

## 2. Seminar EVU RegAut

Sigurd Meldgaard

7. februar 2009

# Plan

## Nondeterministiske automater

### Determinisering

### NFA- $\Lambda$ 'er

### Kleenes sætning

Kleenes sætning del 1

Kleenes sætning del 2

### Frokost

### Minimering

Myhill Nerode

### Java projekt

# Ækvivalens ml. Regulære udtryk og FA'er

- Definition af NFA'er og deres sprog
- Delmængdekonstruktionen:  $\text{NFA} \rightarrow \text{FA}$
- Definition af NFA- $\Lambda$ 'er og deres sprog
- $\Lambda$ -eliminering:  $\text{NFA-}\Lambda \rightarrow \text{FA}$
- Kleenes sætning:  $\text{regulært udtryk} \rightarrow \text{NFA-}\Lambda \rightarrow \text{NFA} \rightarrow \text{FA} \rightarrow \text{regulært udtryk}$

# Ækvivalens ml. Regulære udtryk og FA'er

- Definition af NFA'er og deres sprog
- Delmængdekonstruktionen:  $\text{NFA} \rightarrow \text{FA}$
- Definition af NFA- $\Lambda$ 'er og deres sprog
- $\Lambda$ -eliminering:  $\text{NFA-}\Lambda \rightarrow \text{FA}$
- Kleenes sætning:  $\text{regulært udtryk} \rightarrow \text{NFA-}\Lambda \rightarrow \text{NFA} \rightarrow \text{FA} \rightarrow \text{regulært udtryk}$

# Ækvivalens ml. Regulære udtryk og FA'er

- Definition af NFA'er og deres sprog
- Delmængdekonstruktionen:  $\text{NFA} \rightarrow \text{FA}$
- Definition af NFA- $\Lambda$ 'er og deres sprog
- $\Lambda$ -eliminering:  $\text{NFA-}\Lambda \rightarrow \text{FA}$
- Kleenes sætning:  $\text{regulært udtryk} \rightarrow \text{NFA-}\Lambda \rightarrow \text{NFA} \rightarrow \text{FA} \rightarrow \text{regulært udtryk}$

# Ækvivalens ml. Regulære udtryk og FA'er

- Definition af NFA'er og deres sprog
- Delmængdekonstruktionen:  $\text{NFA} \rightarrow \text{FA}$
- Definition af NFA- $\Lambda$ 'er og deres sprog
- $\Lambda$ -eliminering:  $\text{NFA-}\Lambda \rightarrow \text{FA}$
- Kleenes sætning:  $\text{regulært udtryk} \rightarrow \text{NFA-}\Lambda \rightarrow \text{NFA} \rightarrow \text{FA} \rightarrow \text{regulært udtryk}$

# Ækvivalens ml. Regulære udtryk og FA'er

- Definition af NFA'er og deres sprog
- Delmængdekonstruktionen:  $\text{NFA} \rightarrow \text{FA}$
- Definition af NFA- $\Lambda$ 'er og deres sprog
- $\Lambda$ -eliminering:  $\text{NFA-}\Lambda \rightarrow \text{FA}$
- Kleenes sætning:  $\text{regulært udtryk} \rightarrow \text{NFA-}\Lambda \rightarrow \text{NFA} \rightarrow \text{FA} \rightarrow \text{regulært udtryk}$

# Nondeterministiske automater

- NFA'er: som FA'er men
- Der er ikke altid præcis én udgående transition pr. alfabetsymbol for hver tilstand
- Automaten accepterer en streng, hvis det er muligt at "gætte" en vej til accept



# Nondeterministiske automater

- NFA'er: som FA'er men
- Der er ikke altid præcis én udgående transition pr. alfabetsymbol for hver tilstand
- Automaten accepterer en streng, hvis det er muligt at "gætte" en vej til accept

# Nondeterministiske automater

- NFA'er: som FA'er men
- Der er ikke altid præcis én udgående transition pr. alfabetsymbol for hver tilstand
- Automaten accepterer en streng, hvis det er muligt at "gætte" en vej til accept

# Eksempel

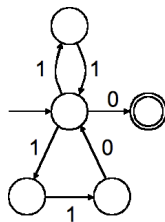
- Hvordan laver man en automat, der svarer til det regulære udtryk  $(11 + 110)^*0$  ?
- Det er ikke trivielt med FA'er...
- En nondeterministisk automat:

# Eksempel

- Hvordan laver man en automat, der svarer til det regulære udtryk  $(11 + 110)^*0$  ?
- Det er ikke trivielt med FA'er...
- En nondeterministisk automat:

# Eksempel

- Hvordan laver man en automat, der svarer til det regulære udtryk  $(11 + 110)^*0$  ?
- Det er ikke trivielt med FA'er...
- En nondeterministisk automat:



# Formel definition af NFA

*Everything is vague to a degree you do not realize till you have tried to make it precise.*

## Bertrand Russell

En nondeterministisk endelig automat (NFA) er et 5-tupel  $(Q, \Sigma, q_0, A, \delta)$  hvor

- $Q$  er en endelig mængde af tilstande
- $\Sigma$  er et alfabet
- $q_0 \in Q$  er en starttilstand
- $A \subseteq Q$  er accepttilstande
- $\delta: Q \times \Sigma \rightarrow 2^Q$  er en transitionsfunktion  
Det betyder at  $\delta(q, a)$  giver en *mængde* af tilstande.

# Formel definition af NFA

*Everything is vague to a degree you do not realize till you have tried to make it precise.*

Bertrand Russell

En nondeterministisk endelig automat (NFA) er et 5-tupel  $(Q, \Sigma, q_0, A, \delta)$  hvor

- $Q$  er en endelig mængde af tilstande
- $\Sigma$  er et alfabet
- $q_0 \in Q$  er en starttilstand
- $A \subseteq Q$  er accepttilstande
- $\delta: Q \times \Sigma \rightarrow 2^Q$  er en transitionsfunktion  
 Det betyder at  $\delta(q, a)$  giver en *mængde* af tilstande.

# Formel definition af NFA

*Everything is vague to a degree you do not realize till you have tried to make it precise.*

Bertrand Russell

En nondeterministisk endelig automat (NFA) er et 5-tupel  $(Q, \Sigma, q_0, A, \delta)$  hvor

- $Q$  er en endelig mængde af tilstande
- $\Sigma$  er et alfabet
- $q_0 \in Q$  er en starttilstand
- $A \subseteq Q$  er accepttilstande
- $\delta: Q \times \Sigma \rightarrow 2^Q$  er en transitionsfunktion  
 Det betyder at  $\delta(q, a)$  giver en *mængde* af tilstande.



# Formel definition af NFA

*Everything is vague to a degree you do not realize till you have tried to make it precise.*

Bertrand Russell

En nondeterministisk endelig automat (NFA) er et 5-tupel  $(Q, \Sigma, q_0, A, \delta)$  hvor

- $Q$  er en endelig mængde af tilstande
  - $\Sigma$  er et alfabet
  - $q_0 \in Q$  er en starttilstand
  - $A \subseteq Q$  er accepttilstande
  - $\delta: Q \times \Sigma \rightarrow 2^Q$  er en transitionsfunktion
- Det betyder at  $\delta(q, a)$  giver en *mængde* af tilstande.

# Formel definition af NFA

*Everything is vague to a degree you do not realize till you have tried to make it precise.*

Bertrand Russell

En nondeterministisk endelig automat (NFA) er et 5-tupel  $(Q, \Sigma, q_0, A, \delta)$  hvor

- $Q$  er en endelig mængde af tilstande
- $\Sigma$  er et alfabet
- $q_0 \in Q$  er en starttilstand
- $A \subseteq Q$  er accepttilstande
- $\delta: Q \times \Sigma \rightarrow 2^Q$  er en transitionsfunktion  
 Det betyder at  $\delta(q, a)$  giver en *mængde* af tilstande.

# Formel definition af NFA

*Everything is vague to a degree you do not realize till you have tried to make it precise.*

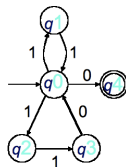
Bertrand Russell

En nondeterministisk endelig automat (NFA) er et 5-tupel  $(Q, \Sigma, q_0, A, \delta)$  hvor

- $Q$  er en endelig mængde af tilstande
  - $\Sigma$  er et alfabet
  - $q_0 \in Q$  er en starttilstand
  - $A \subseteq Q$  er accepttilstande
  - $\delta: Q \times \Sigma \rightarrow 2^Q$  er en transitionsfunktion
- Det betyder at  $\delta(q, a)$  giver en *mængde* af tilstande.

# Eksempel på en NFA

Her er den grafiske representation af en automat:



- Den repræsenterer 5-tuplet:

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$

- $\Sigma = \{0, 1\}$

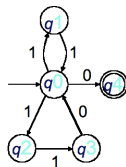
- $A = \{q_4\}$

- $\delta : Q \times \Sigma \rightarrow 2^Q$  Er funktionen i denne tabel:

	0	1
$q_0$	$\{q_4\}$	$\{q_1, q_2\}$
$q_1$	$\emptyset$	$\{q_0\}$
$q_2$	$\emptyset$	$\{q_3\}$
$q_3$	$\{q_0\}$	$\emptyset$
$q_4$	$\emptyset$	$\emptyset$

# Eksempel på en NFA

Her er den grafiske representation af en automat:

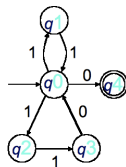


- Den repræsenterer 5-tuplet:
- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- $A = \{q_4\}$
- $\delta : Q \times \Sigma \rightarrow 2^Q$  Er funktionen i denne tabel:

	0	1
$q_0$	$\{q_4\}$	$\{q_1, q_2\}$
$q_1$	$\emptyset$	$\{q_0\}$
$q_2$	$\emptyset$	$\{q_3\}$
$q_3$	$\{q_0\}$	$\emptyset$
$q_4$	$\emptyset$	$\emptyset$

# Eksempel på en NFA

Her er den grafiske representation af en automat:



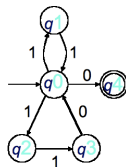
- Den repræsenterer 5-tuplet:
- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- $A = \{q_4\}$

- $\delta : Q \times \Sigma \rightarrow 2^Q$  Er funktionen i denne tabel:

	0	1
$q_0$	$\{q_4\}$	$\{q_1, q_2\}$
$q_1$	$\emptyset$	$\{q_0\}$
$q_2$	$\emptyset$	$\{q_3\}$
$q_3$	$\{q_0\}$	$\emptyset$
$q_4$	$\emptyset$	$\emptyset$

# Eksempel på en NFA

Her er den grafiske representation af en automat:



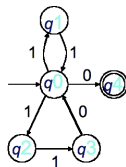
- Den repræsenterer 5-tuplet:
- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- $A = \{q_4\}$

- $\delta : Q \times \Sigma \rightarrow 2^Q$  Er funktionen i denne tabel:

	0	1
$q_0$	$\{q_4\}$	$\{q_1, q_2\}$
$q_1$	$\emptyset$	$\{q_0\}$
$q_2$	$\emptyset$	$\{q_3\}$
$q_3$	$\{q_0\}$	$\emptyset$
$q_4$	$\emptyset$	$\emptyset$

# Eksempel på en NFA

Her er den grafiske representation af en automat:



- Den repræsenterer 5-tuplet:
- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- $A = \{q_4\}$
- $\delta : Q \times \Sigma \rightarrow 2^Q$  Er funktionen i denne tabel:

	0	1
$q_0$	$\{q_4\}$	$\{q_1, q_2\}$
$q_1$	$\emptyset$	$\{q_0\}$
$q_2$	$\emptyset$	$\{q_3\}$
$q_3$	$\{q_0\}$	$\emptyset$
$q_4$	$\emptyset$	$\emptyset$



# Sproget af en NFA

Givet en NFA  $M = (Q, \Sigma, q_0, A, \delta)$ :

- Definer den udvidede transitionsfunktion:

$$\delta^*(q, x) = \begin{cases} \{q\} & \text{hvis } x = \Lambda \\ \bigcup_{r \in \delta^*(q, y)} \delta(r, a) & \text{hvis } x = y \cdot a \end{cases}$$

- $x \in \Sigma^*$  accepteres af en NFA  $M$  hvis og kun hvis  $\delta^*(q_0, x) \cap A \neq \emptyset$
- $L(M) = \{x \mid x \text{ accepteres af } M\}$

# Sproget af en NFA

Givet en NFA  $M = (Q, \Sigma, q_0, A, \delta)$ :

- Definer den udvidede transitionsfunktion:

$$\delta^*(q, x) = \begin{cases} \{q\} & \text{hvis } x = \Lambda \\ \bigcup_{r \in \delta^*(q, y)} \delta(r, a) & \text{hvis } x = y \cdot a \end{cases}$$

- $x \in \Sigma^*$  accepteres af en NFA  $M$  hvis og kun hvis  $\delta^*(q_0, x) \cap A \neq \emptyset$
- $L(M) = \{x \mid x \text{ accepteres af } M\}$

# Sproget af en NFA

Givet en NFA  $M = (Q, \Sigma, q_0, A, \delta)$ :

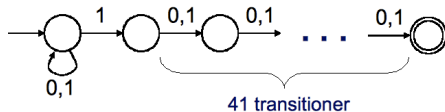
- Definer den udvidede transitionsfunktion:

$$\delta^*(q, x) = \begin{cases} \{q\} & \text{hvis } x = \Lambda \\ \bigcup_{r \in \delta^*(q, y)} \delta(r, a) & \text{hvis } x = y \cdot a \end{cases}$$

- $x \in \Sigma^*$  accepteres af en NFA  $M$  hvis og kun hvis  $\delta^*(q_0, x) \cap A \neq \emptyset$
- $L(M) = \{x \mid x \text{ accepteres af } M\}$

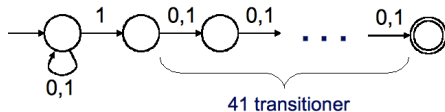
# NFA'er er ofte mindre end FA'er

- $L_{42} = \{x \mid |x| \geq 42 \wedge 42 \text{ tegn fra højre er et } 1\}$
- Sidste seminar viste vi at det kræver mindst  $2^{42}$  tilstande at lave en FA der genkender  $L_{42}$
- En **NFA** der genkender  $L_{42}$  med 43 tilstande:



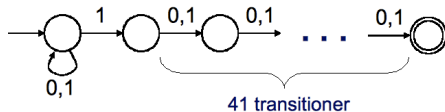
# NFA'er er ofte mindre end FA'er

- $L_{42} = \{x \mid |x| \geq 42 \wedge 42 \text{ tegn fra højre er et } 1\}$
- Sidste seminar viste vi at det kræver mindst  $2^{42}$  tilstande at lave en FA der genkender  $L_{42}$
- En **NFA** der genkender  $L_{42}$  med 43 tilstande:



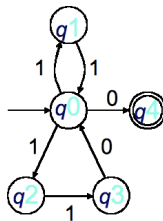
# NFA'er er ofte mindre end FA'er

- $L_{42} = \{x \mid |x| \geq 42 \wedge 42 \text{ tegn fra højre er et } 1\}$
- Sidste seminar viste vi at det kræver mindst  $2^{42}$  tilstande at lave en FA der genkender  $L_{42}$
- En **N**FA der genkender  $L_{42}$  med 43 tilstande:



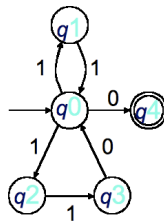
## Quiz

Bliver strengen 110110  
accepteret af denne automat?



## Quiz

Bliver strengen 110110  
accepteret af denne automat?



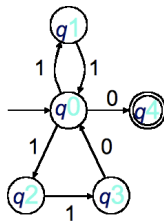
Ja! der findes en sti til accept:

$$q_0 \xrightarrow{1} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_4$$



## Quiz

Bliver strengen 110110  
accepteret af denne automat?



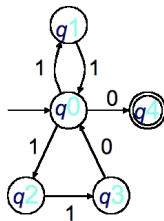
Ja! der findes en sti til accept:

$$q_0 \xrightarrow{1} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_4$$

Vi kan systematisk lede efter sådan en sti:  $\{q_0\}$

## Quiz

Bliver strengen 110110  
accepteret af denne automat?



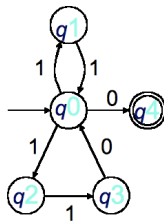
Ja! der findes en sti til accept:

$$q_0 \xrightarrow{1} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_4$$

Vi kan systematisk lede efter sådan en sti:  $\{q_0\} \xrightarrow{1} \{q_1, q_2\}$

## Quiz

Bliver strengen 110110  
accepteret af denne automat?



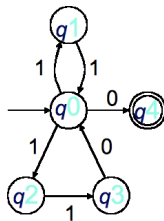
Ja! der findes en sti til accept:

$$q_0 \xrightarrow{1} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_4$$

Vi kan systematisk lede efter sådan en sti:  $\{q_0\} \xrightarrow{1} \{q_1, q_2\} \xrightarrow{1} \{q_0, q_3\}$

## Quiz

Bliver strengen 110110  
accepteret af denne automat?



Ja! der findes en sti til accept:

$$q_0 \xrightarrow{1} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_4$$

Vi kan systematisk lede efter sådan en sti:  $\{q_0\} \xrightarrow{1} \{q_1, q_2\} \xrightarrow{1} \{q_0, q_3\} \xrightarrow{0} \{q_4, q_0\} \xrightarrow{1} \{q_1, q_2\} \xrightarrow{1} \{q_0, q_3\} \xrightarrow{0} \{q_4, q_0\}$

## Øvelser:

- [Martin]: opg. 4.2. (p. 156) Drawing an NFA and using the definition of  $\delta^*$

# Plan

Nondeterministiske automater

Determinisering

NFA- $\Lambda$ 'er

Kleenes sætning

Kleenes sætning del 1

Kleenes sætning del 2

Frokost

Minimering

Myhill Nerode

Java projekt

# Enhver FA kan oversættes til en NFA

- Hvis man ser på den grafiske repræsentation, så er det trivielt, en FA er bare en simpel NFA.
- Formelt: givet en FA:  $N = (Q, \Sigma, q_0, A, \delta)$
- Konstruer en NFA:  $M = (Q, \Sigma, q_0, A, \delta')$  hvor:

$$\delta'(q, a) = \{\delta(q, a)\}$$

- Husk bevis for korrekthed!  $L(M) = L(N)$  fordi... (induktion i længden af en inputstreng)

# Enhver FA kan oversættes til en NFA

- Hvis man ser på den grafiske repræsentation, så er det trivielt, en FA er bare en simpel NFA.
- Formelt: givet en FA:  $N = (Q, \Sigma, q_0, A, \delta)$
- Konstruer en NFA:  $M = (Q, \Sigma, q_0, A, \delta')$  hvor:

$$\delta'(q, a) = \{\delta(q, a)\}$$

- Husk bevis for korrekthed!  $L(M) = L(N)$  fordi... (induktion i længden af en inputstreng)



# Enhver FA kan oversættes til en NFA

- Hvis man ser på den grafiske repræsentation, så er det trivielt, en FA er bare en simpel NFA.
- Formelt: givet en FA:  $N = (Q, \Sigma, q_0, A, \delta)$
- Konstruer en NFA:  $M = (Q, \Sigma, q_0, A, \delta')$  hvor:

$$\delta'(q, a) = \{\delta(q, a)\}$$

- Husk bevis for korrekthed!  $L(M) = L(N)$  fordi... (induktion i længden af en inputstreng)

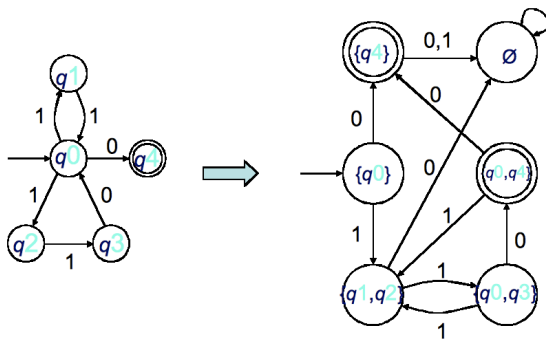
# Enhver FA kan oversættes til en NFA

- Hvis man ser på den grafiske repræsentation, så er det trivielt, en FA er bare en simpel NFA.
- Formelt: givet en FA:  $N = (Q, \Sigma, q_0, A, \delta)$
- Konstruer en NFA:  $M = (Q, \Sigma, q_0, A, \delta')$  hvor:

$$\delta'(q, a) = \{\delta(q, a)\}$$

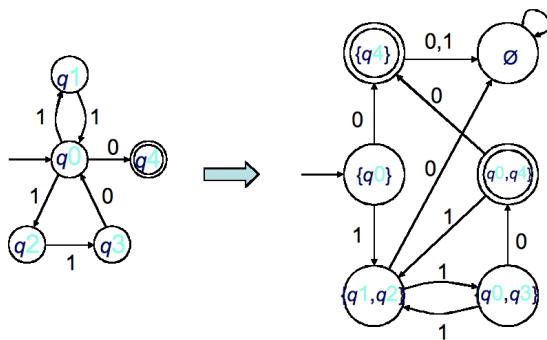
- Husk bevis for korrekthed!  $L(M) = L(N)$  fordi... (induktion i længden af en inputstreng)

# Enhver NFA kan oversættes til en FA



Dette kaldes determinisering

# Enhver NFA kan oversættes til en FA



Dette kaldes determinisering

# Delmængdekonstruktionen (determinisering)

Givet en NFA:  $N = (Q, \Sigma, q_0, A, \delta)$

Konstruer en FA:  $M = (Q', \Sigma, q'_0, A', \delta')$

- $Q' = 2^Q$  (tilstandene i FA'en svarer til en mængde af tilstande i NFA'en)
- $q'_0 = \{q_0\}$
- $A' = \{q \in Q' \mid A \cap q \neq \emptyset\}$
- $\delta'(q, a) = \bigcup_{r \in q} \delta(r, a)$
- Husk bevis for korrekthed...

# Delmængdekonstruktionen (determinisering)

Givet en NFA:  $N = (Q, \Sigma, q_0, A, \delta)$

Konstruer en FA:  $M = (Q', \Sigma, q'_0, A', \delta')$

- $Q' = 2^Q$  (tilstandene i FA'en svarer til en mængde af tilstande i NFA'en)
- $q'_0 = \{q_0\}$
- $A' = \{q \in Q' \mid A \cap q \neq \emptyset\}$
- $\delta'(q, a) = \bigcup_{r \in q} \delta(r, a)$
- Husk bevis for korrekthed...

# Delmængdekonstruktionen (determinisering)

Givet en NFA:  $N = (Q, \Sigma, q_0, A, \delta)$

Konstruer en FA:  $M = (Q', \Sigma, q'_0, A', \delta')$

- $Q' = 2^Q$  (tilstandene i FA'en svarer til en mængde af tilstande i NFA'en)
- $q'_0 = \{q_0\}$
- $A' = \{q \in Q' \mid A \cap q \neq \emptyset\}$
- $\delta'(q, a) = \bigcup_{r \in q} \delta(r, a)$
- Husk bevis for korrekthed...

# Delmængdekonstruktionen (determinisering)

Givet en NFA:  $N = (Q, \Sigma, q_0, A, \delta)$

Konstruer en FA:  $M = (Q', \Sigma, q'_0, A', \delta')$

- $Q' = 2^Q$  (tilstandene i FA'en svarer til en mængde af tilstande i NFA'en)
- $q'_0 = \{q_0\}$
- $A' = \{q \in Q' \mid A \cap q \neq \emptyset\}$
- $\delta'(q, a) = \bigcup_{r \in q} \delta(r, a)$
- Husk bevis for korrekthed...



# Delmængdekonstruktionen (determinisering)

Givet en NFA:  $N = (Q, \Sigma, q_0, A, \delta)$

Konstruer en FA:  $M = (Q', \Sigma, q'_0, A', \delta')$

- $Q' = 2^Q$  (tilstandene i FA'en svarer til en mængde af tilstande i NFA'en)
- $q'_0 = \{q_0\}$
- $A' = \{q \in Q' \mid A \cap q \neq \emptyset\}$
- $\delta'(q, a) = \bigcup_{r \in q} \delta(r, a)$
- Husk bevis for korrekthed...

# Delmængdekonstruktionen (determinisering)

Givet en NFA:  $N = (Q, \Sigma, q_0, A, \delta)$

Konstruer en FA:  $M = (Q', \Sigma, q'_0, A', \delta')$

- $Q' = 2^Q$  (tilstandene i FA'en svarer til en mængde af tilstande i NFA'en)
- $q'_0 = \{q_0\}$
- $A' = \{q \in Q' \mid A \cap q \neq \emptyset\}$
- $\delta'(q, a) = \bigcup_{r \in q} \delta(r, a)$
- Husk bevis for korrekthed...

# Bevis for korrekthed af delmængdekonstruktionen

- Husk definitionen for  $L(M)$  og  $L(N)$

$$L(M) = \{x \mid \delta'^*(q'_0, x) \in A'\}$$

$$L(N) = \{x \mid \delta^*(q_0, x) \cap A \neq \emptyset\}$$

- Lemma:

$$\forall x \in \Sigma^* : \delta'^*(q'_0, x) = \delta^*(q_0, x)$$

Bevis: induktion i strukturen af  $x$ ...

- $\delta'^*(q'_0, x) \in A' \stackrel{\text{lemma}}{\Leftrightarrow} \delta^*(q_0, x) \in A' \stackrel{\text{def af } A}{\Leftrightarrow} \delta^*(q_0, x) \cap A \neq \emptyset$
- D.v.s.  $L(M) = L(N)$

# Bevis for korrekthed af delmængdekonstruktionen

- Husk definitionen for  $L(M)$  og  $L(N)$

$$L(M) = \{x | \delta'^*(q'_0, x) \in A'\}$$

$$L(N) = \{x | \delta^*(q_0, x) \cap A \neq \emptyset\}$$

- Lemma:

$$\forall x \in \Sigma^* : \delta'^*(q'_0, x) = \delta^*(q_0, x)$$

Bevis: induktion i strukturen af  $x$ ...

- $\delta'^*(q'_0, x) \in A' \stackrel{\text{lemma}}{\Leftrightarrow} \delta^*(q_0, x) \in A' \stackrel{\text{def af } A}{\Leftrightarrow} \delta^*(q_0, x) \cap A \neq \emptyset$
- D.v.s.  $L(M) = L(N)$

# Bevis for korrekthed af delmængdekonstruktionen

- Husk definitionen for  $L(M)$  og  $L(N)$

$$L(M) = \{x | \delta'^*(q'_0, x) \in A'\}$$

$$L(N) = \{x | \delta^*(q_0, x) \cap A \neq \emptyset\}$$

- Lemma:

$$\forall x \in \Sigma^* : \delta'^*(q'_0, x) = \delta^*(q_0, x)$$

Bevis: induktion i strukturen af  $x$ ...

- $\delta'^*(q'_0, x) \in A' \stackrel{\text{lemma}}{\Leftrightarrow} \delta^*(q_0, x) \in A' \stackrel{\text{def af } A}{\Leftrightarrow} \delta^*(q_0, x) \cap A \neq \emptyset$
- D.v.s.  $L(M) = L(N)$

# Bevis for korrekthed af delmængdekonstruktionen

- Husk definitionen for  $L(M)$  og  $L(N)$

$$L(M) = \{x | \delta'^*(q'_0, x) \in A'\}$$

$$L(N) = \{x | \delta^*(q_0, x) \cap A \neq \emptyset\}$$

- Lemma:

$$\forall x \in \Sigma^* : \delta'^*(q'_0, x) = \delta^*(q_0, x)$$

Bevis: induktion i strukturen af  $x$ ...

- $\delta'^*(q'_0, x) \in A' \stackrel{\text{lemma}}{\Leftrightarrow} \delta^*(q_0, x) \in A' \stackrel{\text{def af } A}{\Leftrightarrow} \delta^*(q_0, x) \cap A \neq \emptyset$
- D.v.s.  $L(M) = L(N)$

## Nøjes med opnåelige tilstande

- Delmængdekonstruktionen bruger  $Q' = 2^Q$
- Som ved produktkonstruktionen: I praksis er hele tilstandsrummet sjældent nødvendigt
- Som sidste gang: Kun tilstande, der er opnåelige fra starttilstanden er relevante for sproget (Bevis dette: Opg. 3.29, p. 117)

# Nøjes med opnåelige tilstande

- Delmængdekonstruktionen bruger  $Q' = 2^Q$
- Som ved produktkonstruktionen: I praksis er hele tilstandsrummet sjældent nødvendigt
- Som sidste gang: Kun tilstande, der er opnåelige fra starttilstanden er relevante for sproget (Bevis dette: Opg. 3.29, p. 117)



## Nøjes med opnåelige tilstande

- Delmængdekonstruktionen bruger  $Q' = 2^Q$
- Som ved produktkonstruktionen: I praksis er hele tilstandsrummet sjældent nødvendigt
- Som sidste gang: Kun tilstande, der er opnåelige fra starttilstanden er relevante for sproget (Bevis dette: Opg. 3.29, p. 117)

## Øvelser

- [Martin] Opg. 4.10 (a+e) (p. 157)  
Udfør selv delmængdekonstruktionen.

# Plan

Nondeterministiske automater

Determinisering

NFA- $\Lambda$ 'er

Kleenes sætning

Kleenes sætning del 1

Kleenes sætning del 2

Frokost

Minimering

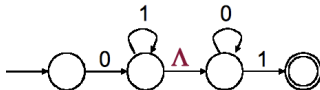
Myhill Nerode

Java projekt

# NFA'er med $\Lambda$ -transitioner

- For nemt at kunne oversætte regulære udtryk til automater generaliserer vi automaterne yderligere
- En  $\Lambda$ -transition er en transition, der ikke læser et symbol fra input-strengen

- Eksempel på en NFA- $\Lambda$ :

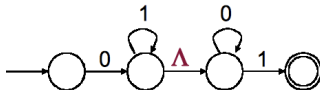


- Automaten “bestemmer selv” om den vil følge  $\Lambda$ -transitionen  
Eksempel: strengen 011 accepteres

## NFA'er med $\Lambda$ -transitioner

- For nemt at kunne oversætte regulære udtryk til automater generaliserer vi automaterne yderligere
- En  $\Lambda$ -transition er en transition, der ikke læser et symbol fra input-strengen

- Eksempel på en NFA- $\Lambda$ :

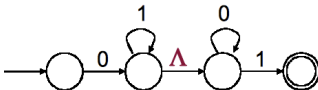


- Automaten “bestemmer selv” om den vil følge  $\Lambda$ -transitionen  
Eksempel: strengen 011 accepteres

# NFA'er med $\Lambda$ -transitioner

- For nemt at kunne oversætte regulære udtryk til automater generaliserer vi automaterne yderligere
- En  $\Lambda$ -transition er en transition, der ikke læser et symbol fra input-strengen

- Eksempel på en NFA- $\Lambda$ :

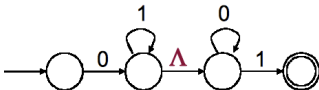


- Automaten “bestemmer selv” om den vil følge  $\Lambda$ -transitionen  
Eksempel: strengen 011 accepteres

# NFA'er med $\Lambda$ -transitioner

- For nemt at kunne oversætte regulære udtryk til automater generaliserer vi automaterne yderligere
- En  $\Lambda$ -transition er en transition, der ikke læser et symbol fra input-strengen

- Eksempel på en NFA- $\Lambda$ :



- Automaten “bestemmer selv” om den vil følge  $\Lambda$ -transitionen  
Eksempel: strengen 011 accepteres

# Formel definition af NFA- $\Lambda$

En nondeterministisk endelig automat med  $\Lambda$ -transitioner (NFA- $\Lambda$ ) er et 5-tupel  $(Q, \Sigma, q_0, A, \delta)$  hvor

- $Q$  er en endelig mængde af tilstande
- $\Sigma$  er et alfabet
- $q_0 \in Q$  er en starttilstand
- $A \subseteq Q$  er accepttilstande
- $\delta : Q \times (\Sigma \cup \Lambda) \rightarrow 2^Q$  er en transitionsfunktion



## Formel definition af NFA- $\Lambda$

En nondeterministisk endelig automat med  $\Lambda$ -transitioner (NFA- $\Lambda$ ) er et 5-tupel  $(Q, \Sigma, q_0, A, \delta)$  hvor

- $Q$  er en endelig mængde af tilstande
- $\Sigma$  er et alfabet
- $q_0 \in Q$  er en starttilstand
- $A \subseteq Q$  er accepttilstande
- $\delta : Q \times (\Sigma \cup \Lambda) \rightarrow 2^Q$  er en transitionsfunktion

## Formel definition af NFA- $\Lambda$

En nondeterministisk endelig automat med  $\Lambda$ -transitioner (NFA- $\Lambda$ ) er et 5-tupel  $(Q, \Sigma, q_0, A, \delta)$  hvor

- $Q$  er en endelig mængde af tilstande
- $\Sigma$  er et alfabet
- $q_0 \in Q$  er en starttilstand
- $A \subseteq Q$  er accepttilstande
- $\delta : Q \times (\Sigma \cup \Lambda) \rightarrow 2^Q$  er en transitionsfunktion

## Formel definition af NFA- $\Lambda$

En nondeterministisk endelig automat med  $\Lambda$ -transitioner (NFA- $\Lambda$ ) er et 5-tupel  $(Q, \Sigma, q_0, A, \delta)$  hvor

- $Q$  er en endelig mængde af tilstande
- $\Sigma$  er et alfabet
- $q_0 \in Q$  er en starttilstand
- $A \subseteq Q$  er accepttilstande
- $\delta : Q \times (\Sigma \cup \Lambda) \rightarrow 2^Q$  er en transitionsfunktion

## Formel definition af NFA- $\Lambda$

En nondeterministisk endelig automat med  $\Lambda$ -transitioner (NFA- $\Lambda$ ) er et 5-tupel  $(Q, \Sigma, q_0, A, \delta)$  hvor

- $Q$  er en endelig mængde af tilstande
- $\Sigma$  er et alfabet
- $q_0 \in Q$  er en starttilstand
- $A \subseteq Q$  er accepttilstande
- $\delta : Q \times (\Sigma \cup \Lambda) \rightarrow 2^Q$  er en transitionsfunktion

## $\Lambda$ -lukning af en tilstandsmængde ( $\Lambda$ -closure)

- Hvor kan man komme til ved kun at bruge  $\Lambda$ -transitioner?
- Givet en mængde  $S \subseteq Q$ , definer  $\Lambda$ -lukningen  $\Lambda(S)$  som den mindste mængde der opfylder flg.:
- $S \subseteq \Lambda(S)$
- $\forall q \in \Lambda(S) : \delta(q, \Lambda) \in \Lambda(S)$

## $\Lambda$ -lukning af en tilstandsmængde ( $\Lambda$ -closure)

- Hvor kan man komme til ved kun at bruge  $\Lambda$ -transitioner?
- Givet en mængde  $S \subseteq Q$ , definer  $\Lambda$ -lukningen  $\Lambda(S)$  som den mindste mængde der opfylder flg.:
- $S \subseteq \Lambda(S)$
- $\forall q \in \Lambda(S) : \delta(q, \Lambda) \in \Lambda(S)$

## $\Lambda$ -lukning af en tilstandsmængde ( $\Lambda$ -closure)

- Hvor kan man komme til ved kun at bruge  $\Lambda$ -transitioner?
- Givet en mængde  $S \subseteq Q$ , definer  $\Lambda$ -lukningen  $\Lambda(S)$  som den mindste mængde der opfylder flg.:
- $S \subseteq \Lambda(S)$
- $\forall q \in \Lambda(S) : \delta(q, \Lambda) \in \Lambda(S)$

## $\Lambda$ -lukning af en tilstandsmængde ( $\Lambda$ -closure)

- Hvor kan man komme til ved kun at bruge  $\Lambda$ -transitioner?
- Givet en mængde  $S \subseteq Q$ , definer  $\Lambda$ -lukningen  $\Lambda(S)$  som den mindste mængde der opfylder flg.:
- $S \subseteq \Lambda(S)$
- $\forall q \in \Lambda(S) : \delta(q, \Lambda) \in \Lambda(S)$



# Sproget for en NFA- $\Lambda$

- Givet en NFA- $\Lambda$   $M = (Q, \Sigma, q_0, A, \delta)$ , definer den udvidede transitionsfunktion  $\delta^* : Q \times \Sigma^* \rightarrow 2^Q$  ved

$$\delta^*(q, x) = \begin{cases} \Lambda(q) & \text{hvis } x = \Lambda \\ \Lambda(\bigcup_{r \in \delta^*(q, y)} \delta(r, a)) & \text{hvis } x = y \cdot a \end{cases}$$

# Sproget for en NFA- $\Lambda$

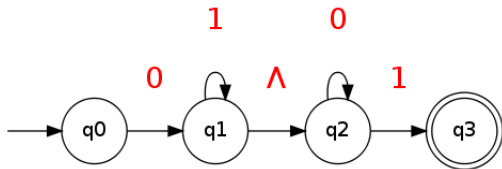
- Givet en NFA- $\Lambda$   $M = (Q, \Sigma, q_0, A, \delta)$ , definer den udvidede transitionsfunktion  $\delta^* : Q \times \Sigma^* \rightarrow 2^Q$  ved

- 

$$\delta^*(q, x) = \begin{cases} \Lambda(q) & \text{hvis } x = \Lambda \\ \Lambda(\bigcup_{r \in \delta^*(q, y)} \delta(r, a)) & \text{hvis } x = y \cdot a \end{cases}$$

## Quiz

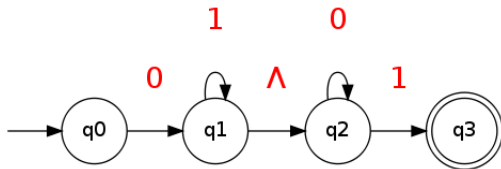
- Hvad er  $\delta^*(q_0, 01)$  for denne NFA- $\Lambda$ ?



- $\delta^*(q_0, \Lambda) = \Lambda(q_0) = \{q_0\}$
- $\delta^*(q_0, \Lambda \cdot 0) = \Lambda(\bigcup_{r \in \delta^*(q_0, \Lambda)} \delta(r, 0)) = \{q_1, q_2\}$
- $\delta^*(q_0, \Lambda \cdot 0 \cdot 1) = \Lambda(\bigcup_{r \in \delta^*(q_0, \Lambda \cdot 0)} \delta(r, 1)) = \Lambda(\{q_1, q_2\} \cup \{q_3\}) = \{q_1, q_2, q_3\}$
- d.v.s. strengen 01 bliver accepteret af automaten.

## Quiz

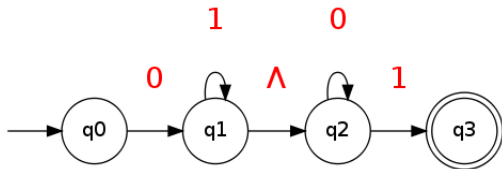
- Hvad er  $\delta^*(q_0, 01)$  for denne NFA- $\Lambda$ ?



- $\delta^*(q_0, \Lambda) = \Lambda(q_0) = \{q_0\}$
- $\delta^*(q_0, \Lambda \cdot 0) = \Lambda(\bigcup_{r \in \delta^*(q_0, \Lambda)} \delta(r, 0)) = \{q_1, q_2\}$
- $\delta^*(q_0, \Lambda \cdot 0 \cdot 1) = \Lambda(\bigcup_{r \in \delta^*(q_0, \Lambda \cdot 0)} \delta(r, 1)) = \Lambda(\{q_1, q_2\} \cup \{q_3\}) = \{q_1, q_2, q_3\}$
- d.v.s. strengen 01 bliver accepteret af automaten.

## Quiz

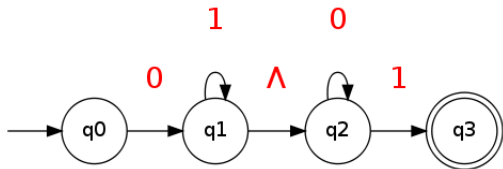
- Hvad er  $\delta^*(q_0, 01)$  for denne NFA- $\Lambda$ ?



- $\delta^*(q_0, \Lambda) = \Lambda(q_0) = \{q_0\}$
- $\delta^*(q_0, \Lambda \cdot 0) = \Lambda(\bigcup_{r \in \delta^*(q_0, \Lambda)} \delta(r, 0)) = \{q_1, q_2\}$
- $\delta^*(q_0, \Lambda \cdot 0 \cdot 1) = \Lambda(\bigcup_{r \in \delta^*(q_0, \Lambda \cdot 0)} \delta(r, 1)) = \Lambda(\{q_1, q_2\} \cup \{q_3\}) = \{q_1, q_2, q_3\}$
- d.v.s. strengen 01 bliver accepteret af automaten.

## Quiz

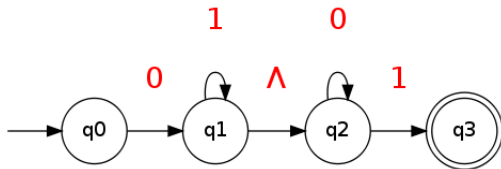
- Hvad er  $\delta^*(q_0, 01)$  for denne NFA- $\Lambda$ ?



- $\delta^*(q_0, \Lambda) = \Lambda(q_0) = \{q_0\}$
- $\delta^*(q_0, \Lambda \cdot 0) = \Lambda(\bigcup_{r \in \delta^*(q_0, \Lambda)} \delta(r, 0)) = \{q_1, q_2\}$
- $\delta^*(q_0, \Lambda \cdot 0 \cdot 1) = \Lambda(\bigcup_{r \in \delta^*(q_0, \Lambda \cdot 0)} \delta(r, 1)) = \Lambda(\{q_1, q_2\} \cup \{q_3\}) = \{q_1, q_2, q_3\}$
- d.v.s. strengen 01 bliver accepteret af automaten.

## Quiz

- Hvad er  $\delta^*(q_0, 01)$  for denne NFA- $\Lambda$ ?



- $\delta^*(q_0, \Lambda) = \Lambda(q_0) = \{q_0\}$
- $\delta^*(q_0, \Lambda \cdot 0) = \Lambda(\bigcup_{r \in \delta^*(q_0, \Lambda)} \delta(r, 0)) = \{q_1, q_2\}$
- $\delta^*(q_0, \Lambda \cdot 0 \cdot 1) = \Lambda(\bigcup_{r \in \delta^*(q_0, \Lambda \cdot 0)} \delta(r, 1)) = \Lambda(\{q_1, q_2\} \cup \{q_3\}) = \{q_1, q_2, q_3\}$
- d.v.s. strengen 01 bliver accepteret af automaten.

# Enhver NFA kan oversættes til en NFA- $\Lambda$

- Med den grafiske repræsentation er det trivielt

- Med de formelle definitioner:

Givet en NFA  $M = (Q, \Sigma, q_0, A, \delta_M)$ ,

definer en NFA- $\Lambda$   $N = (Q, \Sigma, q_0, A, \delta_N)$  hvor  $\delta_N(q, a) = \delta_M(q, a)$  for alle  $q \in Q$  og  $a \in \Sigma$   $\delta_N(q, \Lambda) = A$  for alle  $q \in Q$

Bevis for at  $L(N) = L(M)$ : induktion...



# Enhver NFA kan oversættes til en NFA- $\Lambda$

- Med den grafiske repræsentation er det trivielt
- Med de formelle definitioner:

Givet en NFA  $M = (Q, \Sigma, q_0, A, \delta_M)$ ,

definer en NFA- $\Lambda$   $N = (Q, \Sigma, q_0, A, \delta_N)$  hvor  $\delta_N(q, a) = \delta_M(q, a)$  for alle  $q \in Q$  og  $a \in \Sigma$   $\delta_N(q, \Lambda) = \emptyset$  for alle  $q \in Q$

Bevis for at  $L(N) = L(M)$ : induktion...

# Enhver NFA- $\Lambda$ kan oversættes til en NFA ( $\Lambda$ -eliminering)

- Givet en NFA- $\Lambda$   $M = (Q, \Sigma, q_0, A, \delta)$ ,
- definer en NFA  $M_1 = (Q, \Sigma, q_0, A_1, \delta_1)$  ved
- $\delta_1(q, a) = \delta^*(q, a)$
- $A_1 = \begin{cases} A \cup \{q_0\} & \text{hvis } \Lambda(\{q_0\}) \cap A \neq \emptyset \\ A & \text{ellers} \end{cases}$
- Der gælder nu:  $L(M_1) = L(M)$

# Enhver NFA- $\Lambda$ kan oversættes til en NFA ( $\Lambda$ -eliminering)

- Givet en NFA- $\Lambda$   $M = (Q, \Sigma, q_0, A, \delta)$ ,
- definer en NFA  $M_1 = (Q, \Sigma, q_0, A_1, \delta_1)$  ved
- $\delta_1(q, a) = \delta^*(q, a)$
- $A_1 = \begin{cases} A \cup \{q_0\} & \text{hvis } \Lambda(\{q_0\}) \cap A \neq \emptyset \\ A & \text{ellers} \end{cases}$
- Der gælder nu:  $L(M_1) = L(M)$

# Enhver NFA- $\Lambda$ kan oversættes til en NFA ( $\Lambda$ -eliminering)

- Givet en NFA- $\Lambda$   $M = (Q, \Sigma, q_0, A, \delta)$ ,
- definer en NFA  $M_1 = (Q, \Sigma, q_0, A_1, \delta_1)$  ved
- $\delta_1(q, a) = \delta^*(q, a)$
- $A_1 = \begin{cases} A \cup \{q_0\} & \text{hvis } \Lambda(\{q_0\}) \cap A \neq \emptyset \\ A & \text{ellers} \end{cases}$
- Der gælder nu:  $L(M_1) = L(M)$

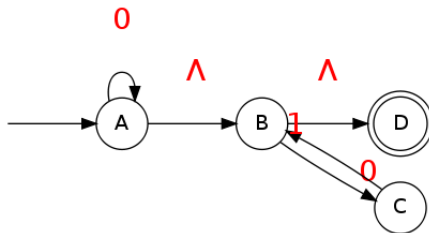
# Enhver NFA- $\Lambda$ kan oversættes til en NFA ( $\Lambda$ -eliminering)

- Givet en NFA- $\Lambda$   $M = (Q, \Sigma, q_0, A, \delta)$ ,
- definer en NFA  $M_1 = (Q, \Sigma, q_0, A_1, \delta_1)$  ved
- $\delta_1(q, a) = \delta^*(q, a)$
- $A_1 = \begin{cases} A \cup \{q_0\} & \text{hvis } \Lambda(\{q_0\}) \cap A \neq \emptyset \\ A & \text{ellers} \end{cases}$
- Der gælder nu:  $L(M_1) = L(M)$

# Enhver NFA- $\Lambda$ kan oversættes til en NFA ( $\Lambda$ -eliminering)

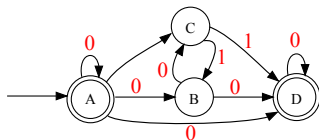
- Givet en NFA- $\Lambda$   $M = (Q, \Sigma, q_0, A, \delta)$ ,
- definer en NFA  $M_1 = (Q, \Sigma, q_0, A_1, \delta_1)$  ved
- $\delta_1(q, a) = \delta^*(q, a)$
- $A_1 = \begin{cases} A \cup \{q_0\} & \text{hvis } \Lambda(\{q_0\}) \cap A \neq \emptyset \\ A & \text{ellers} \end{cases}$
- Der gælder nu:  $L(M_1) = L(M)$

# Eksempel

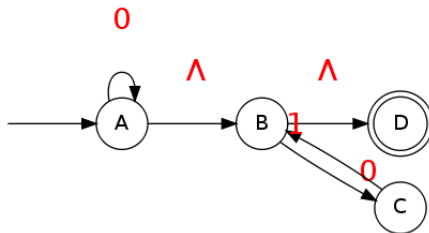


- NFA- $\Lambda$ :
- Find  $\delta^*(q, a)$  for alle  $q \in Q$  og  $a \in \Sigma$
- Se om  $\Lambda(\{q_0\}) \cap A \neq \emptyset$

$q$	$\delta(q, \Lambda)$	$\delta(q, 0)$	$\delta(q, 1)$	$\delta^*(q, 0)$	$\delta^*(q, 1)$
A	$\{B\}$	$\{A\}$	$\{\}$	$\{A, B, C, D\}$	$\{\}$
B	$\{D\}$	$\{C\}$	$\{\}$	$\{C, D\}$	$\{\}$
C	$\{\}$	$\{\}$	$\{B\}$	$\{\}$	$\{B, D\}$
D	$\{\}$	$\{D\}$	$\{\}$	$\{D\}$	$\{\}$

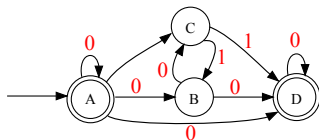


# Eksempel



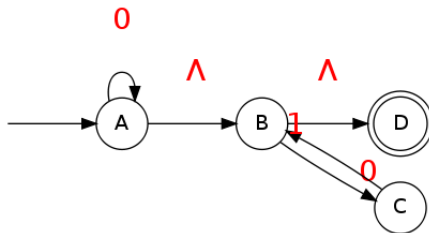
- NFA- $\Lambda$ :
- Find  $\delta^*(q, a)$  for alle  $q \in Q$  og  $a \in \Sigma$
- Se om  $\Lambda(\{q_0\}) \cap A \neq \emptyset$

$q$	$\delta(q, \Lambda)$	$\delta(q, 0)$	$\delta(q, 1)$	$\delta^*(q, 0)$	$\delta^*(q, 1)$
A	$\{B\}$	$\{A\}$	$\{\}$	$\{A, B, C, D\}$	$\{\}$
B	$\{D\}$	$\{C\}$	$\{\}$	$\{C, D\}$	$\{\}$
C	$\{\}$	$\{\}$	$\{B\}$	$\{\}$	$\{B, D\}$
D	$\{\}$	$\{D\}$	$\{\}$	$\{D\}$	$\{D\}$



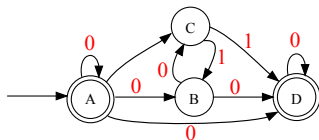


# Eksempel



- NFA- $\Lambda$ :
- Find  $\delta^*(q, a)$  for alle  $q \in Q$  og  $a \in \Sigma$
- Se om  $\Lambda(\{q_0\}) \cap A \neq \emptyset$

$q$	$\delta(q, \Lambda)$	$\delta(q, 0)$	$\delta(q, 1)$	$\delta^*(q, 0)$	$\delta^*(q, 1)$
A	$\{B\}$	$\{A\}$	$\{\}$	$\{A, B, C, D\}$	$\{\}$
B	$\{D\}$	$\{C\}$	$\{\}$	$\{C, D\}$	$\{\}$
C	$\{\}$	$\{\}$	$\{B\}$	$\{\}$	$\{B, D\}$
D	$\{\}$	$\{D\}$	$\{\}$	$\{D\}$	$\{D\}$



## Bevis for korrekthed af $\Lambda$ -eliminering

- Vi skal vise:  $\forall x \in \Sigma^* : x \in L(M_1) \Leftrightarrow x \in L(M) \text{ } x = \Lambda : \text{brug definition af } A_1 \text{ og } \Lambda\text{-lukning} \dots$
- $x = a \in \Sigma : \text{brug } A_1, \delta^*(q, a) = \delta_1(q, a) \text{ og } \Lambda\text{-lukning} \dots$
- $x = ya, y \neq \Lambda$ : Induktionshypotese:  
 $\forall y \in \Sigma^*, y \neq \Lambda : \delta^*(q_0, y) = \delta_1(q_0, y) \Rightarrow \delta^*(q_0, x) = \delta_1^*(q_0, x)$   
 $\dots$
- – se bogen Th. 4.2 p. 141.

## Bevis for korrekthed af $\Lambda$ -eliminering

- Vi skal vise:  $\forall x \in \Sigma^* : x \in L(M_1) \Leftrightarrow x \in L(M) \text{ } x = \Lambda : \text{brug definition af } A_1 \text{ og } \Lambda\text{-lukning} \dots$
- $x = a \in \Sigma : \text{brug } A_1, \delta^*(q, a) = \delta_1(q, a) \text{ og } \Lambda\text{-lukning} \dots$
- $x = ya, y \neq \Lambda$ : Induktionshypotese:  
 $\forall y \in \Sigma^*, y \neq \Lambda : \delta^*(q_0, y) = \delta_1(q_0, y) \Rightarrow \delta^*(q_0, x) = \delta_1^*(q_0, x)$   
 $\dots$
- – se bogen Th. 4.2 p. 141.

## Bevis for korrekthed af $\Lambda$ -eliminering

- Vi skal vise:  $\forall x \in \Sigma^* : x \in L(M_1) \Leftrightarrow x \in L(M) \text{ } x = \Lambda : \text{brug definition af } A_1 \text{ og } \Lambda\text{-lukning...}$
- $x = a \in \Sigma : \text{brug } A_1, \delta^*(q, a) = \delta_1(q, a) \text{ og } \Lambda\text{-lukning...}$
- $x = ya, y \neq \Lambda$ : Induktionshypotese:  
 $\forall y \in \Sigma^*, y \neq \Lambda : \delta^*(q_0, y) = \delta_1(q_0, y) \Rightarrow \delta^*(q_0, x) = \delta_1^*(q_0, x)$   
 ...
- – se bogen Th. 4.2 p. 141.

## Bevis for korrekthed af $\Lambda$ -eliminering

- Vi skal vise:  $\forall x \in \Sigma^* : x \in L(M_1) \Leftrightarrow x \in L(M) \text{ } x = \Lambda : \text{brug definition af } A_1 \text{ og } \Lambda\text{-lukning...}$
- $x = a \in \Sigma : \text{brug } A_1, \delta^*(q, a) = \delta_1(q, a) \text{ og } \Lambda\text{-lukning...}$
- $x = ya, y \neq \Lambda$ : Induktionshypotese:  
 $\forall y \in \Sigma^*, y \neq \Lambda : \delta^*(q_0, y) = \delta_1(q_0, y) \Rightarrow \delta^*(q_0, x) = \delta_1^*(q_0, x)$   
 ...
- – se bogen Th. 4.2 p. 141.

# Øvelser

- [Martin] Opg. 4.13 (p.159) Kør strenge på en NFA- $\Lambda$
- [Martin] Opg. 4.28 (e) Brug algoritmen til  $\Lambda$ -eliminering

# Plan

Nondeterministiske automater

Determinisering

NFA- $\Lambda$ 'er

Kleenes sætning

Kleenes sætning del 1

Kleenes sætning del 2

Frokost

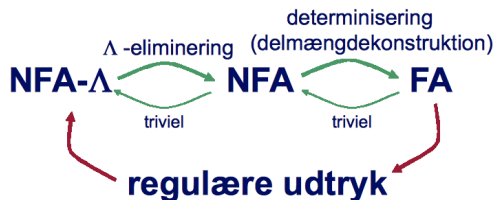
Minimering

Myhill Nerode

Java projekt

# Status

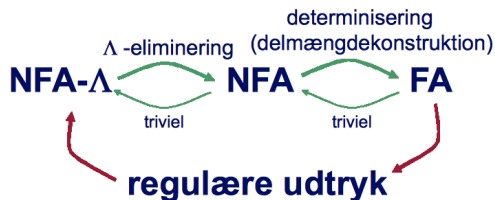
- Vi har defineret 4 formalismer:
  - regulære udtryk
  - FA,
  - NFA,
  - NFA- $\Lambda$
- og er ved konstruktivt at bevise ækvivalens i udtrykskraft





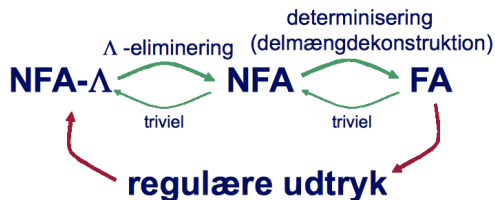
# Status

- Vi har defineret 4 formalismer:
  - regulære udtryk
  - FA,
  - NFA,
  - NFA- $\Lambda$
- og er ved konstruktivt at bevise ækvivalens i udtrykskraft



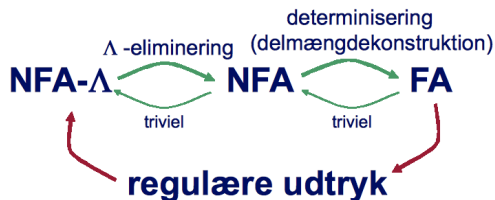
# Status

- Vi har defineret 4 formalismer:
  - regulære udtryk
  - FA,
  - NFA,
  - NFA- $\Lambda$
- og er ved konstruktivt at bevise ækvivalens i udtrykskraft



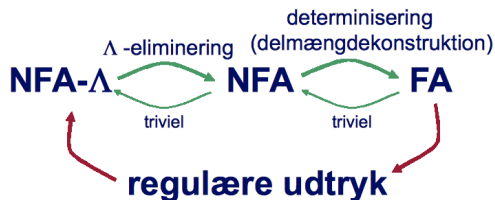
# Status

- Vi har defineret 4 formalismer:
  - regulære udtryk
  - FA,
  - NFA,
  - NFA- $\Lambda$
- og er ved konstruktivt at bevise ækvivalens i udtrykskraft



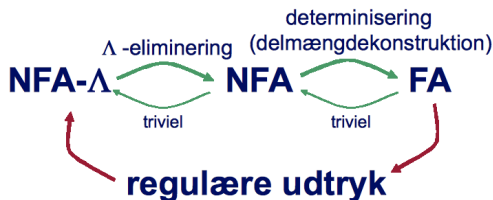
# Status

- Vi har defineret 4 formalismer:
  - regulære udtryk
  - FA,
  - NFA,
  - NFA- $\Lambda$
- og er ved konstruktivt at bevise ækvivalens i udtrykskraft



# Status

- Vi har defineret 4 formalismer:
  - regulære udtryk
  - FA,
  - NFA,
  - NFA- $\Lambda$
- og er ved konstruktivt at bevise ækvivalens i udtrykskraft



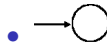
# Ethvert regulært udtryk kan oversættes til en NFA- $\Lambda$

(Kleenes sætning, del 1)

- Bevis: Induktion i strukturen af det regulære udtryk  $r$ .
- Vis konstruktivt for hvert tilfælde hvordan man kan lave den korrekte NFA- $\Lambda$

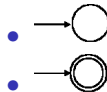
# Basis

- $r = \emptyset$
- $r = \Lambda$
- $r = a$ , hvor  $a \in \Sigma$



# Basis

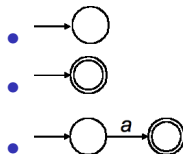
- $r = \emptyset$
- $r = \Lambda$
- $r = a$ , hvor  $a \in \Sigma$





# Basis

- $r = \emptyset$
- $r = \Lambda$
- $r = a$ , hvor  $a \in \Sigma$



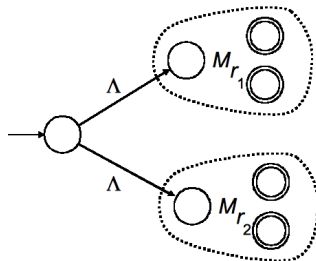
# Induktionsskridt

- For alle deludtryk  $s$  af  $r$  kan vi udnytte induktionshypotesen:
- Der eksisterer en NFA- $\Lambda$   $M_s$  hvor  $L(M_s) = L(s)$

# Induktionsskridt

- For alle deludtryk  $s$  af  $r$  kan vi udnytte induktionshypotesen:
- Der eksisterer en NFA- $\Lambda$   $M_s$  hvor  $L(M_s) = L(s)$

$$r = r_1 + r_2$$



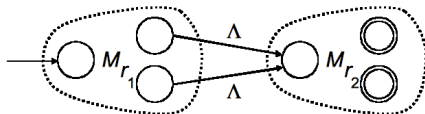
# Induktionsskridt (part 2)

- $r = r_1 \cdot r_2$

- $r = r_1^*$

# Induktionsskridt (part 2)

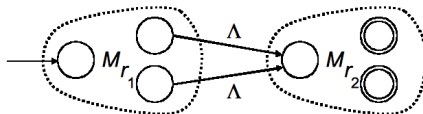
- $r = r_1 \cdot r_2$



- $r = r_1^*$

# Induktionsskridt (part 2)

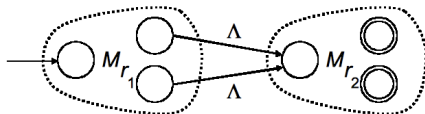
- $r = r_1 \cdot r_2$



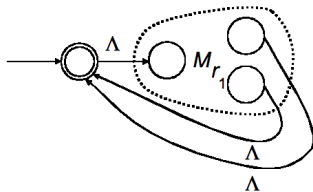
- $r = r_1^*$

# Induktionsskridt (part 2)

- $r = r_1 \cdot r_2$



- $r = r_1^*$



# Formel beskrivelse og bevis for korrekthed

Se beviset i bogen: p. 146



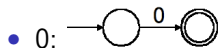
## Eksempel (1/3)

Konstruer en NFA- $\Lambda$  for det regulære udtryk  $(00 + 1)^*(10)^*$

- 0:
- 1:
- 00:
- 10:

# Eksempel (1/3)

Konstruer en NFA- $\Lambda$  for det regulære udtryk  $(00 + 1)^*(10)^*$



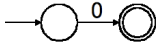
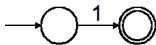
- 1:

- 00:

- 10:

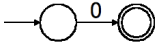
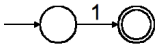

# Eksempel (1/3)

Konstruer en NFA- $\Lambda$  for det regulære udtryk  $(00 + 1)(10)^*$

- 0: 
- 1: 
- 00:
- 10:

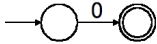
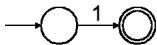


# Eksempel (1/3)

Konstruer en NFA- $\Lambda$  for det regulære udtryk  $(00 + 1)(10)^*$

- 0: 
- 1: 
- 00: 
- 10:

# Eksempel (1/3)

Konstruer en NFA- $\Lambda$  for det regulære udtryk  $(00 + 1)^*(10)^*$

- 0: 
- 1: 
- 00: 
- 10: 

## Eksempel (2/3)

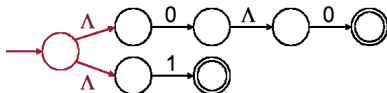
Konstruer en NFA- $\Lambda$  for det regulære udtryk  $(00 + 1)^*(10)^*$

- $00 + 1$ :
- $(00 + 1)^*$ :

## Eksempel (2/3)

Konstruer en NFA- $\Lambda$  for det regulære udtryk  $(00 + 1)^*(10)^*$

- $00 + 1$ :

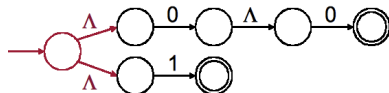


- $(00 + 1)^*$ :

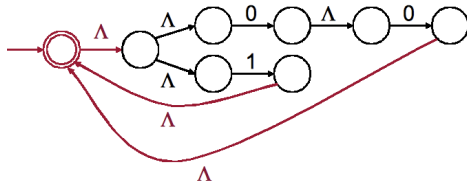
## Eksempel (2/3)

Konstruer en NFA- $\Lambda$  for det regulære udtryk  $(00 + 1)^*(10)^*$

- $00 + 1$ :



- $(00 + 1)^*$ :





## Eksempel (3/3)

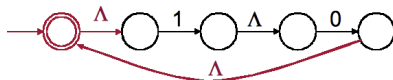
Konstruer en NFA- $\Lambda$  for det regulære udtryk  $(00 + 1)^*(10)^*$

- $(10)^*$ :
- $(00 + 1)^*(10)^*$ :

## Eksempel (3/3)

Konstruer en NFA- $\Lambda$  for det regulære udtryk  $(00 + 1)^*(10)^*$

- $(10)^*$ :

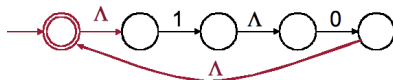


- $(00 + 1)^*(10)^*$ :

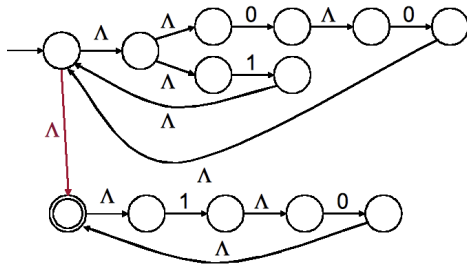
# Eksempel (3/3)

Konstruer en NFA- $\Lambda$  for det regulære udtryk  $(00 + 1)^*(10)^*$

- $(10)^*$ :



- $(00 + 1)^*(10)^*$ :



# Øvelser

- [Martin] Opg. 4.35 (a) (p. 163)  
Udfør algoritmen for konstruktion af NFA- $\Lambda$  fra regulært udtryk.

# Enhver FA kan oversættes til et regulært udtryk

- Kleene's sætning del 2.
- Vi laver bevis med induktion naturligvis, men induktion i hvad?

## Fra FA til regulært udtryk

- For en FA  $M = (Q, \Sigma, q_0, A, \delta)$  er  $L(M)$  defineret som  $L(M) = \{x \in \Sigma^* \mid \delta^*(q_0, x) \in A\}$
- Da  $A$  er endelig kan  $L(M)$  udtrykkes som en endelig forening af sprog på form  $L(p, q) = \{x \in \Sigma^* \mid \delta^*(p, x) = q\}$
- Vi vil vise at hvert af disse sprog kan oversættes til et regulært udtryk,  $r(p, q)$ , og derefter kombinere disse med “+”

## Fra FA til regulært udtryk

- For en FA  $M = (Q, \Sigma, q_0, A, \delta)$  er  $L(M)$  defineret som
$$L(M) = \{x \in \Sigma^* \mid \delta^*(q_0, x) \in A\}$$
- Da  $A$  er endelig kan  $L(M)$  udtrykkes som en endelig forening af sprog på form  $L(p, q) = \{x \in \Sigma^* \mid \delta^*(p, x) = q\}$
- Vi vil vise at hvert af disse sprog kan oversættes til et regulært udtryk,  $r(p, q)$ , og derefter kombinere disse med “+”

## Fra FA til regulært udtryk

- For en FA  $M = (Q, \Sigma, q_0, A, \delta)$  er  $L(M)$  defineret som  $L(M) = \{x \in \Sigma^* \mid \delta^*(q_0, x) \in A\}$
- Da  $A$  er endelig kan  $L(M)$  udtrykkes som en endelig forening af sprog på form  $L(p, q) = \{x \in \Sigma^* \mid \delta^*(p, x) = q\}$
- Vi vil vise at hvert af disse sprog kan oversættes til et regulært udtryk,  $r(p, q)$ , og derefter kombinere disse med “+”

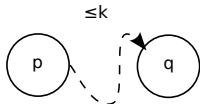


# Induktion i antal tilstande

- Antag tilstandene i  $M$  er nummereret  $1, \dots, |Q|$
- Definer  $L(p, q, k)$  hvor  $p, q \in Q$  og  $k \in \{1..|Q|\}$  som:  
Mængden af strenge, der fører fra  $p$  til  $q$  og kun går gennem tilstande med nummer  $\leq k$  (fraregnet endepunkterne)
- dvs.  $L(p, q) = L(p, q, |Q|)$
- Vi vil vise ved induktion i  $k$  at  $L(p, q, k)$  svarer til et regulært udtryk,  $r(p, q, k)$
- dvs. vælg  $r(p, q) = r(p, q, |Q|)$

# Induktion i antal tilstande

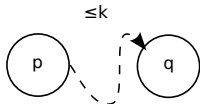
- Antag tilstandene i  $M$  er nummereret  $1, \dots, |Q|$
- Definer  $L(p, q, k)$  hvor  $p, q \in Q$  og  $k \in \{1..|Q|\}$  som:  
Mængden af strenge, der fører fra  $p$  til  $q$  og kun går gennem tilstande med nummer  $\leq k$  (fraregnet endepunkterne)



- dvs.  $L(p, q) = L(p, q, |Q|)$
- Vi vil vise ved induktion i  $k$  at  $L(p, q, k)$  svarer til et regulært udtryk,  $r(p, q, k)$
- dvs. vælg  $r(p, q) = r(p, q, |Q|)$

# Induktion i antal tilstande

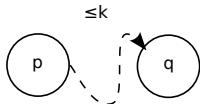
- Antag tilstandene i  $M$  er nummereret  $1, \dots, |Q|$
- Definer  $L(p, q, k)$  hvor  $p, q \in Q$  og  $k \in \{1..|Q|\}$  som:  
Mængden af strenge, der fører fra  $p$  til  $q$  og kun går gennem tilstande med nummer  $\leq k$  (fraregnet endepunkterne)



- dvs.  $L(p, q) = L(p, q, |Q|)$
- Vi vil vise ved induktion i  $k$  at  $L(p, q, k)$  svarer til et regulært udtryk,  $r(p, q, k)$
- dvs. vælg  $r(p, q) = r(p, q, |Q|)$

# Induktion i antal tilstande

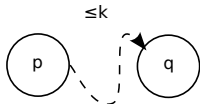
- Antag tilstandene i  $M$  er nummereret  $1, \dots, |Q|$
- Definer  $L(p, q, k)$  hvor  $p, q \in Q$  og  $k \in \{1..|Q|\}$  som:  
Mængden af strenge, der fører fra  $p$  til  $q$  og kun går gennem tilstande med nummer  $\leq k$  (fraregnet endepunkterne)



- dvs.  $L(p, q) = L(p, q, |Q|)$
- Vi vil vise ved induktion i  $k$  at  $L(p, q, k)$  svarer til et regulært udtryk,  $r(p, q, k)$
- dvs. vælg  $r(p, q) = r(p, q, |Q|)$

# Induktion i antal tilstande

- Antag tilstandene i  $M$  er nummereret  $1, \dots, |Q|$
- Definer  $L(p, q, k)$  hvor  $p, q \in Q$  og  $k \in \{1..|Q|\}$  som:  
Mængden af strenge, der fører fra  $p$  til  $q$  og kun går gennem tilstande med nummer  $\leq k$  (fraregnet endepunkterne)



- dvs.  $L(p, q) = L(p, q, |Q|)$
- Vi vil vise ved induktion i  $k$  at  $L(p, q, k)$  svarer til et regulært udtryk,  $r(p, q, k)$
- dvs. vælg  $r(p, q) = r(p, q, |Q|)$

# Basis

$k = 0$

- $L(p, q, 0)$  er mængden af strenge, der fører fra  $p$  til  $q$  uden at gå gennem nogen tilstande (fraregnet endepunkterne)
- hvis  $p \neq q$ :  $L(p, q, 0) = \{a \in \Sigma \mid \delta(p, a) = q\}$
- hvis  $p = q$ :  $L(p, q, 0) = \{a \in \Sigma \mid \delta(p, a) = p\} \cup \{\Lambda\}$
- dvs. vi kan altid finde et regulært udtryk  $r(p, q, 0)$  for  $L(p, q, 0)$

# Basis

$k = 0$

- $L(p, q, 0)$  er mængden af strenge, der fører fra  $p$  til  $q$  uden at gå gennem nogen tilstande (fraregnet endepunkterne)
- hvis  $p \neq q$ :  $L(p, q, 0) = \{a \in \Sigma \mid \delta(p, a) = q\}$
- hvis  $p = q$ :  $L(p, q, 0) = \{a \in \Sigma \mid \delta(p, a) = p\} \cup \{\Lambda\}$
- dvs. vi kan altid finde et regulært udtryk  $r(p, q, 0)$  for  $L(p, q, 0)$

# Basis

$k = 0$

- $L(p, q, 0)$  er mængden af strenge, der fører fra  $p$  til  $q$  uden at gå gennem nogen tilstande (fraregnet endepunkterne)
- hvis  $p \neq q$ :  $L(p, q, 0) = \{a \in \Sigma \mid \delta(p, a) = q\}$
- hvis  $p = q$ :  $L(p, q, 0) = \{a \in \Sigma \mid \delta(p, a) = p\} \cup \{\Lambda\}$
- dvs. vi kan altid finde et regulært udtryk  $r(p, q, 0)$  for  $L(p, q, 0)$



# Basis

$k = 0$

- $L(p, q, 0)$  er mængden af strenge, der fører fra  $p$  til  $q$  uden at gå gennem nogen tilstande (fraregnet endepunkterne)
- hvis  $p \neq q$ :  $L(p, q, 0) = \{a \in \Sigma \mid \delta(p, a) = q\}$
- hvis  $p = q$ :  $L(p, q, 0) = \{a \in \Sigma \mid \delta(p, a) = p\} \cup \{\Lambda\}$
- dvs. vi kan altid finde et regulært udtryk  $r(p, q, 0)$  for  $L(p, q, 0)$

# Induktionsskridt

$k + 1$

- $L(p, q, k + 1)$  er mængden af strenge, der fører fra  $p$  til  $q$  og kun går gennem tilstande med nummer  $\leq k + 1$
- To tilfælde:
  - Strengene der ikke går gennem tilstand  $k + 1$ :  $L(p, q, k)$
  - Strengene der går gennem tilstand  $k + 1$ :  
 $L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$
- dvs.  

$$L(p, q, k + 1) = L(p, q, k) \cup L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$$
- som vha. induktionshypotesen svarer til et regulært udtryk  

$$r(p, q, k + 1) = r(p, q, k) + r(p, k + 1, k)r(k + 1, k + 1, k)^*r(k + 1, q, k)$$

# Induktionsskridt

$k + 1$

- $L(p, q, k + 1)$  er mængden af strenge, der fører fra  $p$  til  $q$  og kun går gennem tilstande med nummer  $\leq k + 1$
- To tilfælde:
  - Strenge der ikke går gennem tilstand  $k + 1$ :  $L(p, q, k)$
  - Strenge der går gennem tilstand  $k + 1$ :  
 $L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$
- dvs.  
 $L(p, q, k + 1) = L(p, q, k) \cup L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$
- som vha. induktionshypotesen svarer til et regulært udtryk  
 $r(p, q, k + 1) = r(p, q, k) + r(p, k + 1, k)r(k + 1, k + 1, k)^*r(k + 1, q, k)$

# Induktionsskridt

$k + 1$

- $L(p, q, k + 1)$  er mængden af strenge, der fører fra  $p$  til  $q$  og kun går gennem tilstande med nummer  $\leq k + 1$
- To tilfælde:
  - Strengene der ikke går gennem tilstand  $k + 1$ :  $L(p, q, k)$
  - Strengene der går gennem tilstand  $k + 1$ :  
 $L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$
- dvs.  
 $L(p, q, k + 1) = L(p, q, k) \cup L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$
- som vha. induktionshypotesen svarer til et regulært udtryk  
 $r(p, q, k + 1) = r(p, q, k) + r(p, k + 1, k)r(k + 1, k + 1, k)^*r(k + 1, q, k)$

# Induktionsskridt

$k + 1$

- $L(p, q, k + 1)$  er mængden af strenge, der fører fra  $p$  til  $q$  og kun går gennem tilstande med nummer  $\leq k + 1$
- To tilfælde:
  - Strenge der ikke går gennem tilstand  $k + 1$ :  $L(p, q, k)$
  - Strenge der går gennem tilstand  $k + 1$ :  
 $L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$
- dvs.  
 $L(p, q, k + 1) = L(p, q, k) \cup L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$
- som vha. induktionshypotesen svarer til et regulært udtryk  
 $r(p, q, k + 1) = r(p, q, k) + r(p, k + 1, k)r(k + 1, k + 1, k)^*r(k + 1, q, k)$

# Induktionsskridt

$k + 1$

- $L(p, q, k + 1)$  er mængden af strenge, der fører fra  $p$  til  $q$  og kun går gennem tilstande med nummer  $\leq k + 1$
- To tilfælde:
  - Strengene der ikke går gennem tilstand  $k + 1$ :  $L(p, q, k)$
  - Strengene der går gennem tilstand  $k + 1$ :  
 $L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$
- dvs.  

$$L(p, q, k + 1) = L(p, q, k) \cup L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$$
- som vha. induktionshypotesen svarer til et regulært udtryk  

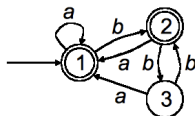
$$r(p, q, k + 1) = r(p, q, k) + r(p, k + 1, k)r(k + 1, k + 1, k)^*r(k + 1, q, k)$$

# Induktionsskridt

$k + 1$

- $L(p, q, k + 1)$  er mængden af strenge, der fører fra  $p$  til  $q$  og kun går gennem tilstande med nummer  $\leq k + 1$
- To tilfælde:
  - Strenge der ikke går gennem tilstand  $k + 1$ :  $L(p, q, k)$
  - Strenge der går gennem tilstand  $k + 1$ :  
 $L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$
- dvs.  
 $L(p, q, k + 1) = L(p, q, k) \cup L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$
- som vha. induktionshypotesen svarer til et regulært udtryk  
 $r(p, q, k + 1) = r(p, q, k) + r(p, k + 1, k)r(k + 1, k + 1, k)^*r(k + 1, q, k)$

# Eksempel

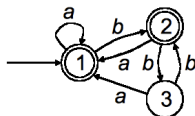


Oversæt denne FA til et regulært udtryk

- $r = r(1, 1, 3) + r(1, 2, 3)$
- $r(1, 1, 3) = r(1, 1, 2) + r(1, 3, 2)r(3, 3, 2)^*r(3, 1, 2)$
- $r(1, 1, 2) = r(1, 1, 1) + r(1, 2, 1)r(2, 2, 1)^*r(2, 1, 1)$
- $r(1, 1, 1) = r(1, 1, 0) + r(1, 1, 0)r(1, 1, 0)^*r(1, 1, 0)$
- $r(1, 1, 0) = a + \Lambda \dots$
- Heldigvis kan vi sætte en computer til det!



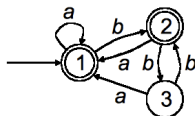
# Eksempel



Oversæt denne FA til et regulært udtryk

- $r = r(1, 1, 3) + r(1, 2, 3)$
- $r(1, 1, 3) = r(1, 1, 2) + r(1, 3, 2)r(3, 3, 2)^*r(3, 1, 2)$
- $r(1, 1, 2) = r(1, 1, 1) + r(1, 2, 1)r(2, 2, 1)^*r(2, 1, 1)$
- $r(1, 1, 1) = r(1, 1, 0) + r(1, 1, 0)r(1, 1, 0)^*r(1, 1, 0)$
- $r(1, 1, 0) = a + \Lambda \dots$
- Heldigvis kan vi sætte en computer til det!

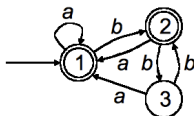
# Eksempel



Oversæt denne FA til et regulært udtryk

- $r = r(1, 1, 3) + r(1, 2, 3)$
- $r(1, 1, 3) = r(1, 1, 2) + r(1, 3, 2)r(3, 3, 2)^*r(3, 1, 2)$
- $r(1, 1, 2) = r(1, 1, 1) + r(1, 2, 1)r(2, 2, 1)^*r(2, 1, 1)$
- $r(1, 1, 1) = r(1, 1, 0) + r(1, 1, 0)r(1, 1, 0)^*r(1, 1, 0)$
- $r(1, 1, 0) = a + \Lambda \dots$
- Heldigvis kan vi sætte en computer til det!

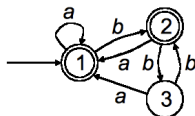
# Eksempel



Oversæt denne FA til et regulært udtryk

- $r = r(1, 1, 3) + r(1, 2, 3)$
- $r(1, 1, 3) = r(1, 1, 2) + r(1, 3, 2)r(3, 3, 2)^*r(3, 1, 2)$
- $r(1, 1, 2) = r(1, 1, 1) + r(1, 2, 1)r(2, 2, 1)^*r(2, 1, 1)$
- $r(1, 1, 1) = r(1, 1, 0) + r(1, 1, 0)r(1, 1, 0)^*r(1, 1, 0)$
- $r(1, 1, 0) = a + \Lambda \dots$
- Heldigvis kan vi sætte en computer til det!

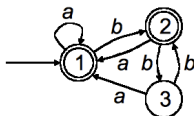
# Eksempel



Oversæt denne FA til et regulært udtryk

- $r = r(1, 1, 3) + r(1, 2, 3)$
- $r(1, 1, 3) = r(1, 1, 2) + r(1, 3, 2)r(3, 3, 2)^*r(3, 1, 2)$
- $r(1, 1, 2) = r(1, 1, 1) + r(1, 2, 1)r(2, 2, 1)^*r(2, 1, 1)$
- $r(1, 1, 1) = r(1, 1, 0) + r(1, 1, 0)r(1, 1, 0)^*r(1, 1, 0)$
- $r(1, 1, 0) = a + \Lambda \dots$
- Heldigvis kan vi sætte en computer til det!

# Eksempel



Oversæt denne FA til et regulært udtryk

- $r = r(1, 1, 3) + r(1, 2, 3)$
- $r(1, 1, 3) = r(1, 1, 2) + r(1, 3, 2)r(3, 3, 2)^*r(3, 1, 2)$
- $r(1, 1, 2) = r(1, 1, 1) + r(1, 2, 1)r(2, 2, 1)^*r(2, 1, 1)$
- $r(1, 1, 1) = r(1, 1, 0) + r(1, 1, 0)r(1, 1, 0)^*r(1, 1, 0)$
- $r(1, 1, 0) = a + \Lambda \dots$
- Heldigvis kan vi sætte en computer til det!

## Eksempel fortsat

- (Hvis programmet ikke simplificerer undervejs...)

## Eksempel fortsat

- (Hvis programmet ikke simplificerer undervejs...)

# Øvelser

- [Martin] Opg. 4.38(b) (p. 164)  
Brug algoritmen fra Kleenes sætning del 2.



# Plan

Nondeterministiske automater

Determinisering

NFA- $\Lambda$ 'er

Kleenes sætning

Kleenes sætning del 1

Kleenes sætning del 2

Frokost

Minimering

Myhill Nerode

Java projekt



# Resume

- Regulære udtryk, FA'er, NFA'er og NFA- $\Lambda$ 'er svarer alle til klassen af regulære sprog
- Algoritmer fra de konstruktive beviser:
- determinisering (delmængdekonstruktionen)
- $\Lambda$ -eliminering
- regulært udtryk  $\rightarrow$  NFA- $\Lambda$
- FA  $\rightarrow$  regulære udtryk (primært et teoretisk resultat)

# Plan

Nondeterministiske automater

Determinisering

NFA- $\Lambda$ 'er

Kleenes sætning

Kleenes sætning del 1

Kleenes sætning del 2

Frokost

Minimering

Myhill Nerode

Java projekt

# Karakteristik af de regulære sprog

Et sprog er regulært hviss (hvis og kun hvis)

- L beskrives af et regulært udtryk
- L genkendes af en FA/NFA/NFA- $\Lambda$
- Der ikke findes uendeligt mange strenge der er parvist skelnelige mht. L

# Karakteristik af de regulære sprog

Et sprog er regulært hviss (hvis og kun hvis)

- L beskrives af et regulært udtryk
- L genkendes af en FA/NFA/NFA- $\Lambda$
- Der ikke findes uendeligt mange strenge der er parvist skelnelige mht. L

# Karakteristik af de regulære sprog

Et sprog er regulært hviss (hvis og kun hvis)

- L beskrives af et regulært udtryk
- L genkendes af en FA/NFA/NFA- $\Lambda$
- Der ikke findes uendeligt mange strenge der er parvist skelnelige mht. L

## Skelnelighed (fra 1. seminar)

- $x$  og  $y$  er skelnelige mht.  $L$  hvis
$$\exists z \in \Sigma^* : (xz \in L \wedge yz \notin L) \vee (xz \notin L \wedge yz \in L)$$
- Hvis to skelnelige strenge mht.  $L$  køres på en FA, der accepterer  $L$ , vil de ende i forskellige tilstande
- Intuition bag FA-minimering:
- hvis to strenge er **uskelnelige** mht. FA'ens sprog, er der ingen grund til at den skelner mellem dem!



## Skelnelighed (fra 1. seminar)

- $x$  og  $y$  er skelnelige mht.  $L$  hvis
$$\exists z \in \Sigma^* : (xz \in L \wedge yz \notin L) \vee (xz \notin L \wedge yz \in L)$$
- Hvis to skelnelige strenge mht.  $L$  køres på en FA, der accepterer  $L$ , vil de ende i forskellige tilstande
- Intuition bag FA-minimering:
- hvis to strenge er **uskelnelige** mht. FA'ens sprog, er der ingen grund til at den skelner mellem dem!

## Skelnelighed (fra 1. seminar)

- $x$  og  $y$  er skelnelige mht.  $L$  hvis
$$\exists z \in \Sigma^* : (xz \in L \wedge yz \notin L) \vee (xz \notin L \wedge yz \in L)$$
- Hvis to skelnelige strenge mht.  $L$  køres på en FA, der accepterer  $L$ , vil de ende i forskellige tilstande
- Intuition bag FA-minimering:
- hvis to strenge er **uskelnelige** mht. FA'ens sprog, er der ingen grund til at den skelner mellem dem!

## Skelnelighed (fra 1. seminar)

- $x$  og  $y$  er skelnelige mht.  $L$  hvis
$$\exists z \in \Sigma^* : (xz \in L \wedge yz \notin L) \vee (xz \notin L \wedge yz \in L)$$
- Hvis to skelnelige strenge mht.  $L$  køres på en FA, der accepterer  $L$ , vil de ende i forskellige tilstande
- Intuition bag FA-minimering:
- hvis to strenge er **uskelnelige** mht. FA'ens sprog, er der ingen grund til at den skelner mellem dem!

# Uskelnelighedsrelationen $I_L$

- Definition: Givet et sprog  $L \subseteq \Sigma^*$ , definer relationen  $I_L$  over  $\Sigma^*$  ved:  
 $x I_L y \Leftrightarrow x$  og  $y$  er uskelnelige mht.  $L$

# Relationer

- En (binær) relation  $R$  over en mængde  $A$  er en delmængde af  $A \times A$
- Eksempler:  $\leq$  er en relation over mængden af reelle tal  $I_L$  er en relation over  $\Sigma^*$
- Notation:  $xRy \Leftrightarrow (x, y) \in R$

# Relationer

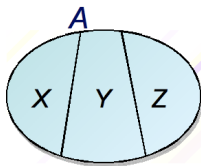
- En (binær) relation  $R$  over en mængde  $A$  er en delmængde af  $A \times A$
- Eksempler:  $\leq$  er en relation over mængden af reelle tal  $I_L$  er en relation over  $\Sigma^*$
- Notation:  $xRy \Leftrightarrow (x, y) \in R$

# Relationer

- En (binær) relation  $R$  over en mængde  $A$  er en delmængde af  $A \times A$
- Eksempler:  $\leq$  er en relation over mængden af reelle tal  $I_L$  er en relation over  $\Sigma^*$
- Notation:  $xRy \Leftrightarrow (x, y) \in R$

# Ækvivalensrelationer

- R er en ækvivalensrelation hvis den er
  - refleksiv ( $\forall x : xRx$ )
  - symmetrisk ( $\forall x, y : xRy \Rightarrow yRx$ )
  - transitiv ( $\forall x, y, z : xRy \wedge yRz \Rightarrow xRz$ )
- En ækvivalensrelation over A definerer en partitionering af A

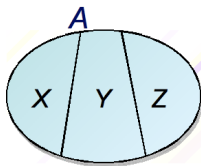


- Notation:  $[x] = \{y | xRy\}$  kaldes ækvivalensklassen for x mht. R



# Ækvivalensrelationer

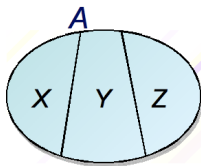
- R er en ækvivalensrelation hvis den er
- refleksiv ( $\forall x : xRx$ )
- symmetrisk ( $\forall x, y : xRy \Rightarrow yRx$ )
- transitiv ( $\forall x, y, z : xRy \wedge yRz \Rightarrow xRz$ )
- En ækvivalensrelation over A definerer en partitionering af A



- Notation:  $[x] = \{y | xRy\}$  kaldes ækvivalensklassen for x mht. R

# Ækvivalensrelationer

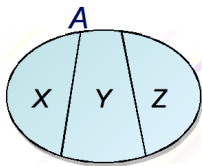
- $R$  er en ækvivalensrelation hvis den er
- refleksiv ( $\forall x : xRx$ )
- symmetrisk ( $\forall x, y : xRy \Rightarrow yRx$ )
- transitiv ( $\forall x, y, z : xRy \wedge yRz \Rightarrow xRz$ )
- En ækvivalensrelation over  $A$  definerer en partitionering af  $A$



- Notation:  $[x] = \{y | xRy\}$  kaldes ækvivalensklassen for  $x$  mht.  $R$

# Ækvivalensrelationer

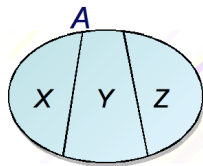
- $R$  er en ækvivalensrelation hvis den er
- refleksiv ( $\forall x : xRx$ )
- symmetrisk ( $\forall x, y : xRy \Rightarrow yRx$ )
- transitiv ( $\forall x, y, z : xRy \wedge yRz \Rightarrow xRz$ )
- En ækvivalensrelation over  $A$  definerer en partitionering af  $A$



- Notation:  $[x] = \{y | xRy\}$  kaldes ækvivalensklassen for  $x$  mht.  $R$

# Ækvivalensrelationer

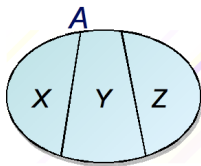
- $R$  er en ækvivalensrelation hvis den er
- refleksiv ( $\forall x : xRx$ )
- symmetrisk ( $\forall x, y : xRy \Rightarrow yRx$ )
- transitiv ( $\forall x, y, z : xRy \wedge yRz \Rightarrow xRz$ )
- En ækvivalensrelation over  $A$  definerer en partitionering af  $A$



- Notation:  $[x] = \{y | xRy\}$  kaldes ækvivalensklassen for  $x$  mht.  $R$

# Ækvivalensrelationer

- $R$  er en ækvivalensrelation hvis den er
- refleksiv ( $\forall x : xRx$ )
- symmetrisk ( $\forall x, y : xRy \Rightarrow yRx$ )
- transitiv ( $\forall x, y, z : xRy \wedge yRz \Rightarrow xRz$ )
- En ækvivalensrelation over  $A$  definerer en partitionering af  $A$



- Notation:  $[x] = \{y | xRy\}$  kaldes ækvivalensklassen for  $x$  mht.  $R$

# Egenskaber ved $I_L$

- $I_L$  er
- refleksiv ( $\forall x : x I_L x$ )
- symmetrisk ( $\forall x, y : x I_L y \Rightarrow y I_L x$ )
- transitiv ( $\forall x, y, z : x I_L y \wedge y I_L z \Rightarrow x I_L z$ )
- dvs.  $I_L$  er en ækvivalensrelation
- $I_L$  partitionerer  $\Sigma^*$
- $[x]$  er mængden af strenge, der er uskelnelige fra  $x$  mht.  $L$

# Egenskaber ved $I_L$

- $I_L$  er
- refleksiv ( $\forall x : x I_L x$ )
- symmetrisk ( $\forall x, y : x I_L y \Rightarrow y I_L x$ )
- transitiv ( $\forall x, y, z : x I_L y \wedge y I_L z \Rightarrow x I_L z$ )
- dvs.  $I_L$  er en ækvivalensrelation
- $I_L$  partitionerer  $\Sigma^*$
- $[x]$  er mængden af strenge, der er uskelnelige fra  $x$  mht.  $L$

# Egenskaber ved $I_L$

- $I_L$  er
- refleksiv ( $\forall x : x I_L x$ )
- symmetrisk ( $\forall x, y : x I_L y \Rightarrow y I_L x$ )
- transitiv ( $\forall x, y, z : x I_L y \wedge y I_L z \Rightarrow x I_L z$ )
- dvs.  $I_L$  er en ækvivalensrelation
- $I_L$  partitionerer  $\Sigma^*$
- $[x]$  er mængden af strenge, der er uskelnelige fra  $x$  mht.  $L$



# Egenskaber ved $I_L$

- $I_L$  er
- refleksiv ( $\forall x : x I_L x$ )
- symmetrisk ( $\forall x, y : x I_L y \Rightarrow y I_L x$ )
- transitiv ( $\forall x, y, z : x I_L y \wedge y I_L z \Rightarrow x I_L z$ )
- dvs.  $I_L$  er en ækvivalensrelation
- $I_L$  partitionerer  $\Sigma^*$
- $[x]$  er mængden af strenge, der er uskelnelige fra  $x$  mht.  $L$

# Egenskaber ved $I_L$

- $I_L$  er
- refleksiv ( $\forall x : x I_L x$ )
- symmetrisk ( $\forall x, y : x I_L y \Rightarrow y I_L x$ )
- transitiv ( $\forall x, y, z : x I_L y \wedge y I_L z \Rightarrow x I_L z$ )
- dvs.  $I_L$  er en ækvivalensrelation
- $I_L$  partitionerer  $\Sigma^*$
- $[x]$  er mængden af strenge, der er uskelnelige fra  $x$  mht.  $L$

# Egenskaber ved $I_L$

- $I_L$  er
- refleksiv ( $\forall x : x I_L x$ )
- symmetrisk ( $\forall x, y : x I_L y \Rightarrow y I_L x$ )
- transitiv ( $\forall x, y, z : x I_L y \wedge y I_L z \Rightarrow x I_L z$ )
- dvs.  $I_L$  er en ækvivalensrelation
- $I_L$  partitionerer  $\Sigma^*$
- $[x]$  er mængden af strenge, der er uskelnelige fra  $x$  mht.  $L$

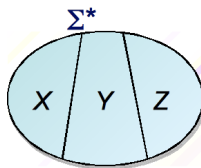
# Egenskaber ved $I_L$

- $I_L$  er
- refleksiv ( $\forall x : x I_L x$ )
- symmetrisk ( $\forall x, y : x I_L y \Rightarrow y I_L x$ )
- transitiv ( $\forall x, y, z : x I_L y \wedge y I_L z \Rightarrow x I_L z$ )
- dvs.  $I_L$  er en ækvivalensrelation
- $I_L$  partitionerer  $\Sigma^*$
- $[x]$  er mængden af strenge, der er uskelnelige fra  $x$  mht.  $L$

## Quiz

$$L = \{0, 1\}^* \{10\}$$

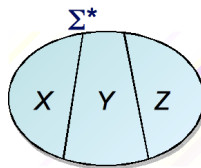
Beskriv ækvivalensklasserne for  $I_L$



## Quiz

$$L = \{0, 1\}^* \{10\}$$

Beskriv ækvivalensklasserne for  $I_L$

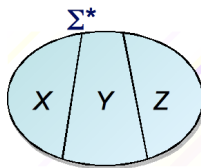


- Hint: der er 3 ækvivalensklasser...

# Quiz

$$L = \{0, 1\}^* \{10\}$$

Beskriv ækvivalensklasserne for  $I_L$

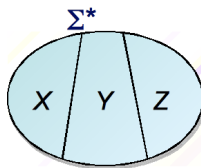


- Hint: der er 3 ækvivalensklasser...
- Hint: find en streng, der er skelnelig fra  $\Lambda$  ...

# Quiz

$$L = \{0, 1\}^* \{10\}$$

Beskriv ækvivalensklasserne for  $I_L$



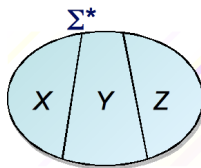
- Hint: der er 3 ækvivalensklasser...
- Hint: find en streng, der er skelnelig fra  $\Lambda$  ...
- Hint: find en streng, der er skelnelig fra både  $\Lambda$  og 1...



## Quiz

$$L = \{0, 1\}^* \{10\}$$

Beskriv ækvivalensklasserne for  $I_L$



- Hint: der er 3 ækvivalensklasser...
- Hint: find en streng, der er skelnelig fra  $\Lambda$  ...
- Hint: find en streng, der er skelnelig fra både  $\Lambda$  og 1...

$$X : \{\Lambda, 0\} \cup \{0, 1\}^* \{00\} = [\Lambda]$$

$$Y : \{0, 1\}^* \{1\} = [1]$$

$$Z : \{0, 1\}^* \{10\} = [10]$$

# MyHill-Nerode-sætningen

- $L$  er regulært  $\Leftrightarrow I_L$  har endeligt mange ækvivalensklasser
- “ $\Rightarrow$ ”: (1. seminar) hvis  $I_L$  har uendeligt mange ækvivalensklasser, så er  $L$  ikke regulært
- “ $\Leftarrow$ ”: Bevis følger...

# MyHill-Nerode-sætningen

- $L$  er regulært  $\Leftrightarrow I_L$  har endeligt mange ækvivalensklasser
- “ $\Rightarrow$ ”: (1. seminar) hvis  $I_L$  har uendeligt mange ækvivalensklasser, så er  $L$  ikke regulært
- “ $\Leftarrow$ ”: Bevis følger...

# MyHill-Nerode-sætningen

- $L$  er regulært  $\Leftrightarrow I_L$  har endeligt mange ækvivalensklasser
- “ $\Rightarrow$ ”: (1. seminar) hvis  $I_L$  har uendeligt mange ækvivalensklasser, så er  $L$  ikke regulært
- “ $\Leftarrow$ ”: Bevis følger...

## Konstruktion af en FA fra $I_L$

- Givet et sprog  $L \subseteq \Sigma^*$ , antag  $I_L$  har endeligt mange ækvivalensklasser.
- Vi kan definere en FA, hvor tilstandene er ækvivalensklasserne af  $I_L$

## Konstruktion af en FA fra $I_L$

- Givet et sprog  $L \subseteq \Sigma^*$ , antag  $I_L$  har endeligt mange ækvivalensklasser.
- Vi kan definere en FA, hvor tilstandene er ækvivalensklasserne af  $I_L$

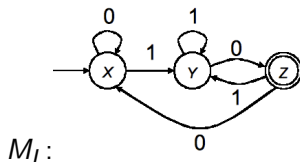
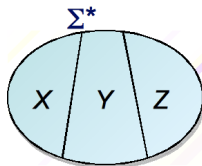
# Eksempel

- Ækvivalensklasserne for  $I_L$  når  $L = \{0, 1\}^* \{10\}$  :

$$X : \{\Lambda, 0\} \cup \{0, 1\}^* \{00\} = [\Lambda]$$

$$Y : \{0, 1\}^* \{1\} = [1]$$

$$Z : \{0, 1\}^* \{10\} = [10]$$



# Konstruktion af en FA fra $I_L$

- Definer en FA:  $M_L = (Q, \Sigma, q_0, A, \delta)$  hvor
- $Q = Q_L$  hvor  $Q_L$  er ækvivalensklasserne af  $I_L$
- $q_0 = [\Lambda]$
- $A = \{q \in Q \mid q \cap L \neq \emptyset\}$
- $\delta(q, a) = p$  hvis  $q = [x]$  og  $p = [xa]$  for en streng  $x$  ( $\delta$  er veldefineret idet  $xI_Ly \Rightarrow xaI_Lya$ )
- Påstand:  $L(M_L) = L$



# Konstruktion af en FA fra $I_L$

- Definer en FA:  $M_L = (Q, \Sigma, q_0, A, \delta)$  hvor
- $Q = Q_L$  hvor  $Q_L$  er ækvivalensklasserne af  $I_L$
- $q_0 = [\Lambda]$
- $A = \{q \in Q \mid q \cap L \neq \emptyset\}$
- $\delta(q, a) = p$  hvis  $q = [x]$  og  $p = [xa]$  for en streng  $x$  ( $\delta$  er veldefineret idet  $xI_Ly \Rightarrow xaI_Lya$ )
- Påstand:  $L(M_L) = L$

# Konstruktion af en FA fra $I_L$

- Definer en FA:  $M_L = (Q, \Sigma, q_0, A, \delta)$  hvor
- $Q = Q_L$  hvor  $Q_L$  er ækvivalensklasserne af  $I_L$
- $q_0 = [\Lambda]$
- $A = \{q \in Q \mid q \cap L \neq \emptyset\}$
- $\delta(q, a) = p$  hvis  $q = [x]$  og  $p = [xa]$  for en streng  $x$  ( $\delta$  er veldefineret idet  $xI_Ly \Rightarrow xaI_Lya$ )
- Påstand:  $L(M_L) = L$

# Konstruktion af en FA fra $I_L$

- Definer en FA:  $M_L = (Q, \Sigma, q_0, A, \delta)$  hvor
- $Q = Q_L$  hvor  $Q_L$  er ækvivalensklasserne af  $I_L$
- $q_0 = [\Lambda]$
- $A = \{q \in Q | q \cap L \neq \emptyset\}$
- $\delta(q, a) = p$  hvis  $q = [x]$  og  $p = [xa]$  for en streng  $x$  ( $\delta$  er veldefineret idet  $xI_Ly \Rightarrow xaI_Lya$ )
- Påstand:  $L(M_L) = L$

# Konstruktion af en FA fra $I_L$

- Definer en FA:  $M_L = (Q, \Sigma, q_0, A, \delta)$  hvor
- $Q = Q_L$  hvor  $Q_L$  er ækvivalensklasserne af  $I_L$
- $q_0 = [\Lambda]$
- $A = \{q \in Q \mid q \cap L \neq \emptyset\}$
- $\delta(q, a) = p$  hvis  $q = [x]$  og  $p = [xa]$  for en streng  $x$  ( $\delta$  er veldefineret idet  $xI_Ly \Rightarrow xaI_Lya$ )
- Påstand:  $L(M_L) = L$

# Konstruktion af en FA fra $I_L$

- Definer en FA:  $M_L = (Q, \Sigma, q_0, A, \delta)$  hvor
- $Q = Q_L$  hvor  $Q_L$  er ækvivalensklasserne af  $I_L$
- $q_0 = [\Lambda]$
- $A = \{q \in Q \mid q \cap L \neq \emptyset\}$
- $\delta(q, a) = p$  hvis  $q = [x]$  og  $p = [xa]$  for en streng  $x$  ( $\delta$  er veldefineret idet  $xI_Ly \Rightarrow xaI_Lya$ )
- Påstand:  $L(M_L) = L$

# Quiz

- Antag ækvivalensklasserne for  $I_L$  er

$$X = \{x \in \{0,1\}^* \mid \text{antal 1'er i } x \text{ er lige} \}$$

$$Y = \{x \in \{0,1\}^* \mid \text{antal 1'er i } x \text{ er ulige} \}$$

og  $111 \in L$

Lav en FA, der accepterer  $L$

## Quiz

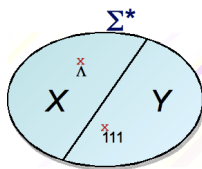
- Antag ækvivalensklasserne for  $I_L$  er

$$X = \{x \in \{0,1\}^* \mid \text{antal 1'er i } x \text{ er lige} \}$$

$$Y = \{x \in \{0,1\}^* \mid \text{antal 1'er i } x \text{ er ulige} \}$$

og  $111 \in L$

Lav en FA, der accepterer L



## Quiz

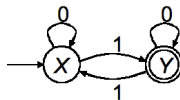
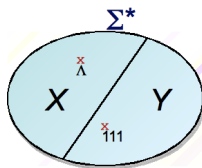
- Antag ækvivalensklasserne for  $I_L$  er

$$X = \{x \in \{0,1\}^* \mid \text{antal 1'er i } x \text{ er lige} \}$$

$$Y = \{x \in \{0,1\}^* \mid \text{antal 1'er i } x \text{ er ulige} \}$$

og  $111 \in L$

Lav en FA, der accepterer L





# Bevis for korrekthed af konstruktionen

- Påstand:  $L(M_L) = L$
- Lemma:  $\forall x, y \in \Sigma^* : \delta^*([x], y) = [xy]$
- Bevis: induktion i strukturen af  $y$ ...
- $\delta^*(q_0, x) = \delta^*([\Lambda], x) = [x]$  (følger af lemmaet og def. af  $q_0$ )
- $x \in L(M_L) \Leftrightarrow \delta^*(q_0, x) \in A \Leftrightarrow [x] \in A \Leftrightarrow [x] \cap L \neq \emptyset$
- $x \in L \Rightarrow [x] \cap L \neq \emptyset$  (da  $x \in [x]$ )
- $[x] \cap L \neq \emptyset \Rightarrow x \in L$  (bruger def. af  $L$ )
- dvs.  $x \in L(M_L) \Leftrightarrow x \in L$

# Bevis for korrekthed af konstruktionen

- Påstand:  $L(M_L) = L$
- Lemma:  $\forall x, y \in \Sigma^* : \delta^*([x], y) = [xy]$
- Bevis: induktion i strukturen af  $y$ ...
- $\delta^*(q_0, x) = \delta^*([\Lambda], x) = [x]$  (følger af lemmaet og def. af  $q_0$ )
- $x \in L(M_L) \Leftrightarrow \delta^*(q_0, x) \in A \Leftrightarrow [x] \in A \Leftrightarrow [x] \cap L \neq \emptyset$
- $x \in L \Rightarrow [x] \cap L \neq \emptyset$  (da  $x \in [x]$ )
- $[x] \cap L \neq \emptyset \Rightarrow x \in L$  (bruger def. af  $L_L$ )
- dvs.  $x \in L(M_L) \Leftrightarrow x \in L$

## Bevis for korrekthed af konstruktionen

- Påstand:  $L(M_L) = L$
- Lemma:  $\forall x, y \in \Sigma^* : \delta^*([x], y) = [xy]$
- Bevis: induktion i strukturen af  $y$ ...
- $\delta^*(q_0, x) = \delta^*([\Lambda], x) = [x]$  (følger af lemmaet og def. af  $q_0$ )
- $x \in L(M_L) \Leftrightarrow \delta^*(q_0, x) \in A \Leftrightarrow [x] \in A \Leftrightarrow [x] \cap L \neq \emptyset$
- $x \in L \Rightarrow [x] \cap L \neq \emptyset$  (da  $x \in [x]$ )
- $[x] \cap L \neq \emptyset \Rightarrow x \in L$  (bruger def. af  $L$ )
- dvs.  $x \in L(M_L) \Leftrightarrow x \in L$

# Bevis for korrekthed af konstruktionen

- Påstand:  $L(M_L) = L$
- Lemma:  $\forall x, y \in \Sigma^* : \delta^*([x], y) = [xy]$
- Bevis: induktion i strukturen af  $y$ ...
- $\delta^*(q_0, x) = \delta^*([\Lambda], x) = [x]$  (følger af lemmaet og def. af  $q_0$ )
- $x \in L(M_L) \Leftrightarrow \delta^*(q_0, x) \in A \Leftrightarrow [x] \in A \Leftrightarrow [x] \cap L \neq \emptyset$
- $x \in L \Rightarrow [x] \cap L \neq \emptyset$  (da  $x \in [x]$ )
- $[x] \cap L \neq \emptyset \Rightarrow x \in L$  (bruger def. af  $L$ )
- dvs.  $x \in L(M_L) \Leftrightarrow x \in L$

# Bevis for korrekthed af konstruktionen

- Påstand:  $L(M_L) = L$
- Lemma:  $\forall x, y \in \Sigma^* : \delta^*([x], y) = [xy]$
- Bevis: induktion i strukturen af  $y$ ...
- $\delta^*(q_0, x) = \delta^*([\Lambda], x) = [x]$  (følger af lemmaet og def. af  $q_0$ )
- $x \in L(M_L) \Leftrightarrow \delta^*(q_0, x) \in A \Leftrightarrow [x] \in A \Leftrightarrow [x] \cap L \neq \emptyset$
- $x \in L \Rightarrow [x] \cap L \neq \emptyset$  (da  $x \in [x]$ )
- $[x] \cap L \neq \emptyset \Rightarrow x \in L$  (bruger def. af  $L_L$ )
- dvs.  $x \in L(M_L) \Leftrightarrow x \in L$

# Bevis for korrekthed af konstruktionen

- Påstand:  $L(M_L) = L$
- Lemma:  $\forall x, y \in \Sigma^* : \delta^*([x], y) = [xy]$
- Bevis: induktion i strukturen af  $y$ ...
- $\delta^*(q_0, x) = \delta^*([\Lambda], x) = [x]$  (følger af lemmaet og def. af  $q_0$ )
- $x \in L(M_L) \Leftrightarrow \delta^*(q_0, x) \in A \Leftrightarrow [x] \in A \Leftrightarrow [x] \cap L \neq \emptyset$
- $x \in L \Rightarrow [x] \cap L \neq \emptyset$  (da  $x \in [x]$ )
- $[x] \cap L \neq \emptyset \Rightarrow x \in L$  (bruger def. af  $L_L$ )
- dvs.  $x \in L(M_L) \Leftrightarrow x \in L$

# Bevis for korrekthed af konstruktionen

- Påstand:  $L(M_L) = L$
- Lemma:  $\forall x, y \in \Sigma^* : \delta^*([x], y) = [xy]$
- Bevis: induktion i strukturen af  $y$ ...
- $\delta^*(q_0, x) = \delta^*([\Lambda], x) = [x]$  (følger af lemmaet og def. af  $q_0$ )
- $x \in L(M_L) \Leftrightarrow \delta^*(q_0, x) \in A \Leftrightarrow [x] \in A \Leftrightarrow [x] \cap L \neq \emptyset$
- $x \in L \Rightarrow [x] \cap L \neq \emptyset$  (da  $x \in [x]$ )
- $[x] \cap L \neq \emptyset \Rightarrow x \in L$  (bruger def. af  $L_L$ )
- dvs.  $x \in L(M_L) \Leftrightarrow x \in L$

## Bevis for korrekthed af konstruktionen

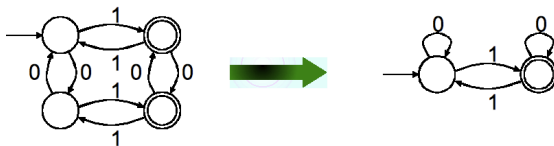
- Påstand:  $L(M_L) = L$
- Lemma:  $\forall x, y \in \Sigma^* : \delta^*([x], y) = [xy]$
- Bevis: induktion i strukturen af  $y$ ...
- $\delta^*(q_0, x) = \delta^*([\Lambda], x) = [x]$  (følger af lemmaet og def. af  $q_0$ )
- $x \in L(M_L) \Leftrightarrow \delta^*(q_0, x) \in A \Leftrightarrow [x] \in A \Leftrightarrow [x] \cap L \neq \emptyset$
- $x \in L \Rightarrow [x] \cap L \neq \emptyset$  (da  $x \in [x]$ )
- $[x] \cap L \neq \emptyset \Rightarrow x \in L$  (bruger def. af  $L_L$ )
- dvs.  $x \in L(M_L) \Leftrightarrow x \in L$



# Øvelser

- [Martin] Opg. 5.2 (p. 191) Find selv ækvivalensklasser
- [Martin] Opg. 5.7 Konstruer en FA ud fra en beskrivelse af  $I_L$

# Minimering af automater



- Man kan i visse tilfælde opnå en mindre FA ved at “slå tilstande sammen”...
- Kan vi gøre det systematisk?
- Vil den resulterende FA blive minimal?

# En algoritme til FA-minimering

Fra MyHill-Nerode-sætningen kan vi udlede en algoritme, der givet en vilkårlig FA  $M = (Q, \Sigma, q_0, A, \delta)$ , finder en minimal FA  $M_1$  hvor  $L(M_1) = L(M)$

## To partitioneringer af $\Sigma^*$

- 1: Ækvivalensklasserne af  $I_L$  (svarer til tilstandene i den minimale FA  $M_L$ )
- 2: En opdeling af alle  $x \in \Sigma^*$  efter værdien af  $\delta^*(q_0, x)$  (svarer til tilstandene i den givne FA  $M$ )
- Kan vi konstruere 1 ud fra 2?
- Definer for alle  $q \in Q$ :  $L_q = \{x \in \Sigma^* \mid \delta^*(q_0, x) = q\}$

## To partitioneringer af $\Sigma^*$

- 1: Ækvivalensklasserne af  $I_L$  (svarer til tilstandene i den minimale FA  $M_L$ )
- 2: En opdeling af alle  $x \in \Sigma^*$  efter værdien af  $\delta^*(q_0, x)$  (svarer til tilstandene i den givne FA  $M$ )
- Kan vi konstruere 1 ud fra 2?
- Definer for alle  $q \in Q$ :  $L_q = \{x \in \Sigma^* \mid \delta^*(q_0, x) = q\}$

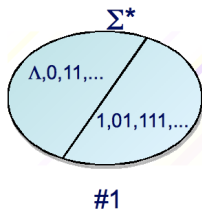
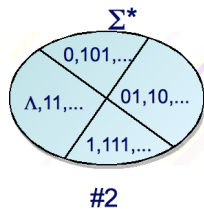
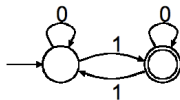
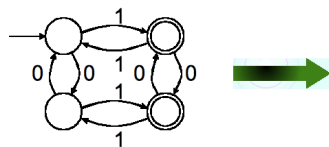
## To partitioneringer af $\Sigma^*$

- 1: Ækvivalensklasserne af  $I_L$  (svarer til tilstandene i den minimale FA  $M_L$ )
- 2: En opdeling af alle  $x \in \Sigma^*$  efter værdien af  $\delta^*(q_0, x)$  (svarer til tilstandene i den givne FA  $M$ )
- Kan vi konstruere 1 ud fra 2?
- Definer for alle  $q \in Q$ :  $L_q = \{x \in \Sigma^* \mid \delta^*(q_0, x) = q\}$

## To partitioneringer af $\Sigma^*$

- 1: Ækvivalensklasserne af  $I_L$  (svarer til tilstandene i den minimale FA  $M_L$ )
- 2: En opdeling af alle  $x \in \Sigma^*$  efter værdien af  $\delta^*(q_0, x)$  (svarer til tilstandene i den givne FA  $M$ )
- Kan vi konstruere 1 ud fra 2?
- Definer for alle  $q \in Q$ :  $L_q = \{x \in \Sigma^* \mid \delta^*(q_0, x) = q\}$

## Eksempel





## Fjern uopnåelige tilstande

- Ækvivalensklasserne af  $I_L$  indeholder alle mindst 1 streng
- Det er muligt at  $L_q = \emptyset$  for en eller flere  $q \in Q$  (hvis  $q$  er uopnåelig fra  $q_0$ )
- Der findes en algoritme, der kan fjerne uopnåelige tilstande fra en FA uden at ændre sproget
- Vi kan derfor antage at  $L_q \neq \emptyset$  for alle  $q \in Q$

## Fjern uopnåelige tilstande

- Ækvivalensklasserne af  $I_L$  indeholder alle mindst 1 streng
- Det er muligt at  $L_q = \emptyset$  for en eller flere  $q \in Q$  (hvis  $q$  er uopnåelig fra  $q_0$ )
- Der findes en algoritme, der kan fjerne uopnåelige tilstande fra en FA uden at ændre sproget
- Vi kan derfor antage at  $L_q \neq \emptyset$  for alle  $q \in Q$

## Fjern uopnåelige tilstande

- Ækvivalensklasserne af  $I_L$  indeholder alle mindst 1 streng
- Det er muligt at  $L_q = \emptyset$  for en eller flere  $q \in Q$  (hvis  $q$  er uopnåelig fra  $q_0$ )
- Der findes en algoritme, der kan fjerne uopnåelige tilstande fra en FA uden at ændre sproget
- Vi kan derfor antage at  $L_q \neq \emptyset$  for alle  $q \in Q$

## Fjern uopnåelige tilstande

- Ækvivalensklasserne af  $I_L$  indeholder alle mindst 1 streng
- Det er muligt at  $L_q = \emptyset$  for en eller flere  $q \in Q$  (hvis  $q$  er uopnåelig fra  $q_0$ )
- Der findes en algoritme, der kan fjerne uopnåelige tilstande fra en FA uden at ændre sproget
- Vi kan derfor antage at  $Lq \neq \emptyset$  for alle  $q \in Q$

# Opnåelige tilstande

- Givet en FA  $M = (Q, \Sigma, q_0, A, \delta)$
- Lad  $R$  være den mindste mængde, der opfylder:
- $q_0 \in R$
- $\forall q \in R, a \in \Sigma : \delta(q, a) \in R$
- $R$  er mængden af opnåelige tilstande i  $M$
- (minder om def. af  $\Lambda$  – lukning)

# Opnåelige tilstande

- Givet en FA  $M = (Q, \Sigma, q_0, A, \delta)$
- Lad  $R$  være den mindste mængde, der opfylder:
  - $q_0 \in R$
  - $\forall q \in R, a \in \Sigma : \delta(q, a) \in R$
  - $R$  er mængden af opnåelige tilstande i  $M$
  - (minder om def. af  $\Lambda$  – lukning)

# Opnåelige tilstande

- Givet en FA  $M = (Q, \Sigma, q_0, A, \delta)$
- Lad  $R$  være den mindste mængde, der opfylder:
- $q_0 \in R$
- $\forall q \in R, a \in \Sigma : \delta(q, a) \in R$
- $R$  er mængden af opnåelige tilstande i  $M$
- (minder om def. af  $\Lambda$  – lukning)

# Opnåelige tilstande

- Givet en FA  $M = (Q, \Sigma, q_0, A, \delta)$
- Lad  $R$  være den mindste mængde, der opfylder:
- $q_0 \in R$
- $\forall q \in R, a \in \Sigma : \delta(q, a) \in R$
- $R$  er mængden af opnåelige tilstande i  $M$
- (minder om def. af  $\Lambda$  – lukning)



# Opnåelige tilstande

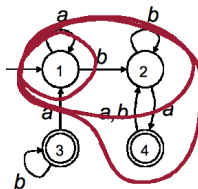
- Givet en FA  $M = (Q, \Sigma, q_0, A, \delta)$
- Lad  $R$  være den mindste mængde, der opfylder:
- $q_0 \in R$
- $\forall q \in R, a \in \Sigma : \delta(q, a) \in R$
- $R$  er mængden af opnåelige tilstande i  $M$
- (minder om def. af  $\Lambda$  – lukning)

# Opnåelige tilstande

- Givet en FA  $M = (Q, \Sigma, q_0, A, \delta)$
- Lad  $R$  være den mindste mængde, der opfylder:
- $q_0 \in R$
- $\forall q \in R, a \in \Sigma : \delta(q, a) \in R$
- $R$  er mængden af opnåelige tilstande i  $M$
- (minder om def. af  $\Lambda$  – lukning)

# Fixpunktsalgoritme

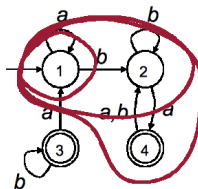
- $R$  kan findes med en fixpunktsalgoritme:



- $1 \in R$
- $\delta(1, b) = 2 \in R$
- $\delta(2, a) = 4 \in R$
- fixpunkt er nu nået  
dvs. de opnåelige tilstande er  $R = \{1, 2, 4\}$

# Fixpunktsalgoritme

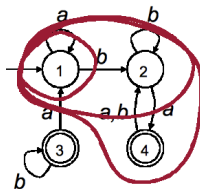
- $R$  kan findes med en fixpunktsalgoritme:



- $1 \in R$
- $\delta(1, b) = 2 \in R$
- $\delta(2, a) = 4 \in R$
- fixpunkt er nu nået  
dvs. de opnåelige tilstande er  $R = \{1, 2, 4\}$

# Fixpunktsalgoritme

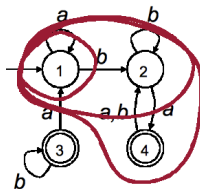
- $R$  kan findes med en fixpunktsalgoritme:



- $1 \in R$
- $\delta(1, b) = 2 \in R$
- $\delta(2, a) = 4 \in R$
- fixpunkt er nu nået  
dvs. de opnåelige tilstande er  $R = \{1, 2, 4\}$

# Fixpunktsalgoritme

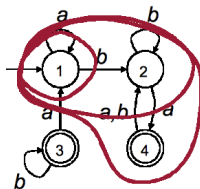
- $R$  kan findes med en fixpunktsalgoritme:



- $1 \in R$
- $\delta(1, b) = 2 \in R$
- $\delta(2, a) = 4 \in R$
- fixpunkt er nu nået  
dvs. de opnåelige tilstande er  $R = \{1, 2, 4\}$

# Fixpunktsalgoritme

- $R$  kan findes med en fixpunktsalgoritme:



- $1 \in R$
- $\delta(1, b) = 2 \in R$
- $\delta(2, a) = 4 \in R$
- fixpunkt er nu nået  
dvs. de opnåelige tilstande er  $R = \{1, 2, 4\}$

## Forholdet mellem partition 1 og 2

- Fra seminar 1:  $\delta^*(q_0, x) = \delta^*(q_0, y) \Rightarrow xI_L y$
- Dvs. enhver  $L_q$ -mængde er helt indeholdt i én  $I_L$ -ækvivalensklasse
- Enhver ækvivalensklasse af  $I_L$  er derfor foreningen af en eller flere af  $L_q$ -mængderne
- Da  $L_q \neq \emptyset$  er hver af disse foreninger unik
- Definition:  $p \equiv q \Leftrightarrow L_p$  og  $L_q$  er delmængder af samme  $I_L$ -ækvivalensklasse
- Dvs. hvis  $p \equiv q$ , så svarer  $p$  og  $q$  til samme tilstand i den minimale automat!



## Forholdet mellem partition 1 og 2

- Fra seminar 1:  $\delta^*(q_0, x) = \delta^*(q_0, y) \Rightarrow xI_L y$
- Dvs. enhver  $L_q$ -mængde er helt indeholdt i én  $I_L$ -ækvivalensklasse
- Enhver ækvivalensklasse af  $I_L$  er derfor foreningen af en eller flere af  $L_q$ -mængderne
- Da  $L_q \neq \emptyset$  er hver af disse foreninger unik
- Definition:  $p \equiv q \Leftrightarrow L_p$  og  $L_q$  er delmængder af samme  $I_L$ -ækvivalensklasse
- Dvs. hvis  $p \equiv q$ , så svarer  $p$  og  $q$  til samme tilstand i den minimale automat!

## Forholdet mellem partition 1 og 2

- Fra seminar 1:  $\delta^*(q_0, x) = \delta^*(q_0, y) \Rightarrow xI_L y$
- Dvs. enhver  $L_q$ -mængde er helt indeholdt i én  $I_L$ -ækvivalensklasse
- Enhver ækvivalensklasse af  $I_L$  er derfor foreningen af en eller flere af  $L_q$ -mængderne
- Da  $L_q \neq \emptyset$  er hver af disse foreninger unik
- Definition:  $p \equiv q \Leftrightarrow L_p$  og  $L_q$  er delmængder af samme  $I_L$ -ækvivalensklasse
- Dvs. hvis  $p \equiv q$ , så svarer  $p$  og  $q$  til samme tilstand i den minimale automat!

## Forholdet mellem partition 1 og 2

- Fra seminar 1:  $\delta^*(q_0, x) = \delta^*(q_0, y) \Rightarrow xI_L y$
- Dvs. enhver  $L_q$ -mængde er helt indeholdt i én  $I_L$ -ækvivalensklasse
- Enhver ækvivalensklasse af  $I_L$  er derfor foreningen af en eller flere af  $L_q$ -mængderne
- Da  $L_q \neq \emptyset$  er hver af disse foreninger unik
- Definition:  $p \equiv q \Leftrightarrow L_p$  og  $L_q$  er delmængder af samme  $I_L$ -ækvivalensklasse
- Dvs. hvis  $p \equiv q$ , så svarer  $p$  og  $q$  til samme tilstand i den minimale automat!

## Forholdet mellem partition 1 og 2

- Fra seminar 1:  $\delta^*(q_0, x) = \delta^*(q_0, y) \Rightarrow xI_L y$
- Dvs. enhver  $L_q$ -mængde er helt indeholdt i én  $I_L$ -ækvivalensklasse
- Enhver ækvivalensklasse af  $I_L$  er derfor foreningen af en eller flere af  $L_q$ -mængderne
- Da  $L_q \neq \emptyset$  er hver af disse foreninger unik
- Definition:  $p \equiv q \Leftrightarrow L_p$  og  $L_q$  er delmængder af samme  $I_L$ -ækvivalensklasse
- Dvs. hvis  $p \equiv q$ , så svarer  $p$  og  $q$  til samme tilstand i den minimale automat!

## Forholdet mellem partition 1 og 2

- Fra seminar 1:  $\delta^*(q_0, x) = \delta^*(q_0, y) \Rightarrow xI_L y$
- Dvs. enhver  $L_q$ -mængde er helt indeholdt i én  $I_L$ -ækvivalensklasse
- Enhver ækvivalensklasse af  $I_L$  er derfor foreningen af en eller flere af  $L_q$ -mængderne
- Da  $L_q \neq \emptyset$  er hver af disse foreninger unik
- Definition:  $p \equiv q \Leftrightarrow L_p$  og  $L_q$  er delmængder af samme  $I_L$ -ækvivalensklasse
- Dvs. hvis  $p \equiv q$ , så svarer  $p$  og  $q$  til samme tilstand i den minimale automat!

# Konstruktion af $\equiv$ (minimeringsalgoritmen)

- Lad  $S$  være den mindste mængde, der opfylder:
- a)  $(p \in A \wedge q \notin A) \vee (p \notin A \wedge q \in A) \Rightarrow (p, q) \in S$
- b)  $(\exists a \in \Sigma : (\delta(p, a), \delta(q, a)) \in S) \Rightarrow (p, q) \in S$
- Påstand:  $p \equiv q$  hvis og kun hvis  $(p, q) \in S$   
S kan beregnes med en fixpunktsalgoritme (i stil med opnåelige tilstande og  $\wedge$ -lukning tidligere...)

# Konstruktion af $\equiv$ (minimeringsalgoritmen)

- Lad  $S$  være den mindste mængde, der opfylder:
- a)  $(p \in A \wedge q \notin A) \vee (p \notin A \wedge q \in A) \Rightarrow (p, q) \in S$
- b)  $(\exists a \in \Sigma : (\delta(p, a), \delta(q, a)) \in S) \Rightarrow (p, q) \in S$
- Påstand:  $p \equiv q$  hvis og kun hvis  $(p, q) \in S$   
 $S$  kan beregnes med en fixpunktsalgoritme (i stil med opnåelige tilstande og  $\wedge$ -lukning tidligere...)

# Konstruktion af $\equiv$ (minimeringsalgoritmen)

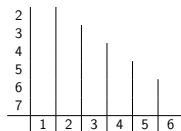
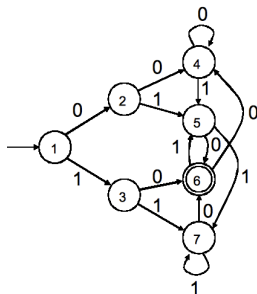
- Lad  $S$  være den mindste mængde, der opfylder:
- a)  $(p \in A \wedge q \notin A) \vee (p \notin A \wedge q \in A) \Rightarrow (p, q) \in S$
- b)  $(\exists a \in \Sigma : (\delta(p, a), \delta(q, a)) \in S) \Rightarrow (p, q) \in S$
- Påstand:  $p \equiv q$  hvis og kun hvis  $(p, q) \in S$   
S kan beregnes med en fixpunktsalgoritme (i stil med opnåelige tilstande og  $\wedge$ -lukning tidligere...)



# Konstruktion af $\equiv$ (minimeringsalgoritmen)

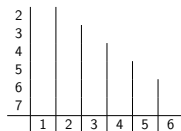
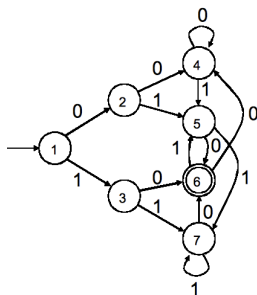
- Lad  $S$  være den mindste mængde, der opfylder:
- a)  $(p \in A \wedge q \notin A) \vee (p \notin A \wedge q \in A) \Rightarrow (p, q) \in S$
- b)  $(\exists a \in \Sigma : (\delta(p, a), \delta(q, a)) \in S) \Rightarrow (p, q) \in S$
- Påstand:  $p \equiv q$  hvis og kun hvis  $(p, q) \in S$   
S kan beregnes med en fixpunktsalgoritme (i stil med opnåelige tilstande og  $\wedge$ -lukning tidligere...)

# Eksempel



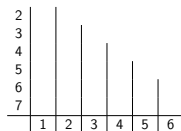
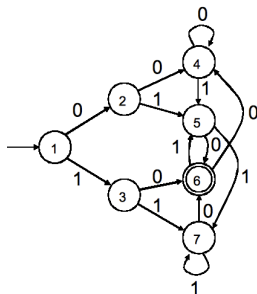
- Find alle par af tilstande
- Fjern uopnåelige tilstande (ingen i denne FA)
- Marker alle par som med/ikke med i  $A$
- Find  $\equiv$  ved at udfylde en tabel for  $S$  (fixpunktsberegning)
- Kombiner tilstande, der svarer til umærkede par

# Eksempel



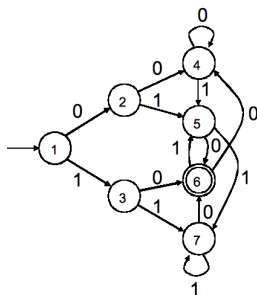
- Find alle par af tilstande
- Fjern uopnåelige tilstande (ingen i denne FA)
- Marker alle par som med/ikke med i  $A$
- Find  $\equiv$  ved at udfylde en tabel for  $S$  (fixpunktsberegning)
- Kombiner tilstande, der svarer til umærkede par

# Eksempel



- Find alle par af tilstande
- Fjern uopnåelige tilstande (ingen i denne FA)
- Marker alle par som med/ikke med i  $A$
- Find  $\equiv$  ved at udfylde en tabel for  $S$  (fixpunktsberegning)
- Kombiner tilstande, der svarer til umærkede par

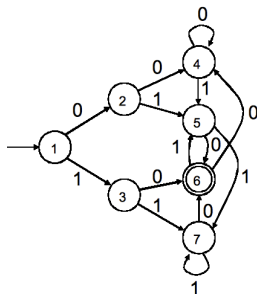
# Eksempel



2						
3						
4						
5						
6	X	X	X	X	X	
7						
	1	2	3	4	5	6

- Find alle par af tilstande
- Fjern uopnåelige tilstande (ingen i denne FA)
- Marker alle par som med/ikke med i  $A$
- Find  $\equiv$  ved at udfylde en tabel for  $S$  (fixpunktsberegning)
- Kombiner tilstande, der svarer til umærkede par

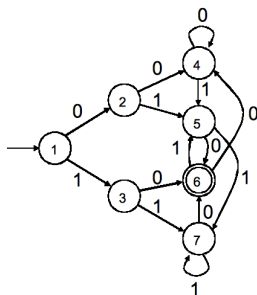
# Eksempel



2						
3						
4						
5						
6	X	X	X	X	X	
7						
	1	2	3	4	5	6

- Find alle par af tilstande
- Fjern uopnåelige tilstande (ingen i denne FA)
- Marker alle par som med/ikke med i  $A$
- Find  $\equiv$  ved at udfylde en tabel for  $S$  (fixpunktsberegning)
- Kombiner tilstande, der svarer til umærkede par

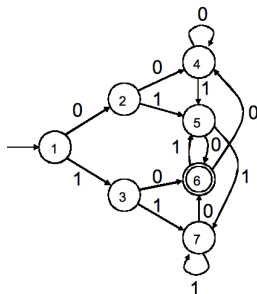
# Eksempel



2						
3						
4						
5						
6	X	X	X	X	X	
7						
	1	2	3	4	5	6

- Find alle par af tilstande
- Fjern uopnåelige tilstande (ingen i denne FA)
- Marker alle par som med/ikke med i  $A$
- Find  $\equiv$  ved at udfylde en tabel for  $S$  (fixpunktsberegning)
- Kombiner tilstande, der svarer til umærkede par

# Eksempel

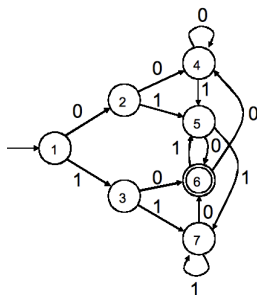


2						
3	X	X				
4			X			
5	X	X		X		
6	X	X	X	X	X	
7	X	X		X		X
	1	2	3	4	5	6

- Find alle par af tilstande
- Fjern uopnåelige tilstande (ingen i denne FA)
- Marker alle par som med/ikke med i  $A$
- Find  $\equiv$  ved at udfylde en tabel for  $S$  (fixpunktsberegning)
- Kombiner tilstande, der svarer til umærkede par



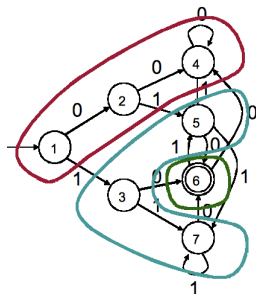
# Eksempel



2						
3	X	X				
4			X			
5	X	X		X		
6	X	X	X	X	X	
7	X	X		X		X
	1	2	3	4	5	6

- Find alle par af tilstande
- Fjern uopnåelige tilstande (ingen i denne FA)
- Marker alle par som med/ikke med i  $A$
- Find  $\equiv$  ved at udfylde en tabel for  $S$  (fixpunktsberegning)
- Kombiner tilstande, der svarer til umærkede par

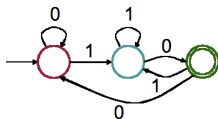
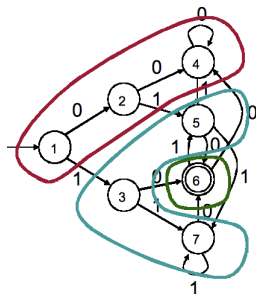
# Eksempel



2							
3	X	X					
4			X				
5	X	X		X			
6	X	X	X	X	X		
7	X	X		X		X	
	1	2	3	4	5	6	

- Find alle par af tilstande
- Fjern uopnåelige tilstande (ingen i denne FA)
- Marker alle par som med/ikke med i  $A$
- Find  $\equiv$  ved at udfylde en tabel for  $S$  (fixpunktsberegning)
- Kombiner tilstande, der svarer til umærkede par

# Eksempel



2							
3	X	X					
4			X				
5	X	X	X	X	X		
6	X	X	X	X		X	
7	X	X		X			X
	1	2	3	4	5	6	

- Find alle par af tilstande
- Fjern uopnåelige tilstande (ingen i denne FA)
- Marker alle par som med/ikke med i  $A$
- Find  $\equiv$  ved at udfylde en tabel for  $S$  (fixpunktsberegning)
- Kombiner tilstande, der svarer til umærkede par

# Bevis for korrekthed af minimeringsalgoritmen

- Antag  $p, q \in Q, x \in L_p, y \in L_q$   
(dvs.  $\delta^*(q_0, x) = p$  og  $\delta^*(q_0, y) = q$ )
- Lemma: Følgende udsagn er ækvivalente:

$$p \equiv q$$

$$x|_L y$$

$$\forall z \in \Sigma^* : \delta^*(p, z) \in A \Leftrightarrow \delta^*(q, z) \in A$$

# Bevis for korrekthed af minimeringsalgoritmen

- Antag  $p, q \in Q, x \in L_p, y \in L_q$   
(dvs.  $\delta^*(q_0, x) = p$  og  $\delta^*(q_0, y) = q$ )
- Lemma: Følgende udsagn er ækvivalente:

$$p \equiv q$$

$$xI_L y$$

$$\forall z \in \Sigma^* : \delta^*(p, z) \in A \Leftrightarrow \delta^*(q, z) \in A$$

## Bevis for korrekthed, fortsat

- Påstand:  $p \equiv q$  hvis og kun hvis  $(p, q) \notin S$
- Iflg. lemmaet:

$$p \not\equiv q \Leftrightarrow (\exists z \in \Sigma^* : (\delta^*(p, z) \in A \wedge \delta^*(q, z) \notin A) \vee (\delta^*(p, z) \notin A \wedge \delta^*(q, z) \in A))$$

- $p \not\equiv q \Rightarrow (p, q) \in S$  (brug lemmaet, lav induktion i  $z$ )
- $(p, q) \in S \Rightarrow p \not\equiv q$  (brug lemmaet, lav induktion i  $S$ )

## Bevis for korrekthed, fortsat

- Påstand:  $p \equiv q$  hvis og kun hvis  $(p, q) \notin S$
- Iflg. lemmaet:

$$p \not\equiv q \Leftrightarrow (\exists z \in \Sigma^* : (\delta^*(p, z) \in A \wedge \delta^*(q, z) \notin A) \vee (\delta^*(p, z) \notin A \wedge \delta^*(q, z) \in A))$$

- $p \not\equiv q \Rightarrow (p, q) \in S$  (brug lemmaet, lav induktion i  $z$ )
- $(p, q) \in S \Rightarrow p \not\equiv q$  (brug lemmaet, lav induktion i  $S$ )

## Bevis for korrekthed, fortsat

- Påstand:  $p \equiv q$  hvis og kun hvis  $(p, q) \notin S$
- Iflg. lemmaet:

$$p \not\equiv q \Leftrightarrow (\exists z \in \Sigma^* : (\delta^*(p, z) \in A \wedge \delta^*(q, z) \notin A) \vee (\delta^*(p, z) \notin A \wedge \delta^*(q, z) \in A))$$

- $p \not\equiv q \Rightarrow (p, q) \in S$  (brug lemmaet, lav induktion i  $z$ )
- $(p, q) \in S \Rightarrow p \not\equiv q$  (brug lemmaet, lav induktion i  $S$ )



## Bevis for korrekthed, fortsat

- Påstand:  $p \equiv q$  hvis og kun hvis  $(p, q) \notin S$
- Iflg. lemmaet:

$$p \not\equiv q \Leftrightarrow (\exists z \in \Sigma^* : (\delta^*(p, z) \in A \wedge \delta^*(q, z) \notin A) \vee (\delta^*(p, z) \notin A \wedge \delta^*(q, z) \in A))$$

- $p \not\equiv q \Rightarrow (p, q) \in S$  (brug lemmaet, lav induktion i  $z$ )
- $(p, q) \in S \Rightarrow p \not\equiv q$  (brug lemmaet, lav induktion i  $S$ )

# Øvelse

- [Martin] 5.16 (a+e) (p.192)

# Plan

Nondeterministiske automater

Determinisering

NFA- $\Lambda$ 'er

Kleenes sætning

Kleenes sætning del 1

Kleenes sætning del 2

Frokost

Minimering

Myhill Nerode

Java projekt

# Regaut pakken

- Udleverede programdele:
- `NFA.java` og `NFALambda.java`: repræsentation af NFA'er og NFA- $\Lambda$ 'er
- `RegExp.java`: repræsentation af regulære udtryk
- parser for regulære udtryk
- de trivielle oversættelser:  $FA \rightarrow NFA$ ,  $NFA \rightarrow NFA - \Lambda$

# Regaut pakken

- Udleverede programdele:
- `NFA.java` og `NFALambda.java`: repræsentation af NFA'er og NFA- $\Lambda$ 'er
- `RegExp.java`: repræsentation af regulære udtryk
- parser for regulære udtryk
- de trivielle oversættelser:  $FA \rightarrow NFA$ ,  $NFA \rightarrow NFA - \Lambda$

# Regaut pakken

- Udleverede programdele:
- `NFA.java` og `NFALambda.java`: repræsentation af NFA'er og NFA- $\Lambda$ 'er
- `RegExp.java`: repræsentation af regulære udtryk
- parser for regulære udtryk
- de trivielle oversættelser:  $FA \rightarrow NFA$ ,  $NFA \rightarrow NFA - \Lambda$

# Regaut pakken

- Udleverede programdele:
- `NFA.java` og `NFALambda.java`: repræsentation af NFA'er og NFA- $\Lambda$ 'er
- `RegExp.java`: repræsentation af regulære udtryk
- parser for regulære udtryk
- de trivielle oversættelser:  $FA \rightarrow NFA$ ,  $NFA \rightarrow NFA - \Lambda$

# Regaut pakken

- Udleverede programdele:
- `NFA.java` og `NFALambda.java`: repræsentation af NFA'er og NFA- $\Lambda$ 'er
- `RegExp.java`: repræsentation af regulære udtryk
- parser for regulære udtryk
- de trivielle oversættelser:  $FA \rightarrow NFA$ ,  $NFA \rightarrow NFA - \Lambda$



# NFA.java

- Repræsentation som `FA.java`, med én undtagelse:
- `transitions` er en funktion fra `StateSymbolPair` til en **mængde** af `State` objekter

# NFA.java

- Repræsentation som `FA.java`, med én undtagelse:
- `transitions` er en funktion fra `StateSymbolPair` til en **mængde** af `State` objekter

# NFALambda.java

- Repræsentation som NFA.java, med én undtagelse:
- $\Lambda$  repræsenteres som `\uFFFF` (`= NFALambda.LAMBDA`)

# NFALambda.java

- Repræsentation som NFA.java, med én undtagelse:
- $\Lambda$  repræsenteres som `\uFFFF` (= `NFALambda.LAMBDA`)

# RegExp.java

- `RegExp(String, Alphabet)` – parser et regulært udtryk
- `toString()` – til udskrift af et parsed regulært udtryk
- `toNFALambda()` – konstruktionen fra Kleene's sætning del 1
- `simplify()` – simplificerer et parsed regulært udtryk, nyttig efter `FA.toRegExp()` (Kleene's sætning del 2)

# RegExp.java

- `RegExp(String, Alphabet)` – parser et regulært udtryk
- `toString()` – til udskrift af et parsed regulært udtryk
- `toNFALambda()` – konstruktionen fra Kleene's sætning del 1
- `simplify()` – simplificerer et parsed regulært udtryk, nyttig efter `FA.toRegExp()` (Kleene's sætning del 2)

# RegExp.java

- `RegExp(String, Alphabet)` – parser et regulært udtryk
- `toString()` – til udskrift af et parsed regulært udtryk
- `toNFALambda()` – konstruktionen fra Kleene's sætning del 1
- `simplify()` – simplificerer et parsed regulært udtryk, nyttig efter `FA.toRegExp()` (Kleene's sætning del 2)

# RegExp.java

- `RegExp(String, Alphabet)` – parser et regulært udtryk
- `toString()` – til udskrift af et parsed regulært udtryk
- `toNFALambda()` – konstruktionen fra Kleene's sætning del 1
- `simplify()` – simplificerer et parsed regulært udtryk, nyttig efter `FA.toRegExp()` (Kleene's sætning del 2)



# Minimering i dRegAut java-pakken

- "pseudo-kode":  
ufornet mellemtng mellem de matematiske definitioner og Java-koden

# FA.minimize()

```
FA minimize() {  
    phase 1: Remove unreachable states  
    phase 2a: Divide into accept/reject states  
    phase 2b: Iteration  
    phase 3: Build and return resulting minimal automaton n  
}
```

# FA.findReachableStates(), version 1

```
Set findReachableStates() {
```

```
     $R = \{q_0\}$ 
```

```
    done = false
```

```
    while not done do
```

```
        done = true
```

```
        for each  $q \in R$  do
```

```
            for each  $a \in \Sigma$  do
```

```
                 $p = \delta(q, a)$ 
```

```
                if  $p \notin R$  then
```

```
                    add  $p$  to  $R$ 
```

```
                    done = false
```

```
    return  $R$ 
```

```
}
```

## FA.findReachableStates(), version 2

Vi kan holde øje med hvilke tilstande der ikke er “færdigbesøgt” for at undgå at besøge hver tilstand flere gange:

```
Set findReachableStates() {
     $R = \{\}$ 
     $pending = \{q_0\}$ 
    while  $pending \neq \emptyset$  do
        pick and remove an element  $q$  from  $pending$ 
        add  $q$  to  $R$ 
        for each  $c \in \Sigma$  do
             $p = \delta(q, c)$ 
            if  $p \notin R$  then add  $p$  to  $pending$ 
    return  $R$ 
}
```

## FA.minimize phase 2a

- Define some ordering on the states  $Q$
- Declare marks: a set of pairs  $(p, q)$  where  $p, q \in Q$  and  $p < q$
- $marks = \emptyset$
- for each pair  $p, q \in Q$  where  $p < q$  do
  - if not  $(p \in A \Leftrightarrow q \in A)$  then
  - add  $(p, q)$  to marks

Mange muligheder for Java-representation af marks...

## FA.minimize() phase 2b

```
done = false
while not done do
  done = true
  for each pair  $p, q \in Q$  where  $p < q$  do
    if  $(p, q) \notin \text{marks}$  then
      for each  $a \in \Sigma$  do
         $r = \delta(p, a)$ 
         $s = \delta(q, a)$ 
        if  $r > s$  then swap  $r$  and  $s$ 
        if  $(r, s) \in \text{marks}$  then
          add  $(p, q)$  to marks
        done = false
```

Kunne gøres smartere med en pending worklist.

## FA.minimize(), phase 3

FA  $n$  = new FA with same alphabet as  $f$  but with no states or transitions yet  
 initialize empty maps  $old2new: f.Q \rightarrow n.Q$  and  $new2old: n.Q \rightarrow f.Q$

for each  $r \in f.Q$  in order do

  if  $(s, r) \in marks$  for every  $s < r$  then

    add a new state  $p$  to  $n.Q$

    add  $old2new(r) = p$  and  $new2old(p) = r$

    if  $r \in f.A$  then add  $p$  to  $n.A$

  else

    add  $old2new(r) = old2new(s)$

  if  $r = f.q_0$  then set  $n.q_0 = old2new(r)$

for each state  $p \in n$  do

  add  $n.\delta(p, c) = old2new(f.\delta(new2old(p), c))$  for each  $c \in \Sigma$

## Eksempel

```
Alphabet a = new Alphabet('0', '1');  
RegExp r = new RegExp("0+(1*+01*+10*+001*01)*0*", a);
```

```
NFALambda n1 = r.toNFALambda();  
NFA n2 = n1.removeLambdas();  
FA n3 = n2.determinize();
```

```
System.out.println("Før: " + n3.getNumberOfStates());  
FA n4 = n3.minimize();  
System.out.println("Efter: " + n4.getNumberOfStates());
```

Før: 13

Efter: 1



## Eksempel

```
Alphabet a = new Alphabet('0', '1');  
RegExp r = new RegExp("0+(1*+01*+10*+001*01)*0*", a);
```

```
NFALambda n1 = r.toNFALambda();  
NFA n2 = n1.removeLambdas();  
FA n3 = n2.determinize();
```

```
System.out.println("Før: " + n3.getNumberOfStates());  
FA n4 = n3.minimize();  
System.out.println("Efter: " + n4.getNumberOfStates());
```

Før: 13

Efter: 1

## Eksempel

```
Alphabet a = new Alphabet('0', '1');  
RegExp r = new RegExp("0+(1*+01*+10*+001*01)*0*", a);
```

```
NFALambda n1 = r.toNFALambda();  
NFA n2 = n1.removeLambdas();  
FA n3 = n2.determinize();
```

```
System.out.println("Før: " + n3.getNumberOfStates());  
FA n4 = n3.minimize();  
System.out.println("Efter: " + n4.getNumberOfStates());
```

Før: 13

Efter: 1

# Resume

- MyHill-Nerode-sætningen:
- endnu en karakteristik af de regulære sprog
- en algoritme til FA minimering
- en algoritme til at fjerne uopnåelige tilstande i en FA