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

Oversikt

- Perioden rett etter 2 verdenskrig $\rightarrow$ 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

Noen applikasjonseksempler

- 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, szczegolnie po wczorajszym ataku*

Noen applikasjonseksempler

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

Noen applikasjonseksempler

- 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 progammeringsspråk (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

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 ϵ eller a eller aa eller aaa eller ...

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]`

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":
  - `/beltedyr.*beltedyr/`

Regulære uttrykk

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)

Regulære uttrykk

- 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*

Regulære uttrykk

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

Regulære uttrykk

- **Et eksempel: Finne alle forekomster av *og* i en tekst**
  - /og/
    - Problem: *Og det ble kveld, og det ble morgen . . .*

Regulære uttrykk

- **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*

Regulære uttrykk

- **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*
  - `/\s[Oo]g\s/`
- Minimere antall falske positiver (*Toget*) og falske negativer (*Og*)

Regulære uttrykk

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

Regulære uttrykk

- 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.

Regulære uttrykk

- 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 /
```

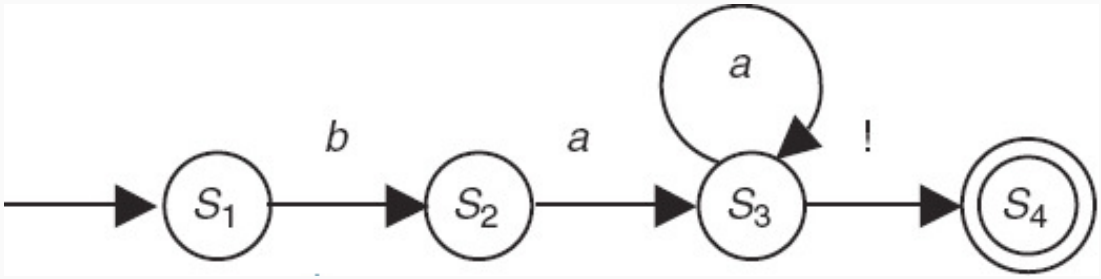
QUIZ

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

Endelige tilstandsmaskiner

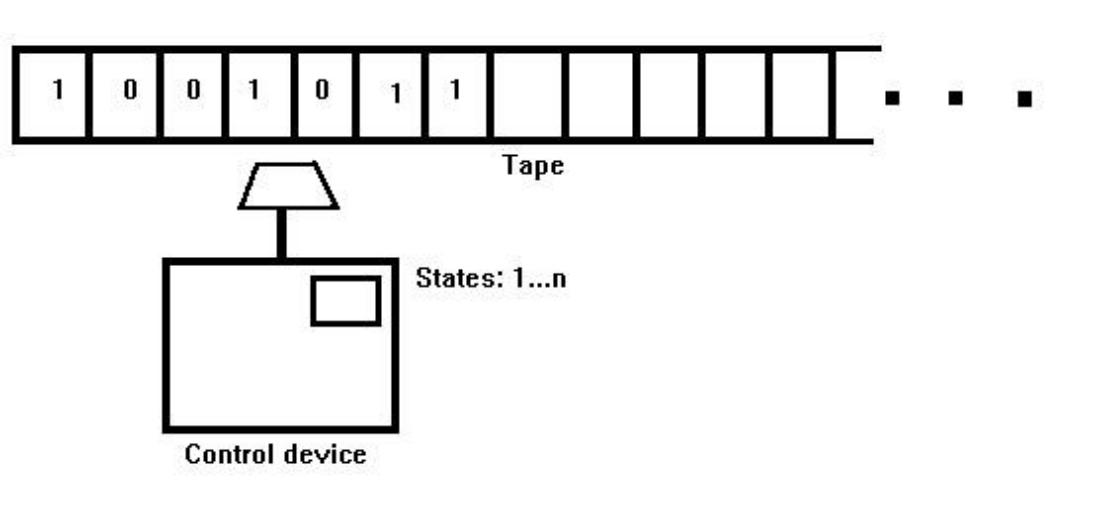
FSA'er

- 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



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 sluttilstand
  - Ja: aksepter, nei: avvis



En liten digresjon

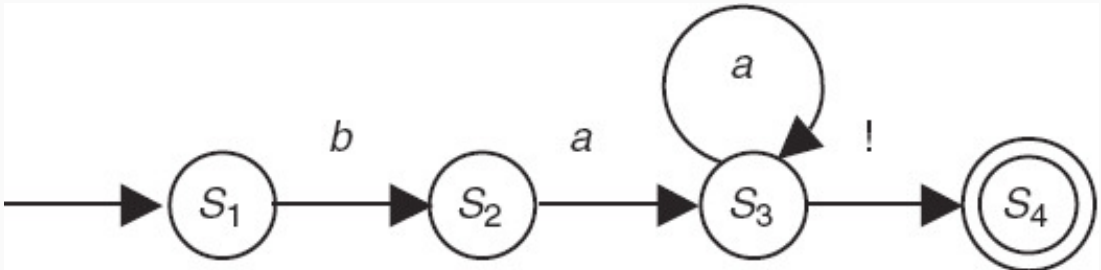
A α	alpha	N ν	nu
B β	beta	Ξ ξ	ksi
Γ γ	gamma	O o	omicron
Δ δ	delta	Π π	pi
E ε	epsilon	P ρ	rho
Z ζ	zeta	Σ σς	sigma
H η	eta	T τ	tau
Θ θ	theta	Y υ	upsilon
I ι	iota	Φ φ	phi
K κ	kappa	X χ	chi
Λ λ	lambda	Ψ ψ	psi
Mμ	mu	Ω ω	omega

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

Formell definisjon

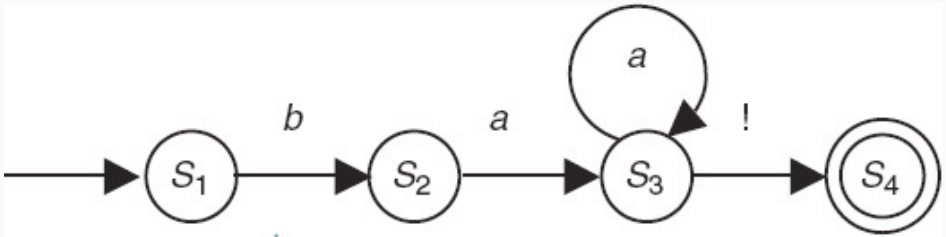
- En endelig tilstandsmaskin er definert ved:
  - $Q = \{q_0, q_1, q_2, \ldots, q_{n-1}\}$: en endelig mengde tilstander
  - Σ: et endelig alfabet
  - q_0: en starttilstand
  - F: mengden av sluttilstander, $F \subseteq Q$
  - $\delta(q_i, i)$: en funksjon som gitt en tilstand q_i og et inputsymbol $i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

- Kan også spesifiseres ved en transisjonstabell:



	a	b	!
S_1	θ	S_2	θ
S_2	S_3	θ	θ
S_3	S_3	θ	S_4
S_4	θ	θ	θ
θ	θ	θ	θ

- 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



	a	b	!
S_1	θ	S_2	θ
S_2	S_3	θ	θ
S_3	S_3	θ	S_4
S_4	θ	θ	θ
θ	θ	θ	θ

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

	a	b
S_0	S_1	S_1
S_1:	S_1	S_1

- Hva blir Q, Σ og F?
- Hvordan ser automaten ut?

- 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	ø	ø
S_1	ø	S_1	S_2
S_2:	ø	ø	ø

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

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

	b	*a*	!
S_0	S_1	ø	ø
S_1	ø	S_1,S_2	ø
S_2	ø	ø	S_3
S_3:	ø	ø	ø

- 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 Σ 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

Kompleksitet

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

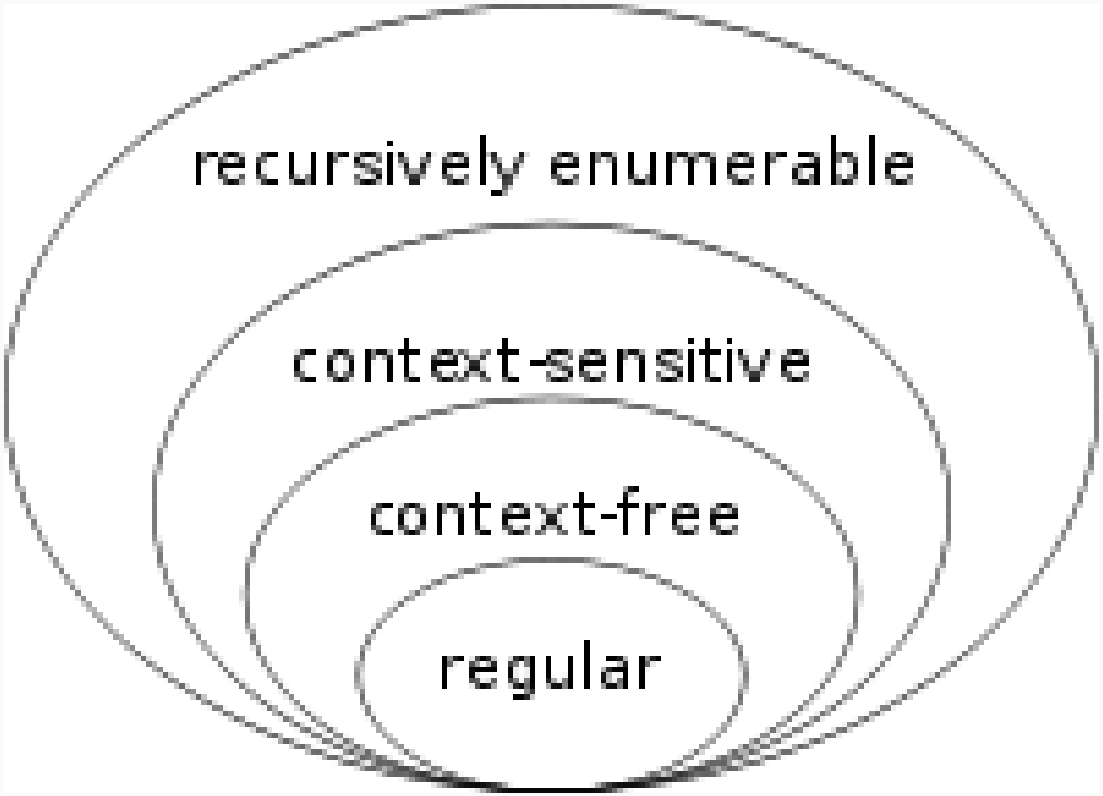
Lingvistisk kompleksitet: hva gjør noen konstruksjoner eller setninger vanskeligere å forstå

- *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.*

Chomsky-hierarkiet

- 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

Chomsky-hierarkiet



Chomsky-hierarkiet

- 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=cYw2ewoO6c4`

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"

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. ...
- (the noun)^n (transitivt-verb)$^{n-1}$ intransitivt-verb
- Lignende vil være regulære:
  - A*B* died

Formell språkteori

- 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.*

Regulære språk

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

Regulære språk

- Klassen av regulære språk over Σ kan defineres formelt som:
  - ø 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

Regulære språk

- 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

Regulære språk

- Egenskapene kan brukes til å bevise medlemskap i en formell språklig klasse
- Feks: Engelsk er ikke regulært
- $L_{ce} = (\text{the noun})^n\ (\text{transitivt-verb})^{n-1}$ 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