Eindige automaten

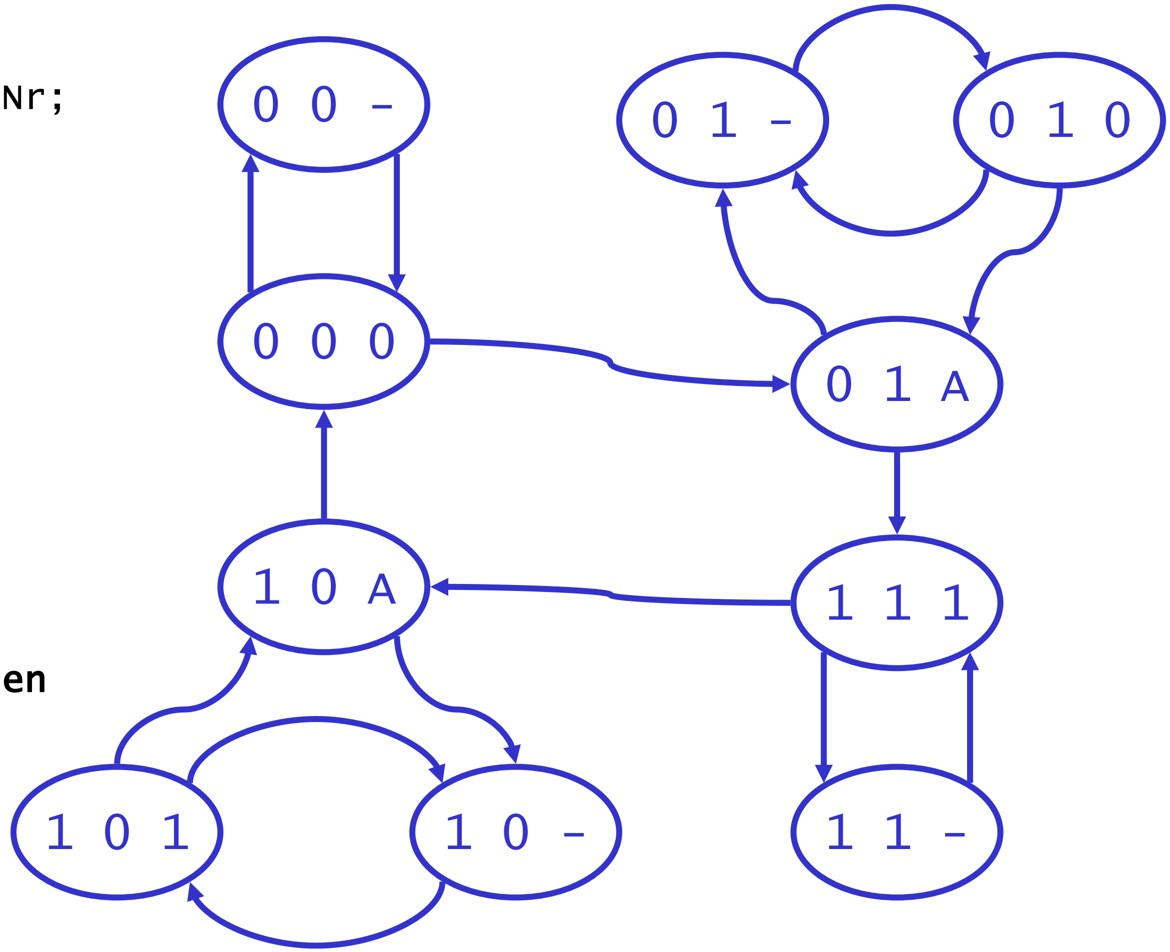
12

zie dictaat

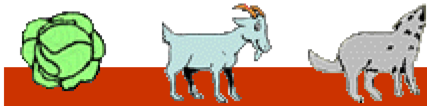
ch.12 Languages, Automata, Grammars

12.5 finite state automata

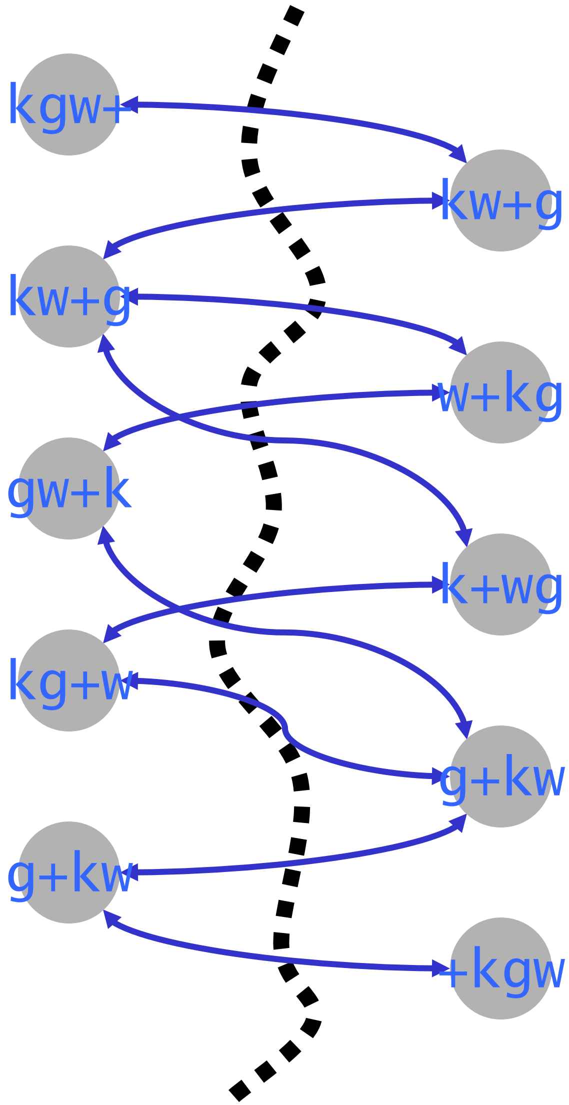
```
const MaxSeq = 1;
type EvType = (FrameArrival, CksumErr, TimeOut );

procedure sender3;
var NextFrameToSend : SequenceNr;
    s : frame;
    buffer : message;
    event : EvType;
begin
    NextFrameToSeend := 0;
    FromHost( buffer );
    repeat
    s.info := buffer;
    s.seq := NextFrameToSend;
    sendf( s );
    StartTimer( s.seq );
    wait( event );
    if event = FrameArrival then
    begin
      FromHost( buffer );
      inc( NextframeToSend );
    end
    until doomsday
end; { sender3 }
```

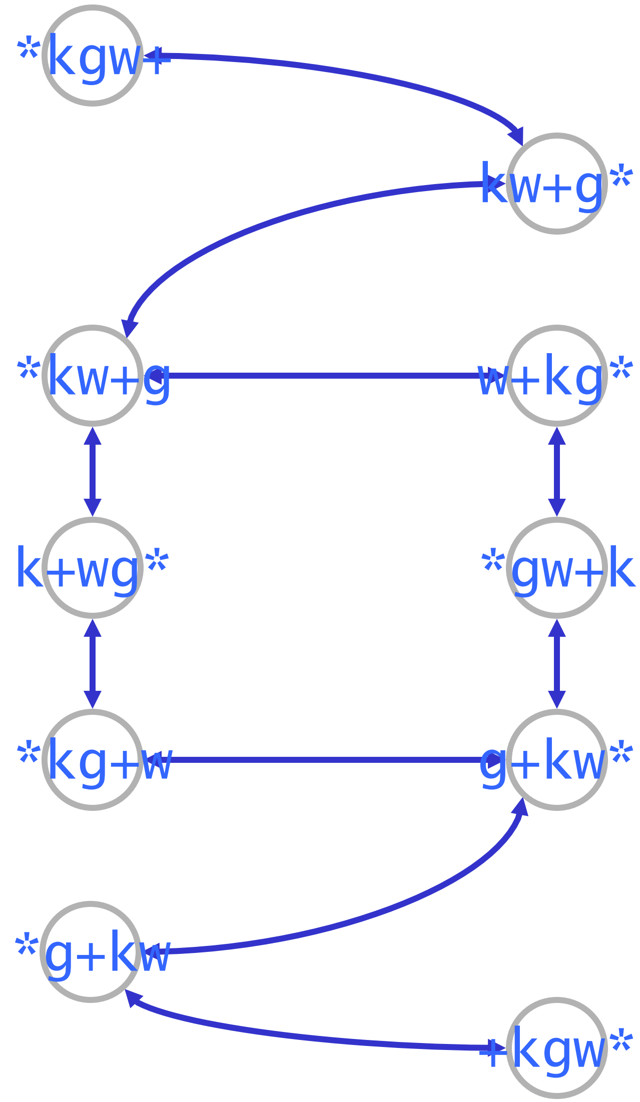


toestanden & acties



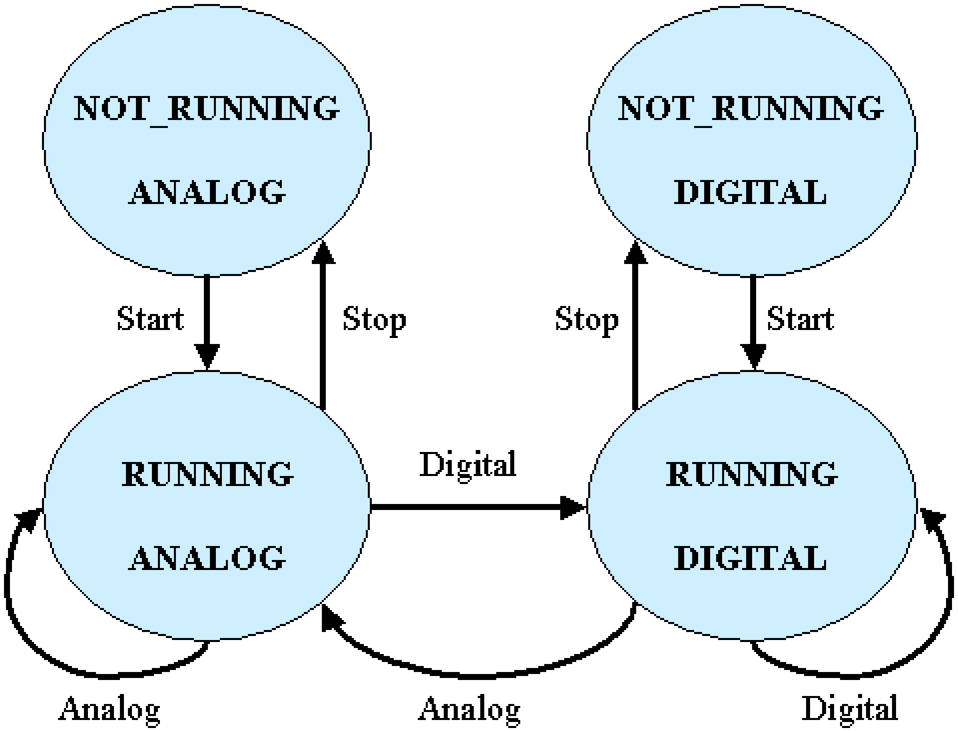
kool, geit, wolf

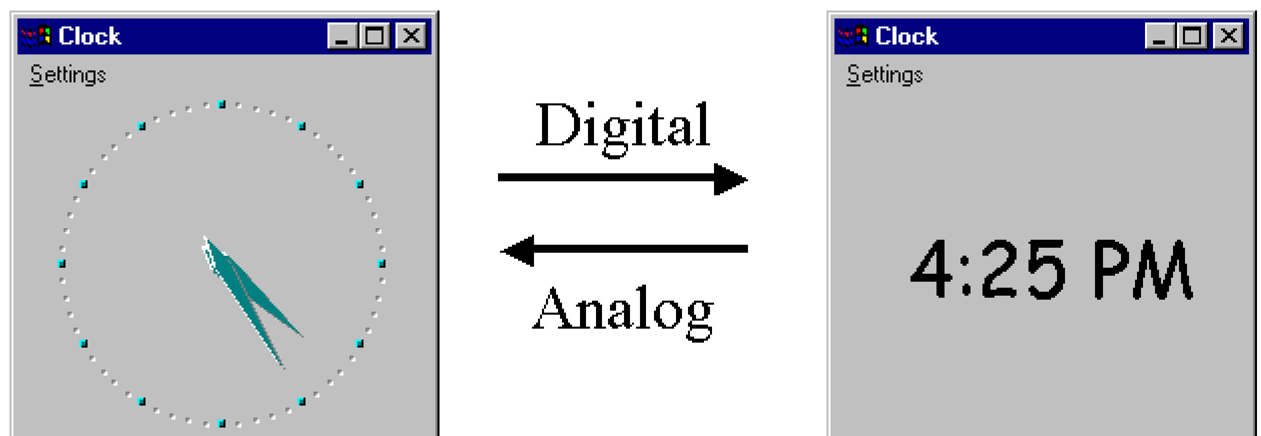




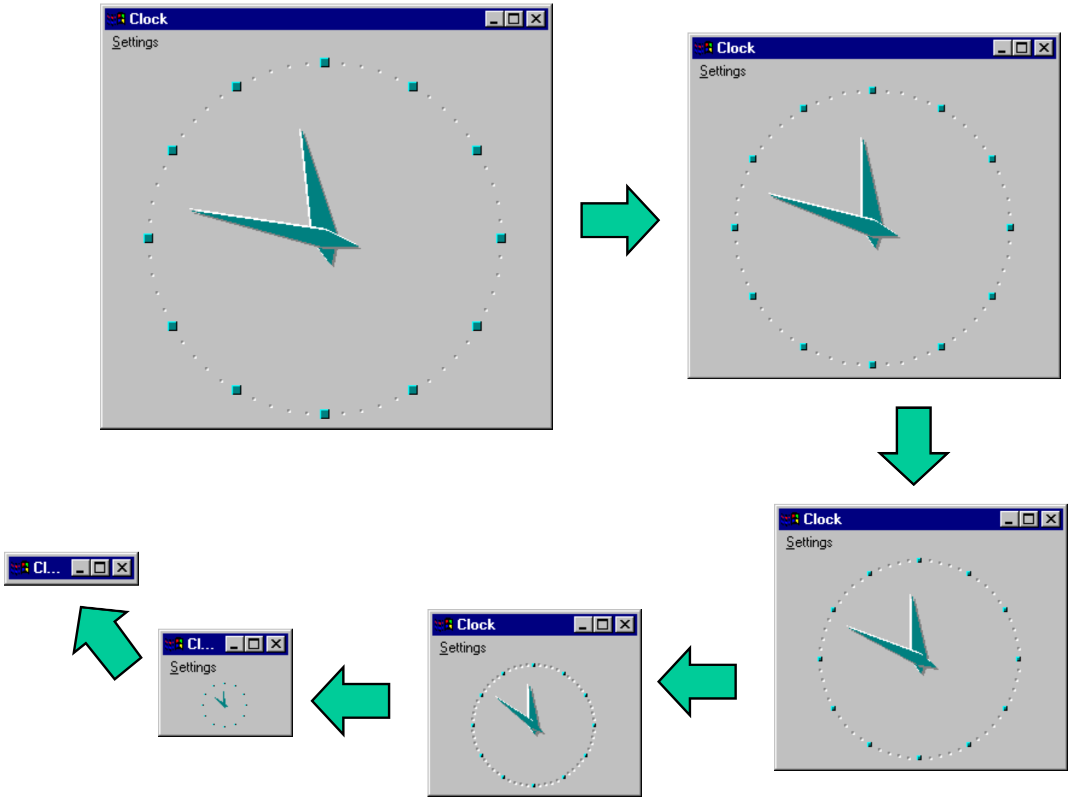
beschrijving

http://www.geocities.com/harry_robinson_testing/shoestring.doc





The Incredible Shrinking Clock Bug



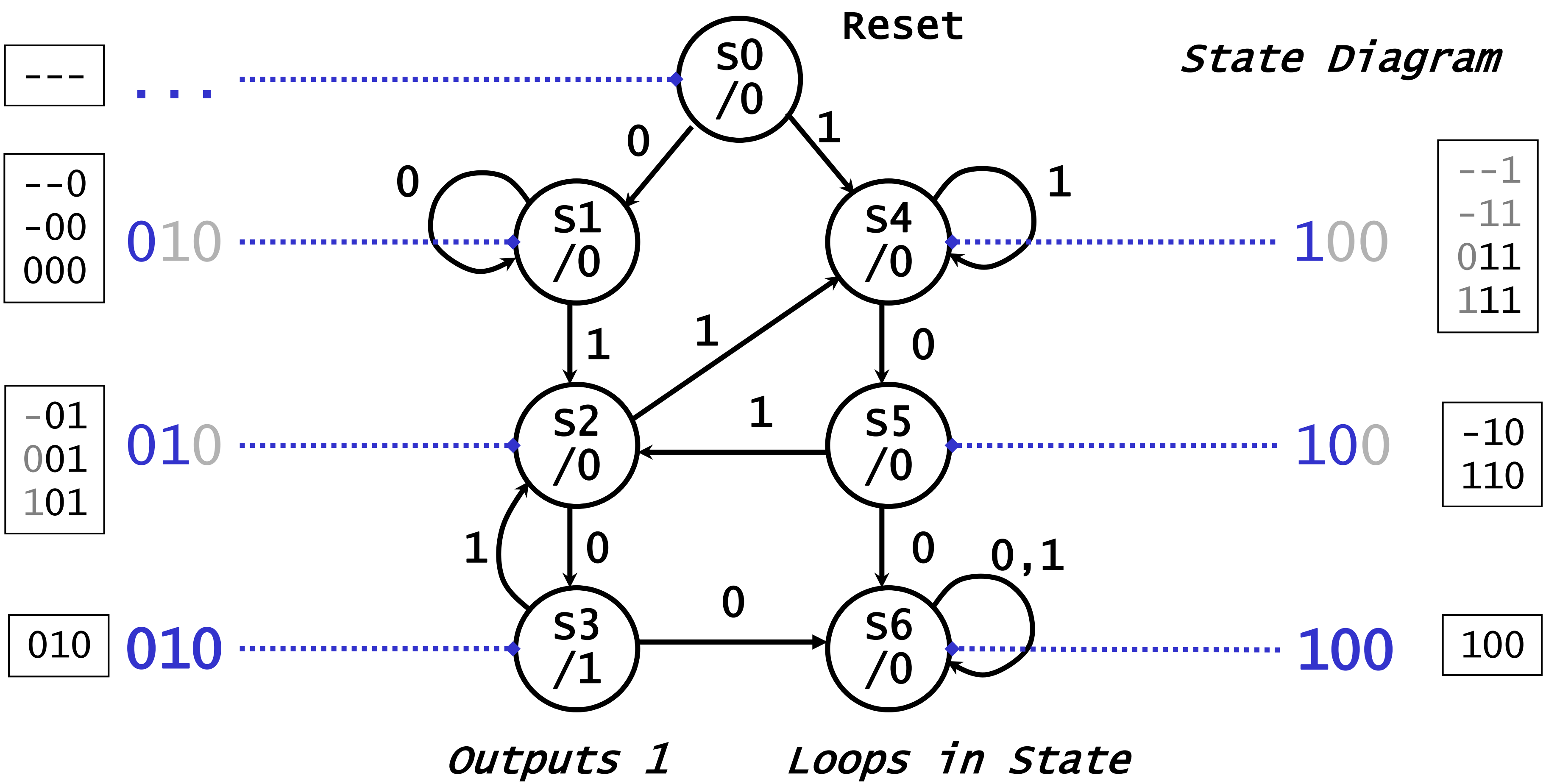
Start
Maximize
Stop
Start
Minimize
Stop
Start
Restore
Stop

… nog een web-voorbeeld

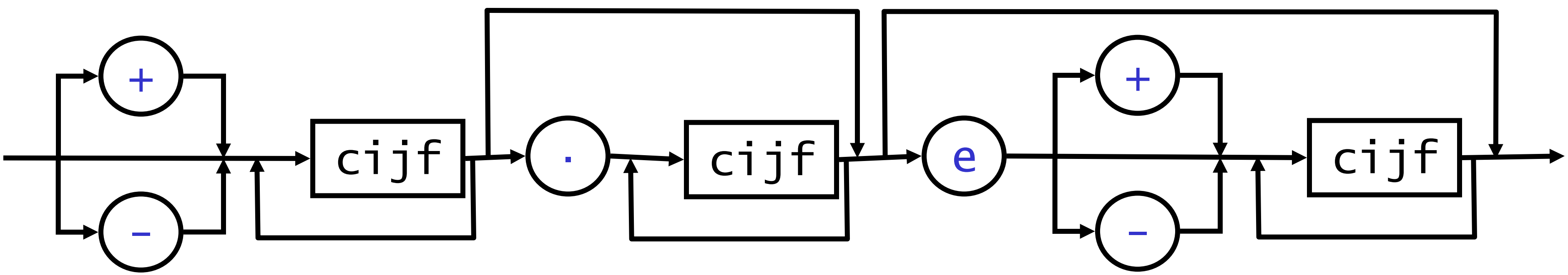
finite State Machine Word Problems

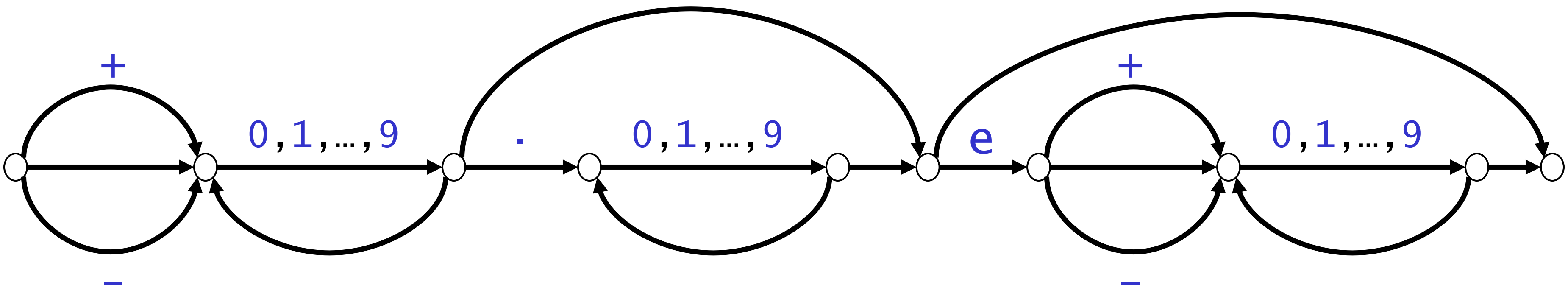
{ 010, 100 }

laatste
3 bits



getal





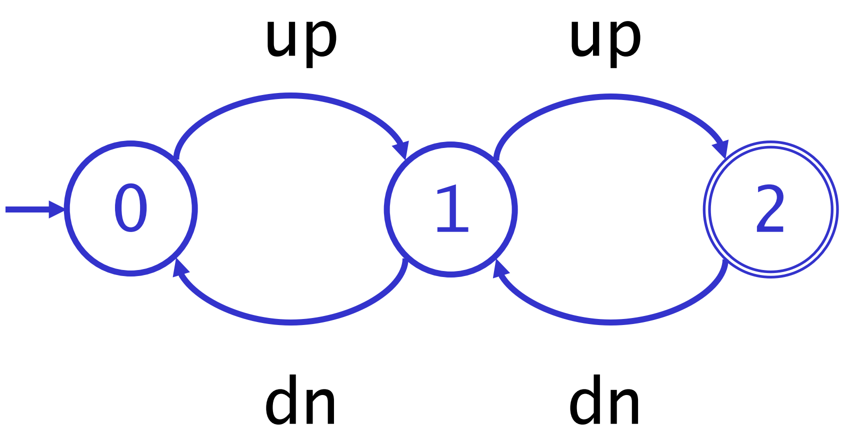
$$\{+,-,\lambda\} \cdot \{0,1,\ldots,9\}^+ \cdot (\{\lambda\} \cup \{.\}\cdot\{0,1,\ldots,9\}^+) \cdot (\{\lambda\} \cup \{e\}\cdot\{+,-,\lambda\}\cdot\{0,1,\ldots,9\}^+)$$

modelleren

- ontwerpen
- gedrag bestuderen
- fouten vinden
- correct bewijzen

gerichte graaf

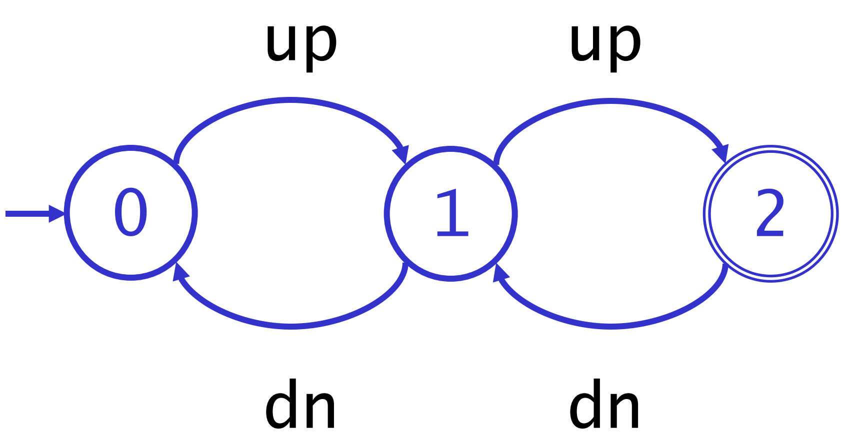
- knopen ~ toestanden
- pijlen met labels ~ acties
- begin- en eindtoestanden



definitie

een *eindige automaat* is een vijf-tupel $\mathcal{A} = (Q,\Sigma,E,q_{in},F)$ waarbij

- Q een eindige verzameling *toestanden*
- Σ een *alfabet*
- $E \subseteq Q\times\Sigma\times Q$ verzameling *takken*
- $q_{in} \in Q$ de *begintoestand*
- $F \subseteq Q$ de *eindtoestanden*



$Q = \{\ 0,1,2\ \}$

$\Sigma = \{\ \mathrm{up},\mathrm{dn}\ \}$

$E =$

$\{\ (0,\mathrm{up},1),\ (1,\mathrm{up},2),$

$(2,\mathrm{dn},1),\ (1,\mathrm{dn},0)\ \}$

$q_{in} = 0$

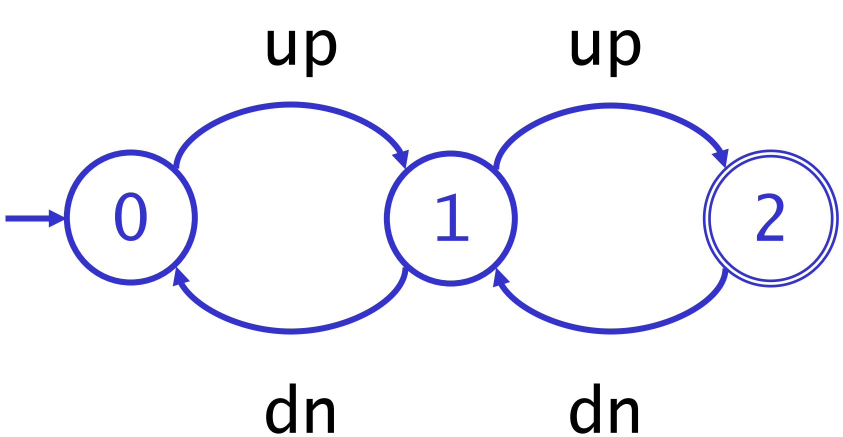
$F = \{\ 2\ \}$

$\mathcal{A} = (Q,\Sigma,E,q_{in},F)$

wandeling : afwisselend knopen (toestanden) en pijlen (takken)

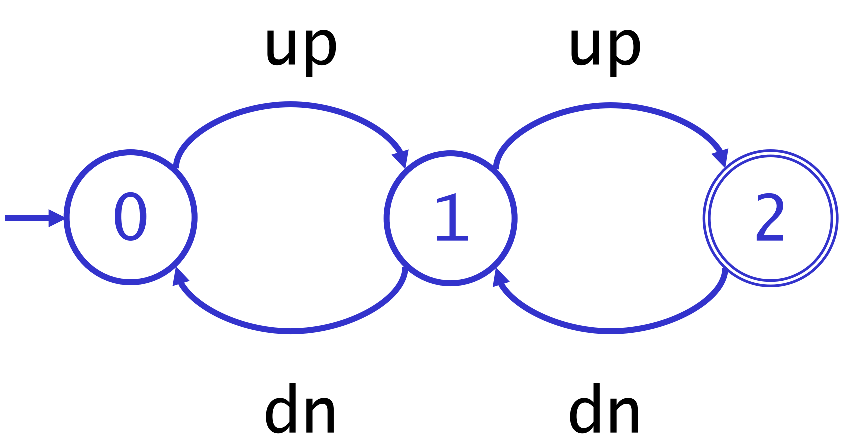
$p_0\ (p_0,a_0,p_1)\ p_1\ (p_1,a_2,p_2)\ \ldots\ p_{n-1}\ (p_{n-1},a_n,p_n)\ p_n$

met *label* $a_0 a_2 \ldots a_n$



$1 \overset{up}{\rightarrow} 2 \overset{dn}{\rightarrow} 1 \overset{dn}{\rightarrow} 0 \overset{up}{\rightarrow} 1$

$\mathcal{A} = (Q,\Sigma,E,q_{in},F)$ eindige automaat
de *taal* van $\mathcal{A}$, genoteerd $L(\mathcal{A})$, is
$\{\ x \in \Sigma^*$ | x is een label van een wandeling
van q_{in} naar een toestand in $F\ \}$

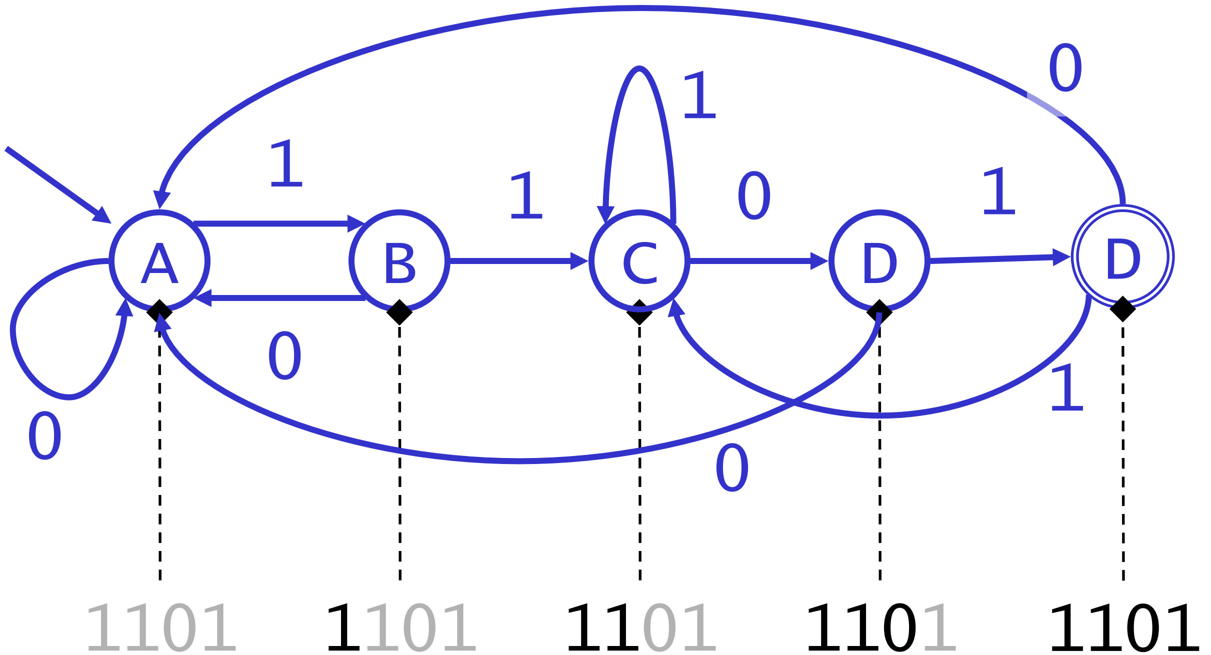


$\cancel{1 \xrightarrow{up} 2 \xrightarrow{dn} 1 \xrightarrow{dn} 0 \xrightarrow{up} 1}$

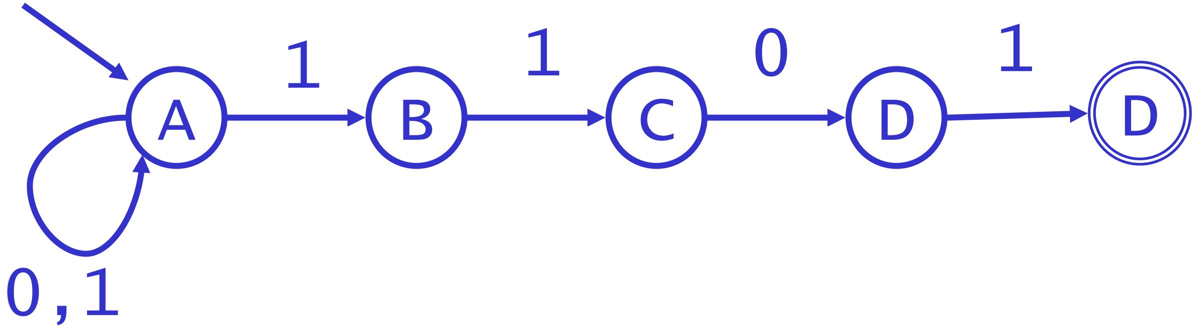
$0 \xrightarrow{up} 1 \xrightarrow{dn} 0 \xrightarrow{up} 1 \xrightarrow{up} 2$

$0 \xrightarrow{up} 1 \xrightarrow{dn} 0 \xrightarrow{dn} ? \xrightarrow{up} ?$

suffix 1101



algoritme

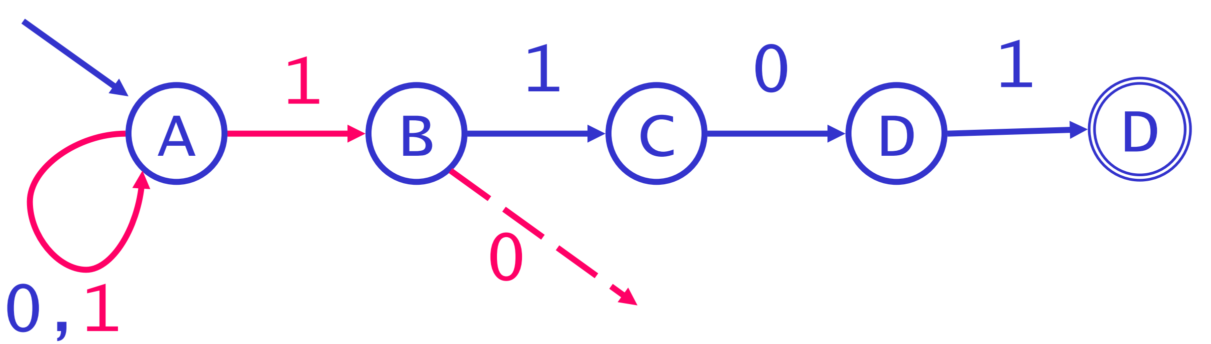


beschrijving

- 'on the fly' algoritme
- steeds van bewandeld prefix weten of dat zèlf tot de taal behoort
- geen keuze

definitie

een eindige automaat $\mathcal{A} = (Q,\Sigma,E,q_{in},F)$ heet *deterministisch* als voor elke toestand $p \in Q$ en elke letter $a \in \Sigma$ er precies één tak $(p,a,q) \in E$ is.



enige voorbeelden

talen over $\{0,1\}$

1) aantal 1-en is een drievoud

2) $\{\ 01,\ 011,\ 0111\ \}^*$

tentamen fi2

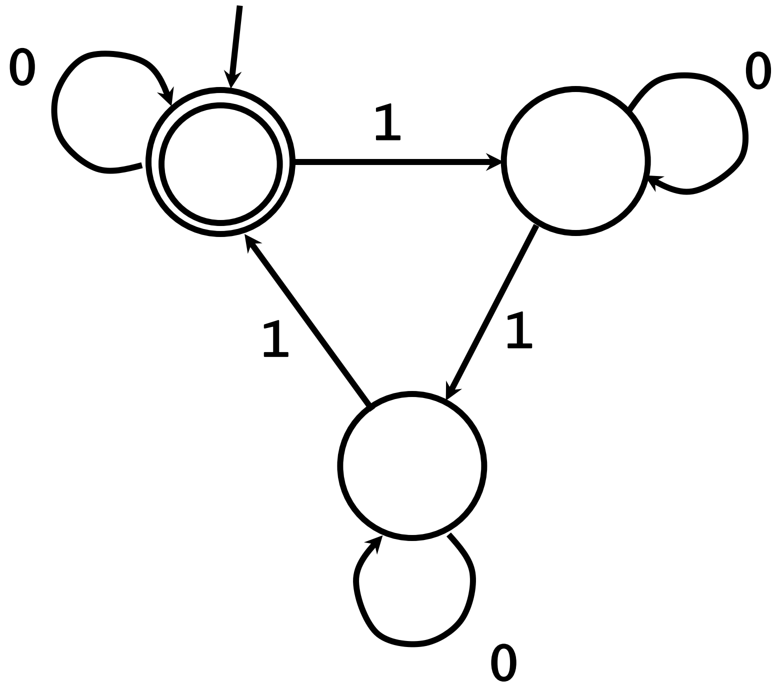
$x \in \{0,1\}^*$ $\quad \mathrm{val}(x) \in \mathbb{N}$

waarde als binair getal

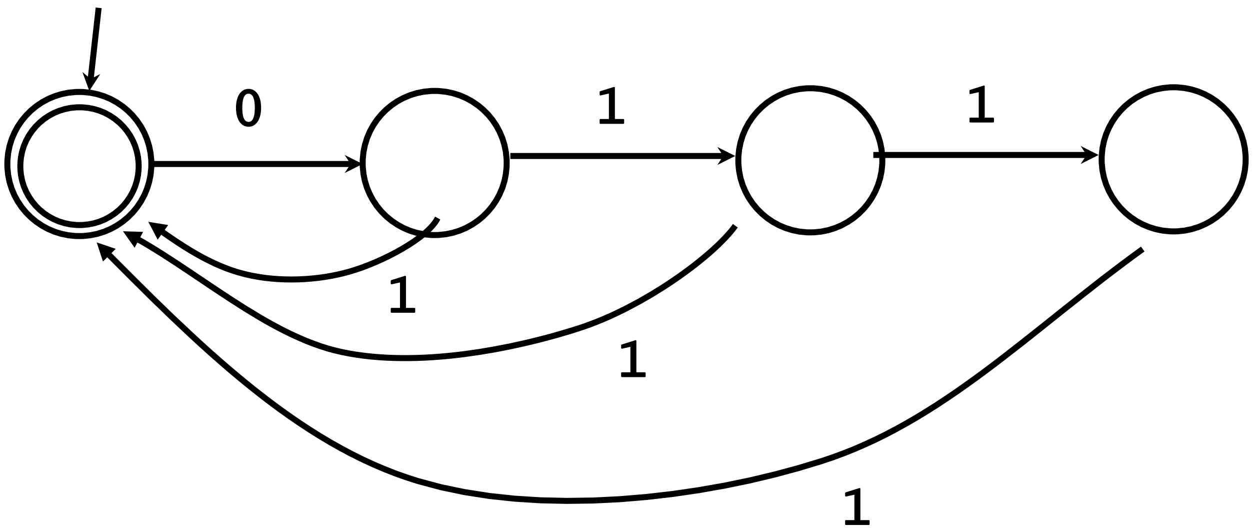
3) val(x) is een drievoud

zie dictaat fi1

aantal 1-en is een drievoud

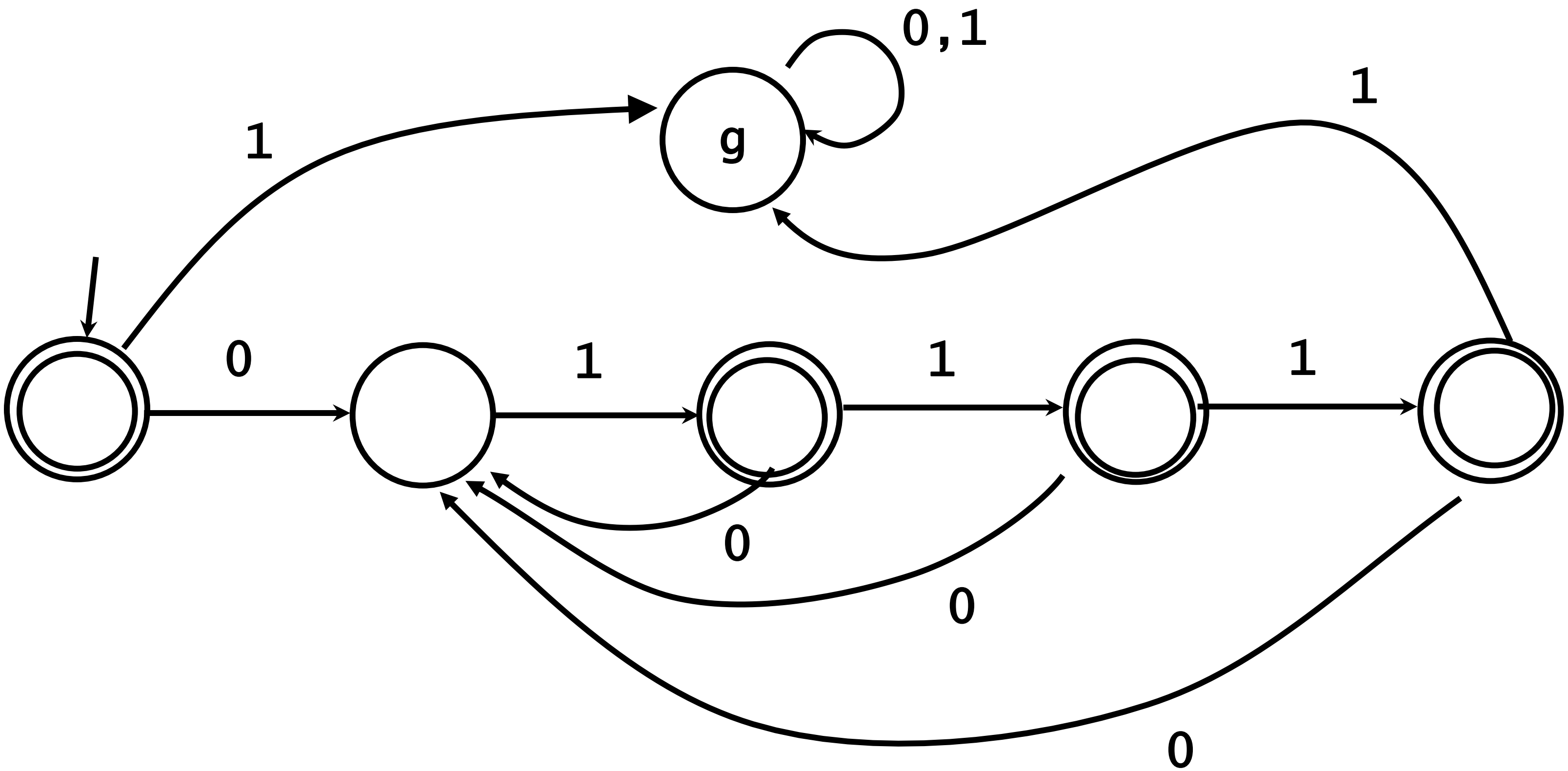


{ 01, 011, 0111 }*



niet deterministisch

{ 01, 011, 0111 }*

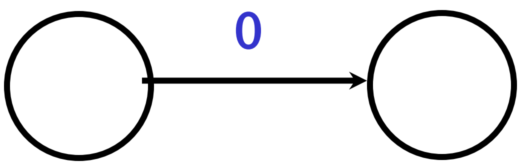


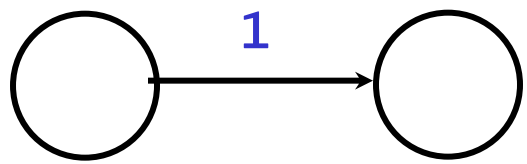
deterministisch
inclusief 'garbage' toestand

val(x) is een drievoud

recept

toestand na lezen van x
val(x) modulo 3



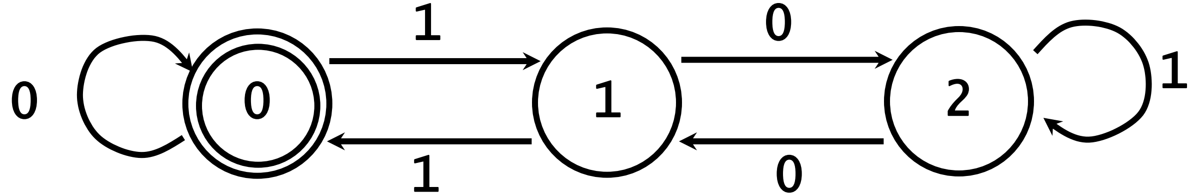


x	x0	x	x1	(input) string {0,1}*
v	2v	v	2v+1	waarde $\mathbb{N}$
0 1 2	0 2 1	0 1 2	1 0 2	waarde (mod 7) toestand

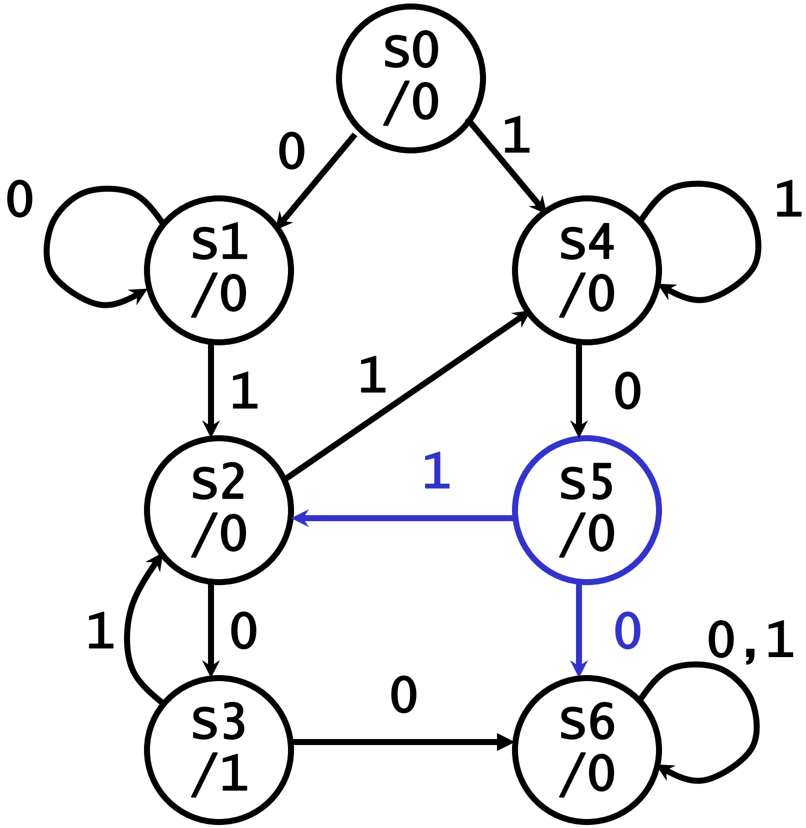
val(x) is een drievoud

uitgewerkt

toestand na lezen van x
val(x) modulo 3



toestand ⇔ eigenschap woord



{ 010, 100 }

woorden $|w| \geq 3$:

state	suffix	100
s1	000	niet
s2	.01	niet
s3	010	niet
s4	.11	niet
s5	110	niet
s6	(100)	wél

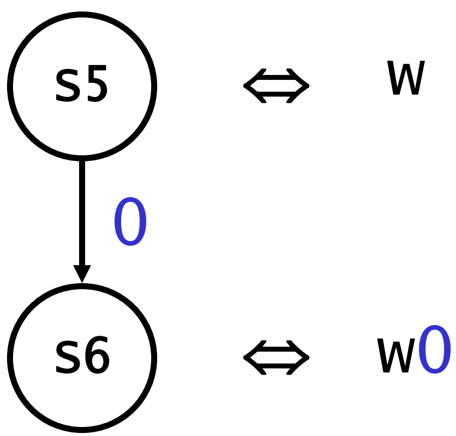
inductiestap

	w ↦	w0	w1
state	s5	s6	s2
suffix	110	1100	1101
subwrd	niet	wel	niet

formuleer een ‘**invariant**’

een relatie tussen (eigenschappen van)
woord en bereikte toestand in automaat

bewijs dat die relatie behouden blijft



state	suffix	100
s5	110	niet
s6	(100)	wél

… inductie

eindige automaat $\mathcal{A} = (Q,\Sigma,E,q_{in},F)$

$E \subseteq Q \times \Sigma \times Q$ éénstapsrelatie

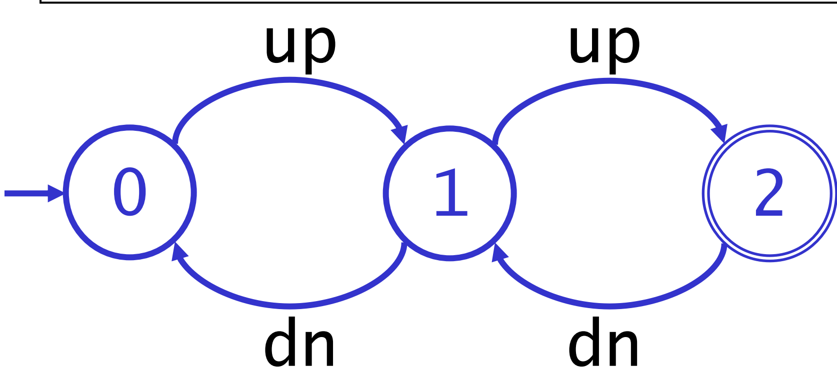
$E^* \subseteq Q \times \Sigma^* \times Q$ meerstapsrelatie

basis $(p,\lambda,p) \in E^*$ voor elke $p \in Q$.

inductiestap

als $(p,w,q) \in E^*$ en $(q,a,r) \in E$

dan $(p,wa,r) \in E^*$



$1 \xrightarrow{up} 2 \xrightarrow{dn} 1 \xrightarrow{dn} 0 \xrightarrow{up} 1$

E	E*
	(1,λ,1)
(1,up,2)	(1,up,2)
(2,dn,1)	(1,up dn,1)
(1,dn,0)	(1,up dn dn,0)
(0,up,1)	(1,up dn dn up,1)

takken rijgen

basis $(p,\lambda,p) \in E^*$ voor elke $p \in Q$.

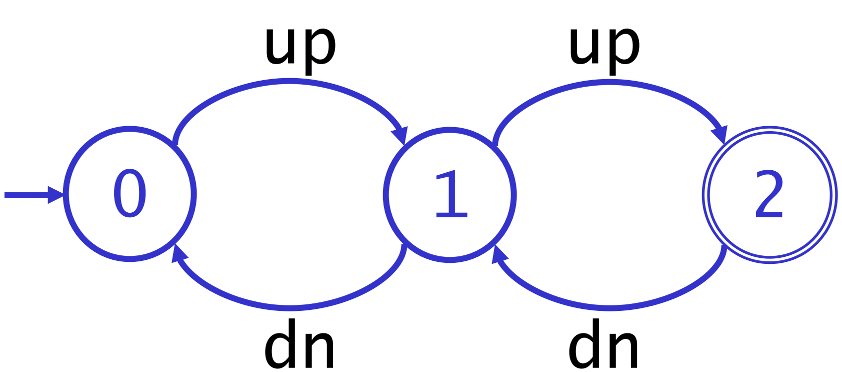
inductiestap

als $(p,w,q) \in E^*$ en $(q,a,r) \in E$

dan $(p,wa,r) \in E^*$

lemma

$(p,w,q) \in E^*$ precies als er een wandeling bestaat van p naar q met label w.



$$1 \xrightarrow{up} 2 \xrightarrow{dn} 1 \xrightarrow{dn} 0 \xrightarrow{up} 1$$

E	E*
	(1,λ,1)
(1,up,2)	(1,up,2)
(2,dn,1)	(1,up dn,1)
(1,dn,0)	(1,up dn dn,0)
(0,up,1)	(1,up dn dn up,1)

een taal K heet *representeerbaar* als er een eindige automaat is met $L(\mathcal{A}) = K$

voorbeelden:

- even aantal 1-en
- {01,001,0001}*
- suffix 1101
- x met val(x) drievoud

representeerbare talen

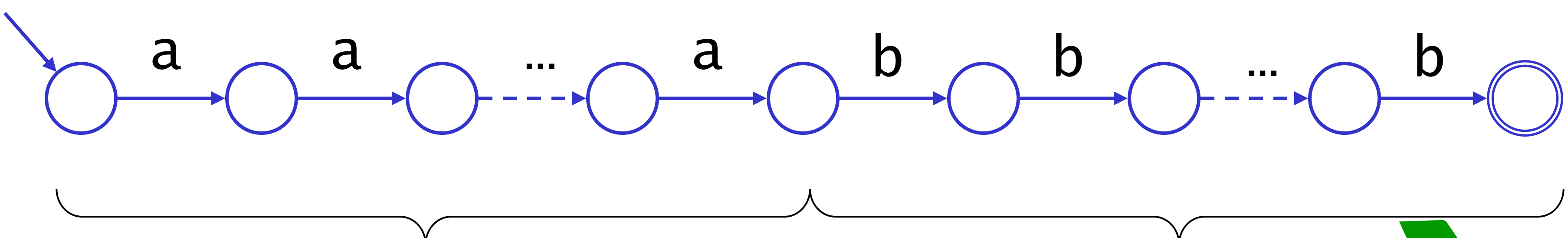
een taal K heet *representeerbaar* als er een eindige automaat is met $L(\mathcal{A}) = K$

eindig veel toestanden … niet elke taal

stelling

$K = \{ a^n b^n \mid n \geq 1 \}$ is *niet* representeerbaar

stel wél … N toestanden



N takken:
hier een toestand dubbel …

Regular Expressions to Finite Automata

compilers (generating a lexical scanner)

Lexical Specification

Regular expressions

NFA

DFA

deterministisch

Table-driven Implementation of DFA

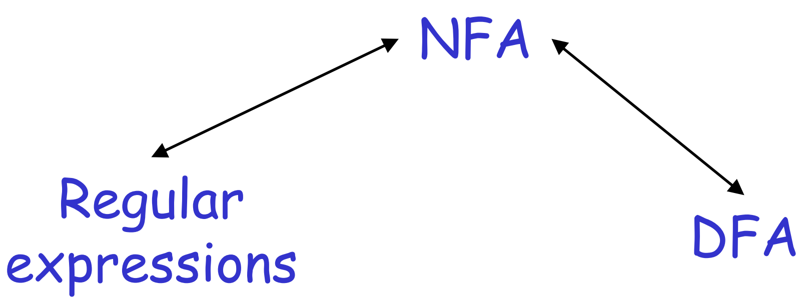
```
state = S    // S is start state
repeat {
     k = next character
     if (k == EOF) // end input
     if (state is a final)
          then accept
          else reject
     state = T[state,k]
     if (state = empty)
     then reject // got stuck
}
```

bekende familie

stelling

dezelfde familie van talen wordt gedefinieerd door

- reguliere expressies 'regulier'
- eindige automaten 'representeerbaar'
- deterministische eindige automaten



zie fi2 'formele talen'

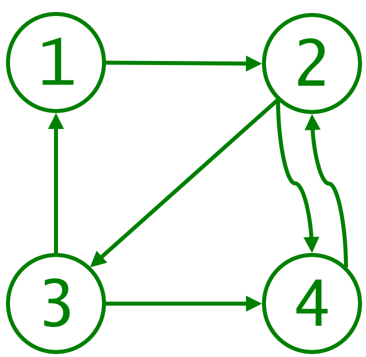
probleem = taal

HPP: Hamilton Path Problem
gegeven: gerichte graaf G, begin- & eindpunt
gevr: is er een Hamilton pad (elke knoop éénmaal)

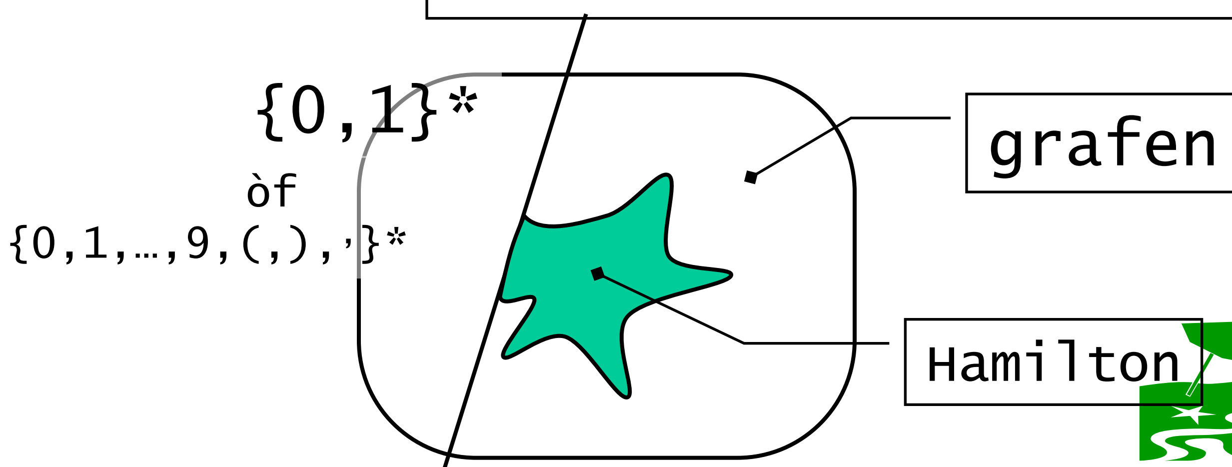
aantal punten   pijlen   begin & eind

4 (1,2)(2,3)(2,4)(3,1)(3,4)(4,2) 1,2

000 100 001010 010011 010100 011001 011100 100010 000 001010



G graaf → woord w(G)
Hamilton grafen → taal Ham
G is Hamilton → w(G) ∈ Ham



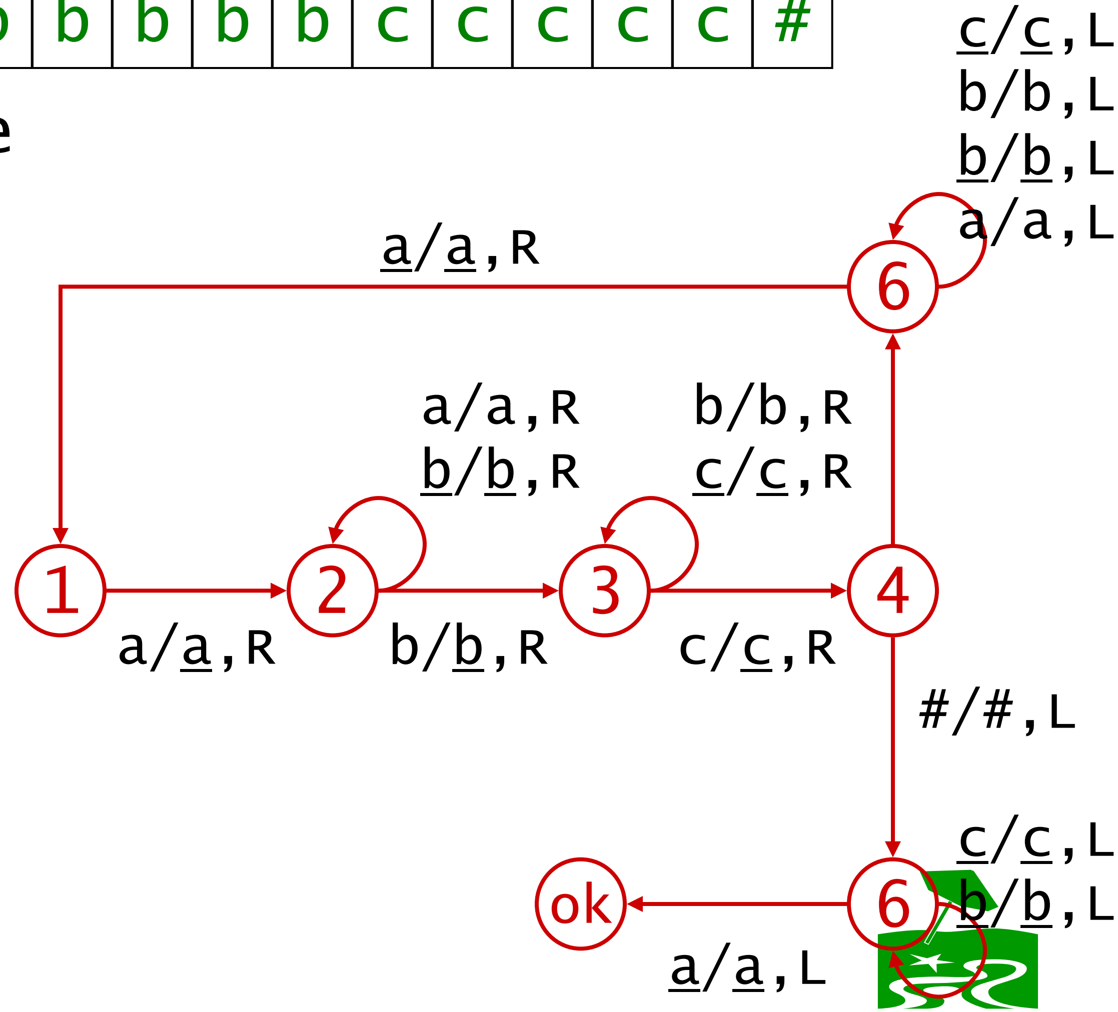
Turing machine

$\{ \ a^n b^n c^n \mid n \geq 0 \ \}$

#	a	a	a	a	a	b	b	b	b	b	c	c	c	c	c	#

tape

1. mark a
2. move to b's
   mark b
3. move to c's
   mark c
4. if another c
5. then back to a's
   goto 1.
   else back to a's
6. check marks
   stop



end...