

# INF1820: Introduksjon til språk-og kommunikasjonsteknologi

Fjerde forelesning

---

Lilja Øvrelid

6 februar, 2017

## FS-metoder

---

## Oversikt

- Såkalt “endelig tilstand (finite-state)”-teknologi er
  - kjapp og effektiv
  - nyttig for et antall språkteknologiske oppgaver
- Vi skal se nærmere på:
  1. Regulære uttrykk
  2. Endelige tilstandsmaskiner (“Finite state automata”, FSA)
  3. Regulære språk
- Vil se at regulære uttrykk og endelige tilstandsmaskiner er matematisk ekvivalente, men gir oss forskjellig innfallsvinkel på oppgaver

2

## Oversikt

- Perioden rett etter 2 verdenskrig → ca 1960: viktig for datalingvistikk
- Automata-teori basert på Turings arbeid (1936) om algoritmisk beregning
- Kleene: (1951, 1956): endelige automata og regulære uttrykk. Beviste ekvivalens
- Chomsky (1956): endelige tilstandsmaskiner som beskrivelser av naturlige språk
- Formell språkteori: algebra og mengdelære for å definere formelle språk

3

## Noen applikasjoneksemppler

- Tokenisering (**oblig1b**)
- Tekstprosessering (finne, erstatte)
  - Finne alle telefonnumre i en tekst, feks:
    - *Du kan også ringe Kundeservice på 815 22 040*
  - Finne flere tilgrensende instanser av samme ord i en tekst:
    - ...vi lurer bare på en en ting
- Bestemme språket i en setning/tekst: spansk eller polsk?
  - *Sytuacja na Bliskim Wschodzie jest napieta, szczególnie po wczorajszym ataku*

4

## Noen applikasjoneksemppler

- Validere felter i en database (datoer, e-postadresser, URL'er)
- Søk i et korpus etter lingvistiske mønstre
  - samle statistikk
- Tildel ordklasse til disse, selv om de ikke fins i ordboken:
  - *conurbation, cadence, disproportionality, Thatcherization*

5

- Utstrakt bruk innenfor:
  - akustisk modellering innenfor talegjenkjenning
  - informasjonshenting (“information extraction”), f.eks. navn på personer og firmaer
  - automatisk morfologisk analyse

## Regulære uttrykk

---

## Regulære uttrykk

- Et regulært uttrykk er en beskrivelse av en mengde strenger
- Finnes en rekke UNIX-verktøy (grep, sed), editorer (emacs) og programmeringsspråk (perl, python) som har funksjonalitet for regulære uttrykk
- Som alle formalismer har ikke regulære uttrykk noe språklig (lingvistisk) innhold, men kan snarere brukes til å referere til lingvistiske enheter

7

## Regulære uttrykk

To slags tegn ("characters"):

- **Bokstaver**
  - ethvert teksttegn er et RU og refererer til seg selv
- **Meta-tegn**
  - spesialtegn som lar deg kombinere RU på forskjellige måter
- Eksempel:
  - /a/ refererer til a
  - /a\*/ refererer til  $\epsilon$  eller a eller aa eller aaa eller ...

8

## Regulære uttrykk

Regulære uttrykk består av:

- strenger bestående av tegn:  
/b/, /INF1820/, /informatikk/
- Disjunksjon:
  - vanlig disjunksjon: /spise|ete/, /penge(r|ne)/
  - tegnklasser: / [Dd]en/, /m[ae]nn/, /bec[oa]me/
  - rekker ("ranges"): [A-Z], [0-9]
- negasjon:
  - [^b]
  - [^A-Z0-9]

9

## Regulære uttrykk

Regulære uttrykk består av (forts.):

- Tellere
  - opsjonalitet (0 eller 1): ?
    - /woodchucks?/
  - hvilket som helst antall (0 eller flere): Kleenes \*
    - /baaa\*!/
    - /[0-9][0-9]\*/
  - Minst en: +:
    - /baa+!/
    - /[0-9]+ kroner/
- "wildcard" for et hvilket som helst tegn: .
  - /beg.n/
  - brukes ofte sammen med stjerne: "hva som helst":
    - /beltedy.\*beltedy/

10

Regulære uttrykk består av (forts.):

- Ankere: spesielle tegn som forankrer det regulære uttrykket til spesifikt sted i strengen/teksten
  - `/^Den/` (^ - begynnelsen av linjen)
  - `/der\.$/` (\$) - slutten av linjen)

- Parenteser grupperer elementer sammen:
  - `/rev(ehi)?/`
  - `/(herre|dame)toalett/`

### Presedenshierarki:

1. Parentes ()
2. Tellere ? \* + {}
3. Sekvenser og ankere the `^my end$`
4. Disjunksjon |

- `/juhu*/` matcher *juhuuuuuuu* men ikke *juhujuhujuhu*

- Nyttige forkortelser:
  - $\backslash d = [0-9]$
  - $\backslash D = [^0-9]$
  - $\backslash w = [A-Za-z0-9]$
  - $\backslash W = [^A-Za-z0-9]$
  - $\backslash s = [ \backslash t \backslash n]$
  - $\backslash S = [^\backslash s]$

- **Et eksempel: Finne alle forekomster av *og* i en tekst**
  - $/og/$ 
    - Problem: *Og det ble kveld, og det ble morgen ...*
  - $/[Oo]g/$ 
    - Problem: *Togets rutetider*
  - $/\backslash s [Oo] g \backslash s /$
- Minimere antall falske positive (*Toget*) og falske negative (*Og*)



- Substitusjon
  - s/color/colour/
  - s/versjon\s2\.4/versjon\s2\.5/
  - s/(versjon\s\d\.)\d/\15/

- ELIZA (Weizenbaum 1966):

```
User:    Men are all alike
ELIZA:  IN WHAT WAY?
User:    They're always bugging us about something
ELIZA:  CAN YOU THINK OF AN EXAMPLE?
User:    Well, my boyfriend made me come here
ELIZA:  YOUR BOYFRIEND MADE YOU COME HERE.
User:    He says I am depressed much of the time
ELIZA:  I'M SORRY TO HEAR THAT YOU ARE DEPRESSED.
```

```
s/. * I am (depressed|sad) . *
```

```
/ I am sorry to hear that you are \1/
```

```
s/. * (all|always) . *
```

```
/ Can you think of an example /
```

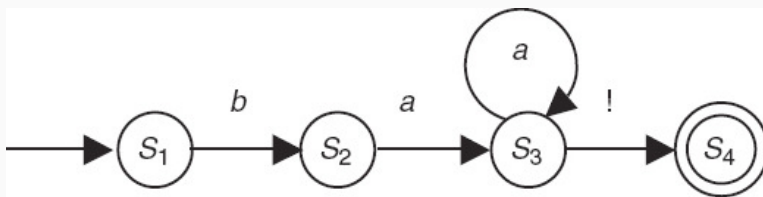
- Beskriv strengene som aksepteres av følgende regulære uttrykk
  - `/ab+a/`
  - `(ab)*`
  - `([^aeiou][aeiou])\1`
  - `\sdis[a-z]+\s`

## Endelige tilstandsmaskiner

---

## FSA'er

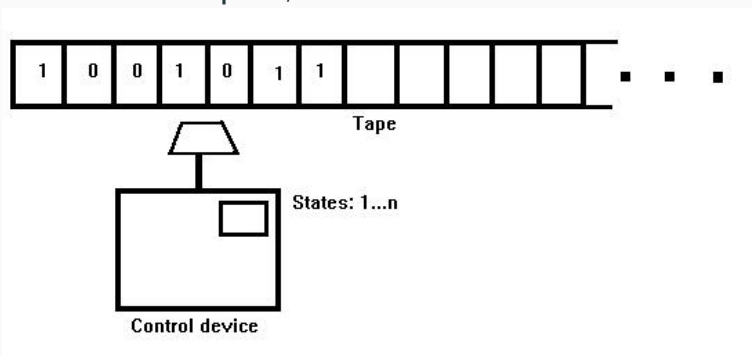
- Ethvert regulært uttrykk kan implementeres som en endelig tilstandsmaskin (og vice-versa)
- Måte å beskrive et spesielt formelt språk, nemlig regulære språk
- Hva er en endelig tilstandsmaskin (FSA)?
  - Abstrakt beregningsmaskin
  - Består av en mengde tilstander (noder i en graf), og en mengde transisjoner (kanter i en graf)
  - Tre typer tilstander: vanlig, start og slutt



18

## FSA'er

- Brukes for å gjenkjenne/akseptere strenger:
  - Antar en "tape", der input-symbolene leses av celle etter celle
  - Maskinen starter i starttilstanden
  - For hvert symbol på tape'en, forsøker tilsvarende transisjon i maskinen
  - Dersom ingen mulig transisjon: avvis
  - Når strengen er ferdig lest, sjekk om tilstanden er slutttilstand
  - Ja: aksepter, nei: avvis



19

## En liten digresjon

A α	alpha	N ν	nu
B β	beta	Ξ ξ	ksi
Γ γ	gamma	Ο ο	omicron
Δ δ	delta	Π π	pi
E ε	epsilon	Ρ ρ	rho
Z ζ	zeta	Σ σς	sigma
H η	eta	Τ τ	tau
Θ θ	theta	Υ υ	upsilon
I ι	iota	Φ φ	phi
K κ	kappa	X χ	chi
Λ λ	lambda	Ψ ψ	psi
Μ μ	mu	Ω ω	omega

Greek alphabet chart © by de Traci Regula; licensed to About.com

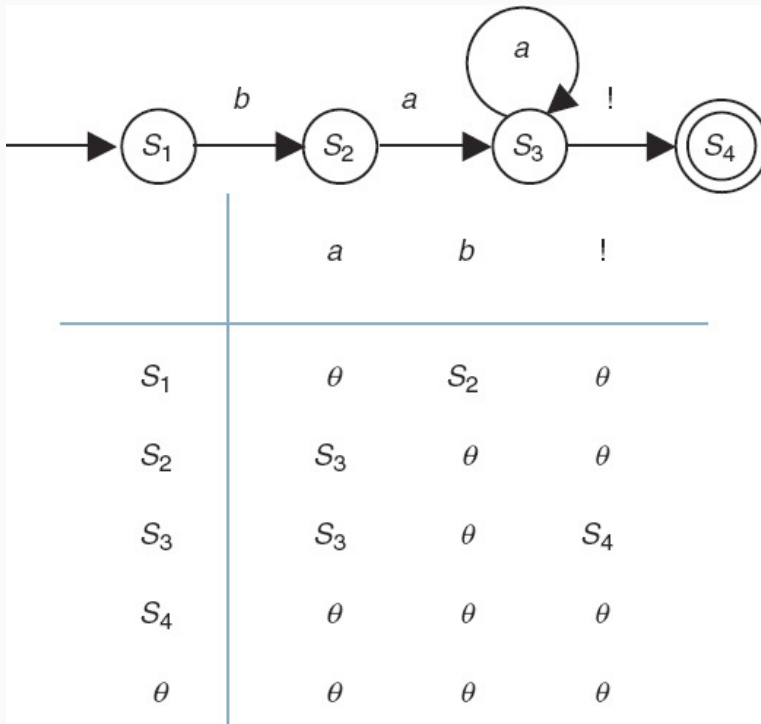
20

## Formell definisjon

- En endelig tilstandsmaskin er definert ved:
  - $Q = \{q_0, q_1, q_2, \dots, q_{n-1}\}$ : en endelig mengde tilstander
  - $\Sigma$ : et endelig alfabet
  - $q_0$ : en starttilstand
  - $F$ : mengden av slutttilstander,  $F \subseteq Q$
  - $\delta(q_i, i)$ : en funksjon som gitt en tilstand  $q_i$  og et inputsymbol  $i \in \Sigma$ , returnerer en ny tilstand  $q' \in Q$

21

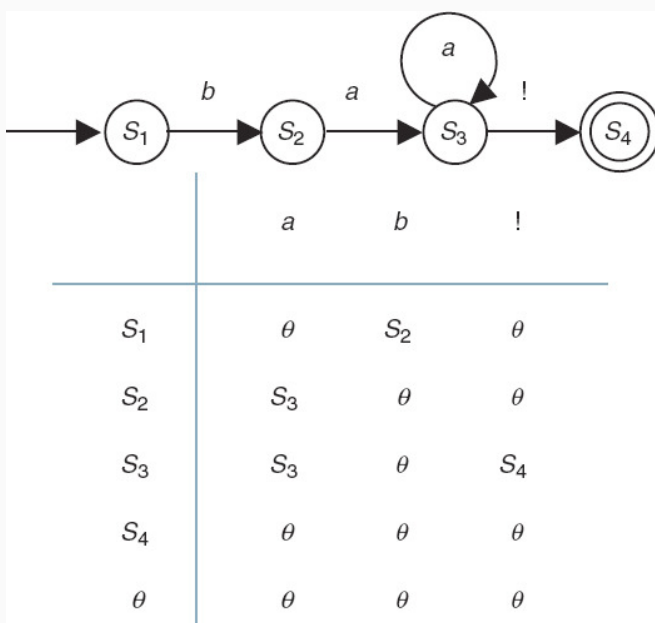
- Kan også spesifiseres ved en transisjonstabell:



22

- For saueprat-eksempelet:

- $Q = \{s_1, s_2, s_3, s_4\}$
- $\Sigma = \{a, b, !\}$
- $F = s_4$
- $\delta(q, i)$  er gitt ved transisjonstabellen



23

- FSA som gjenkjenner strenger på formen  $[ab]^+$
- Feks  $a, b, ab, ba, aab, bab, aba, bba, \text{etc.}$
- FSA definert som:

	a	b
$S_0$	$S_1$	$S_1$
$S_1$ :	$S_1$	$S_1$

- Hva blir  $Q, \Sigma$  og  $F$ ?
- Hvordan ser automaten ut?

24

- FSA for enkle (engelske) substantivfraser
- $d$  for artikler (“determiners”),  $a$  for adjektiver og  $n$  for substantiver
  - *the car*
  - *the green car*
  - *the fast green car*
  - ...

	$d$	$a$	$n$
$S_0$	$S_1$	$\emptyset$	$\emptyset$
$S_1$	$\emptyset$	$S_1$	$S_2$
$S_2$ :	$\emptyset$	$\emptyset$	$\emptyset$

- Hva med *fast cars* eller *cars*?

25

- Ikke-deterministiske FSAer

- En FSA er ikke-deterministisk dersom den
  - inneholder mer enn ett valg for minst en node
  - disse valgene kan involvere tomme ( $\epsilon$ -)transisjoner
  - for hvertfall en tilstand og ett symbol fins det mer enn en transisjon som passer

	$b$	$a$	$!$
$S_0$	$S_1$	$\emptyset$	$\emptyset$
$S_1$	$\emptyset$	$S_1, S_2$	$\emptyset$
$S_2$	$\emptyset$	$\emptyset$	$S_3$
$S_3$ :	$\emptyset$	$\emptyset$	$\emptyset$

- Hvorfor **endelige** ("finite")?

- Antall tilstander er bestemt på forhånd (transisjonstabellen)
- Derfor bruker maskinen begrenset minne
  - Hva den gjør ved hvert skritt bestemmes av transisjonstabellen
  - Tilstanden ved ethvert tidspunkt reflekterer prosesseringsrekkefølgen
- Klasser av formelle språk som ikke er regulære krever ekstra minne for å holde styr på tidligere informasjon, feks såkalte center-embedding konstruksjoner (mer om dette senere)

# Regulære språk

---

## Formell språkteori

- Et formelt språk er en mengde strenger
  - Et endelig alfabet  $\Sigma$  og noen operasjoner for å kombinere strenger
  - Regulære språk er den enkleste klassen av formelle språk
    - Klassen av språk som kan defineres av regulære uttrykk
    - Klassen av språk som kan gjenkjennes av FSAer

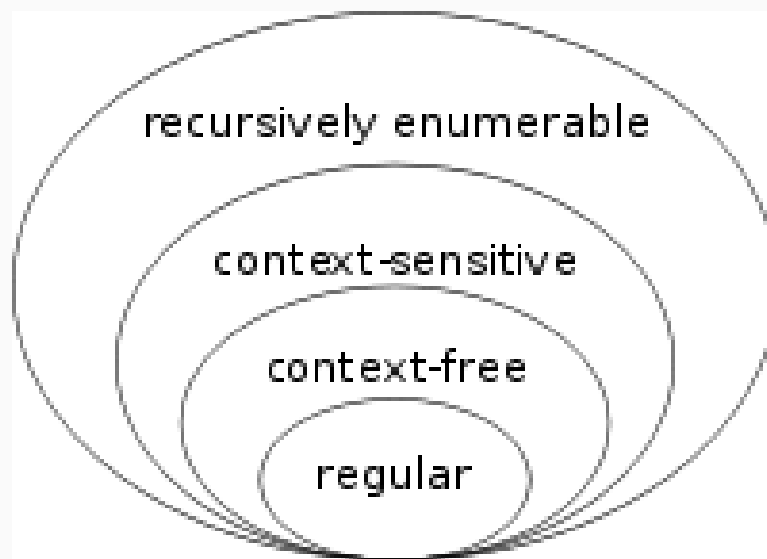


**Komputasjonell kompleksitet:** hvilken uttrykkskraft (og ressurser som kreves for å prosessere) klasser av formelle språk

**Lingvistisk kompleksitet:** hva gjør noen konstruksjoner eller setninger vanskeligere å forstå

- *This is the dog, that worried the cat, that killed the rat, that ate the malt, that lay in the house that Jack built.*
- *This is the malt that the rat that the cat that the dog worried killed ate.*

- et hierarki av klasser av språk (sett på som mengder av strenger), ordnet etter kompleksitet
- språkene i én klasse inkluderer språkene i lavere klasser
- sammenheng mellom klassen av språk og hvilke formelle regelsystemer man kan bruke for å generere språkene



- Regulære språk: endelig tilstandsmaskin (FSA)
- Kontekstfrie språk: kontekstfrie grammatikker (CFG)
- Kontekstsensitive språk: Turingmaskin med endelig tape
- Turing-ekvivalente språk: Turingmaskin  
formell modell som kan beskrive logikken bak enhver algoritme  
<http://www.youtube.com/watch?v=cYw2ewo06c4>

## Hva med naturlige språk?

- Chomsky-hierarkiet har også blitt brukt for å forstå naturlige språks kompleksitet
- og ikke minst hvilke modeller vi kan bruke til å prosessere dem
- Chomsky (1957):  
“English is not a regular language”  
Om kontekstfrie språk: “I do not know whether or not English is itself literally outside the range of such analyses”

33

## Formell språkteori

- Hva slags uttrykk er ikke regulære?
- I naturlige språk: feks såkalt center-embedding
  1. The dog died
  2. The cat the dog saw died
  3. The cat the dog the mouse bit saw died
  4. ...
- $(\text{the noun})^n (\text{transitivt-verb})^{n-1} \text{ intransitivt-verb}$
- Lignende vil være regulære:
  - $A^*B^*$  died

34

- Men representerer dette faktisk språkbruk?
- Autentisk eksempel:
  - *[When the pain, [which nobody [who has not experienced it] can imagine], finally arrives], they can be taken aback by its severity.*

- Hvordan kan vi karakterisere klassen av regulære språk?
  - $\epsilon$  er den tomme strengen
  - $\emptyset$  er den tomme mengden
  - $\Sigma$  er et alfabet (symboler)

- Klassen av regulære språk over  $\Sigma$  kan defineres formelt som:
  - $\emptyset$  er et regulært språk
  - $\forall a \in \Sigma \cup \epsilon, \{a\}$  er et regulært språk
  - Dersom  $L_1$  og  $L_2$  er regulære språk, da er følgende språk også regulære:
    1.  $L_1 \cdot L_2$  (konkatenering)
    2.  $L_1 \cup L_2$  (union eller disjunksjon)
    3.  $L_1^*$  (Kleenes closure)
  - Alle regulære uttrykk faller inn under dette
  - $[]$  er disjunksjon

- De regulære språkene er lukket under:
  - **differanse:**  $L_1 - L_2$ : mengden strenger som er i  $L_1$  men ikke i  $L_2$
  - **snitt:**  $L_1 \cap L_2$ : mengden strenger som er i både  $L_1$  og  $L_2$
  - **komplement:** Dersom  $L_1$  er et regulært språk så er  $\Sigma^* - L_1$ , mengden av alle mulige strenger som ikke er i  $L_1$ , også det
  - **reversering:** Dersom  $L_1$  er et regulært språk så er  $L_1^R$ , mengden av reverserte strenger fra  $L_1$  også det

- Egenskapene kan brukes til å bevise medlemskap i en formell språklig klasse
- Feks: Engelsk er ikke regulært
- $L_{ce} = (\text{the noun})^n (\text{transitivt-verb})^{n-1} \text{ intransitivt-verb}$ 
  - $L_{ce}$  er snittet av det naturlige språket engelsk med den regulære mengden  $L_{reg} = A^*B^*$  intransitivt verb
  - da regulære språk er lukket under snitt og  $L_{reg}$  er et regulært språk
  - så ville snittet mellom engelsk og  $L_{reg}$  være regulært dersom engelsk var regulært
  - engelsk er altså ikke regulært