e e m

3 kak • kok • lel

4 dood • effe • kook • paap

5 kajak • madam • negen • rotor • saga's

6 nekken • nijppijn • pikkip • redder • rottor • serres

7 amokoma • kalklak • karwrak • kortrok • kutstuk • modedom • nepapen

10 moorddroom • regelleger • topkookpot

11 levensnevel • lippepeppil • meetsysteem

12 koortsstrook • parterretrap

14 deelstaatsleed • partyboobytrap

Battus, *Opperlandse Letterkunde*, pp. 67 e.v.

8. Alfabet

Een alfabet is een eindige verzameling symbolen. Notatie: Σ.

Voorbeelden:

- $\Sigma_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. De tien-elements verzameling van de decimale cijfers.
- $\Sigma_2 = \{a, b, c, \ldots, x, y, z\}$. De 26-elements verzameling van alle kleine letters van het Nederlands.

Een niet-voorbeeld:

- $\mathbb{N} = \{0, 1, 2, \ldots\}$. De verzameling van alle natuurlijke getallen is geen alfabet, want deze verzameling is oneindig.

9. Tekenrijtjes over een alfabet

Een tekenrijtje van lengte $n(\geq 0)$ over een alfabet Σ is een geordend n-tal van elementen van Σ, geschreven zonder leestekens.

- Voorbeeld: als $\Sigma = \{a, b, c\}$, dan zijn a, aa, ab, acc en $bbac$ tekenrijtjes over Σ, met respectieve lengtes $1, 2, 2, 3, 4$.

$\Sigma^* \stackrel{\text{def}}{=}$ de verzameling van alle tekenrijtjes over Σ met een eindige lengte.

Er is precies één tekenrijtje over Σ van lengte 0: het *lege* rijtje (notaties: ϵ, of `[]`).

We spreken af dat tekenrijtjes altijd eindig zijn (een eindige lengte hebben).

10. Talen over een alfabet

Een *taal* is een verzameling tekenrijtjes over een alfabet. Dus: een taal over alfabet Σ is een deelverzameling van Σ^*.

Voorbeelden:

- Een *eindige* taal: de verzameling `{aap,noot,mies}`; een taal die slechts drie rijtjes bevat.
- Een *oneindige* taal: de verzameling van alle tekenrijtjes over het alfabet `{a,b}` waar minstens één `a` in zit: `a,ab,bba,baabbb,` ...

Een niet-voorbeeld: de verzameling die bestaat uit het ene rijtje $0,142857142857 14\ldots$ (de decimale expansie van $\frac{1}{7}$). Dit rijtje is oneindig.

11. Reguliere talen

De eerste familie van talen die we bekijken zijn de *reguliere* talen. We kunnen ze op twee gelijkwaardige manieren karakteriseren:

- eindige automaten: het computationele model voor reguliere talen
- reguliere expressies: de specificatietaal voor reguliere patronen

12. Eindige automaten (finite state automata)

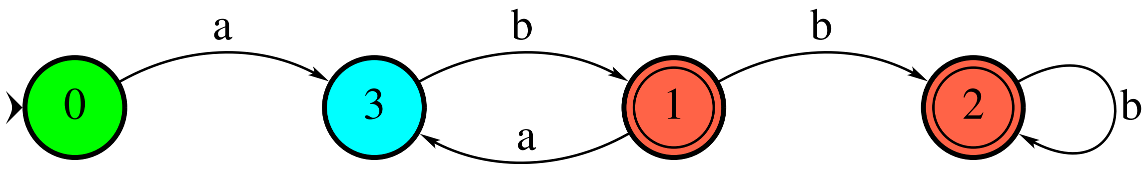
Een eindige automaat bestaat uit een vijftal componenten $(Q, \Sigma, \delta, q_0, F)$:

- Q: een eindige verzameling toestanden, waaronder
- q_0: de begintoestand, en
- $F \subseteq Q$: een verzameling eindtoestanden
- Σ: het alfabet
- δ: regels voor de zetten van de machine — $\delta(q, a) = q'$ betekent dat de machine, als hij in toestand q een symbool a leest, overgaat in toestand q'.

Een eindige automaat M accepteert een *rijtje* w als hij zich vanuit de begintoestand q_0 stap voor stap door het rijtje heen kan werken aan de hand van de overgangen δ, en daarmee in een eindtoestand terechtkomt.

Een automaat M herkent een *taal* L als L de verzameling rijtjes w is die door M geaccepteerd worden.

13. Een voorbeeld



Herkende rijtjes: $ab, abbb, abab, abababbbbb \ldots$ (een of meer herhalingen van het rijtje ab gevolgd door een willekeurig aantal b's).

Toestanden: q_0, q_1, q_2, q_3, start: q_0, eindtoestanden: q_1, q_2, alfabet: $\{a, b\}$. Zetten:

	0	1	2	3
a	3	3		
b		2	2	1

14. Reguliere expressies

Regulier expressies: uitdrukkingen die talen beschrijven.

EXPRESSIE	BETEKENIS
`{}`	de lege taal
`[]`	de taal die uitsluitend uit het lege rijtje bestaat
`a`	de taal die uitsluitend het rijtje `a` bevat, waar `a` een element van het alfabet is
`[A,B]`	*concatenatie* (opeenvolging) van rijtjes w, v, met w een rijtje van taal `A` en v een rijtje van taal `B`
`{A,B}`	*keuze* uit rijtjes w, v, met w een rijtje van taal `A` en v een rijtje van taal `B`
`A*`	*herhaling* van nul of meer rijtjes w uit de taal `A`

15. Handige afkortingen

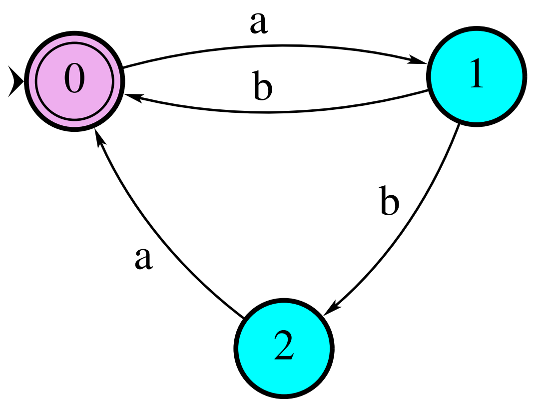
De reguliere operaties (opeenvolging, keuze, herhaling) maken het mogelijk nieuwe operaties als afkortingen in te voeren. Een paar voorbeelden (voor alfabet {`a,b,c`}):

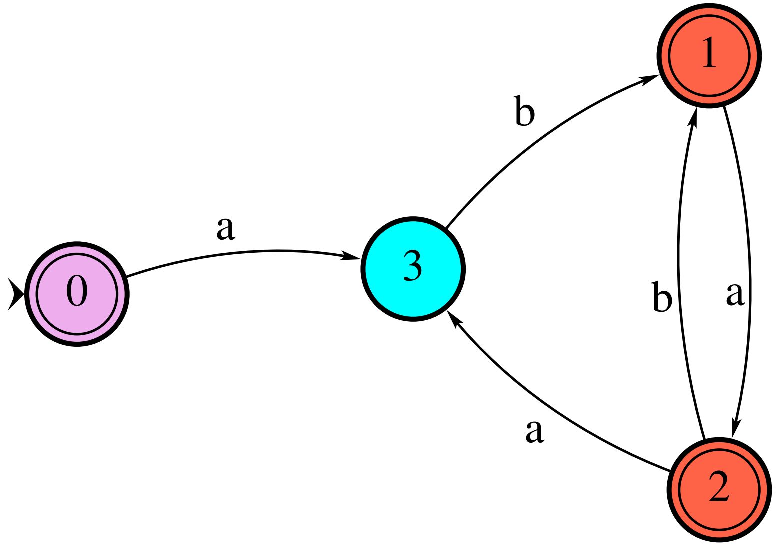
AFKO	DEFINITIE
`?`	`{a,b,c}` (keuze van een willekeurig alfabetsymbool)
`?*`	willekeurige rijtjes over het alfabet, de universele taal (Σ^*)
`A^`	`{[],A}` (misschien een rijtje uit taal `A`, optionaliteit)
`A-B`	verschil: verwijder alle rijtjes van taal `B` uit taal `A`
`A+`	`A* - []` of `[A,A*]` (minstens één rijtje uit taal `A`)
`$A`	`[?*,A,?*]` (rijtjes met een deelrijtje uit taal `A`)
`~A`	`?* - A` (complement van taal `A`)

Voorbeeld: `[[a,b]+,b*]` voor de automaat van §13.

16. Non-deterministische automaten

Vergelijk de volgende automaten voor het patroon `{[a,b],[a,b,a]}*`:

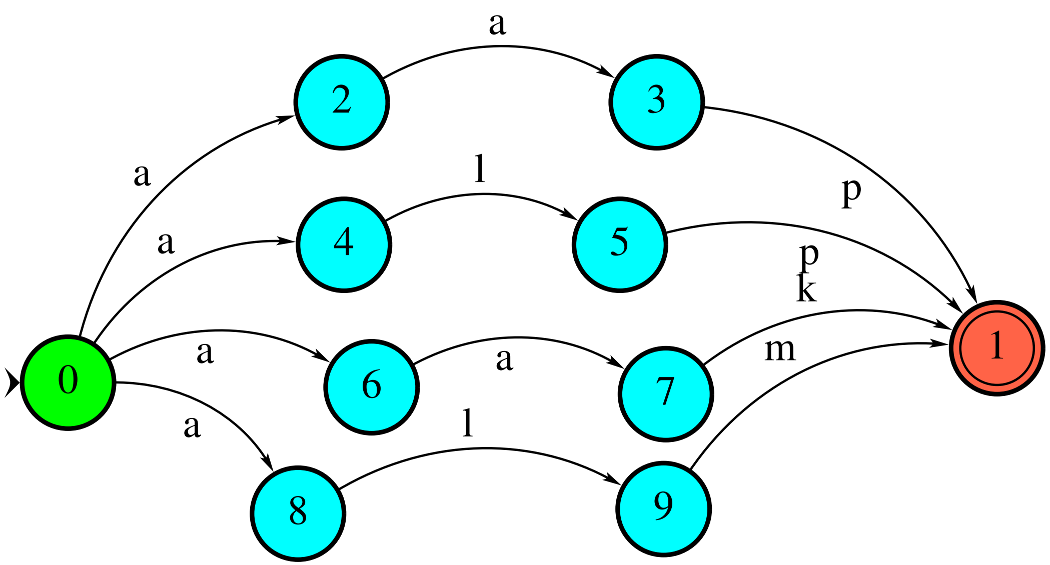


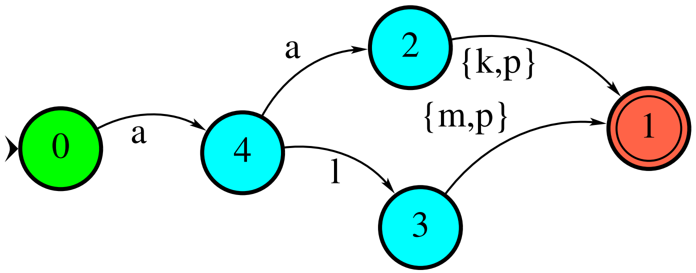


- Non-deterministisch (links): voor een gegeven toestand/invoersymbool is er *keuze* tussen verschillende overgangen.
- Voor elke non-deterministische machine kan je een deterministische bouwen die dezelfde taal herkent!

17. Minimale automaten

Vergelijk de volgende automaten. Ze herkennen allebei de taal {`aap`,`alp`,`aak`,`alm`}. Elke eindige automaat kan geoptimaliseerd tot een *minimale* machine.





18. Combinatorische eigenschappen

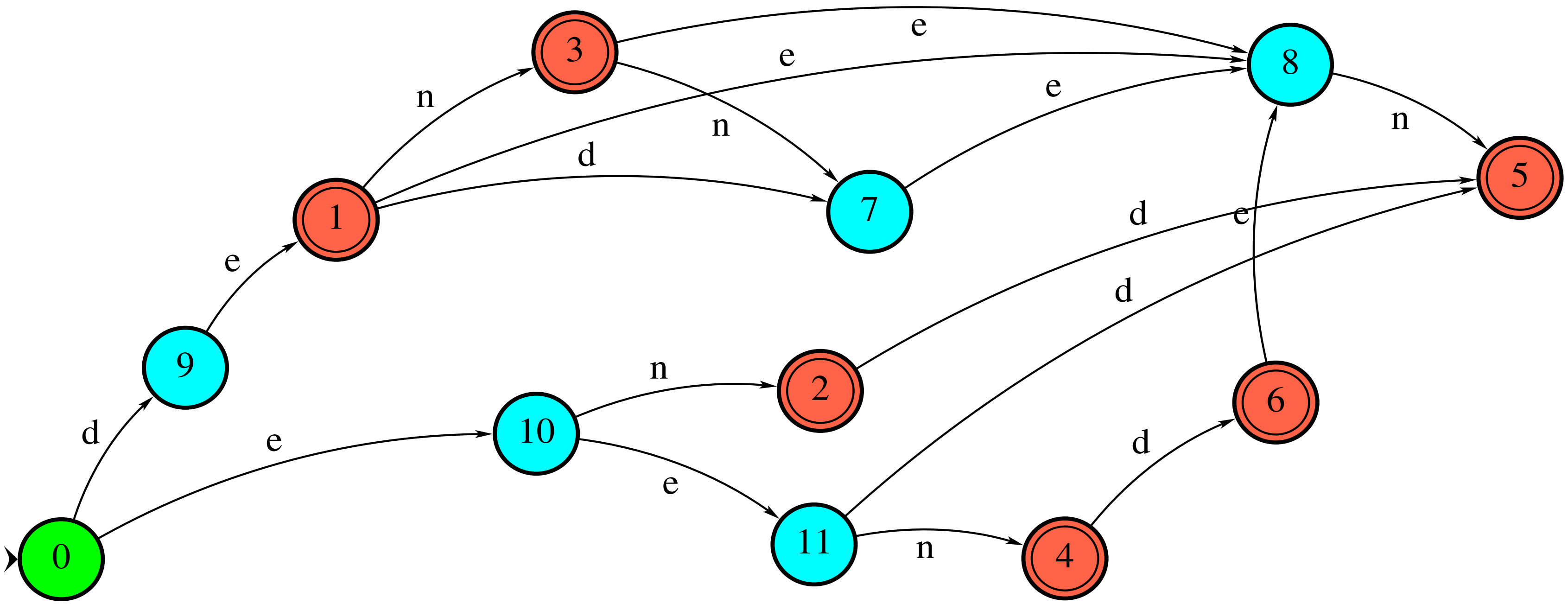
Reguliere patronen hebben rijke combinatorische eigenschappen: reguliere talen zijn gesloten onder de volgende bewerkingen:

- Vereniging
- Concatenatie
- Doorsnee
- Complementatie
- Iteratie (herhaling)

Wat verwerking betreft: determinisering en minimalisering garanderen optimaal gebruik van rekentijd en opslagruimte.

19. Lexicon als eindige automaat

Hieronder een stukje van het lexicon (woorden uit $\{\mathtt{e},\mathtt{n},\mathtt{d}\}^*$).



Mentaal lexicon als eindige automaat $\leadsto$ efficiënt zoeken!

20. Niet-reguliere patronen

Sommige patronen vereisen een krachtiger rekenvermogen dan wat een eindige automaat kan bieden. Er is gelukkig een test om vast te stellen of je met zo'n patroon te maken hebt.

De pomp-eigenschap van reguliere talen Laat L een oneindige reguliere taal zijn. Dan zijn er rijtjes x, y, z te vinden met y verschillend van het lege rijtje, zo dat $\texttt{[x,y}^n\texttt{,z]}$ in L zit voor elke $n \geq 0$.

Idee omdat L regulier is, is er een deterministische automaat die L herkent. Die automaat heeft een eindig aantal toestanden, zeg k. Omdat de taal oneindig is moeten er rijtjes in L zitten die lengte $> k$ hebben. Dus moet de automaat bij het herkennen van zo'n rijtje een toestand q meer dan één keer bereiken. Maar dan bevat de herkenningsprocedure een lus. Tijdens het doorlopen van die lus wordt een niet-leeg rijtje ingelezen. Noem dat rijtje `y`, en klaar!

21. Niet-reguliere patronen: voorbeelden

De taal L: `[a`^{n}`,b`^{n}`]` met $n \geq 0$ is niet regulier.

Gebruik de pompstelling om je hiervan te overtuigen. Neem een rijtje uit L, bijvoorbeeld *aaaabbbb*. We willen het rijtje opbreken in stukken x, y, z waarbij y herhaald kan worden. Elke keuze voert ons buiten L.

- y bestaat uitsluitend uit a's: herhaling geeft meer a's dan b's
- y bestaat uitsluitend uit b's: herhaling geeft meer b's dan a's
- y zit over de grens van ab: herhaling geeft b's voorafgaand aan a's.

Ander voorbeeld: spiegelwoorden!