Syntaks og semantik

Lektion 1

6 februar 2007

I dag

1. Introduktion
2. Ord og sprog
3. Regulære udtryk

Introduktion til kurset

Kursets emne

Grundlæggende aspekter ved programmeringssprog:

- Hvordan kan vi beskrive hvordan et sprog ser ud? (dets form)
- Hvordan kan vi beskrive hvordan et sprog skal forstås? (dets adfærd)

Kursets emne

Grundlæggende aspekter ved programmeringssprog:

- Hvordan kan vi beskrive hvordan et sprog ser ud? (dets form)

  Syntaks:
  - regulære sprog, endelige automater, regulære udtryk
  - kontekst-frie sprog, push-down-automater, kontekst-frie grammatikker
- Hvordan kan vi beskrive hvordan et sprog skal forstås? (dets adfærd)

  Semantik:
  - operationel semantik

Kursets indhold

Syntaks – regulære sprog:

1. Introduktion; sprog; regulære udtryk
2. Endelige automater
3. Regulære udtryk
4. Sprog der *ikke* er regulære

Syntaks – kontekstfrie sprog:

5. Kontekstfrie grammatikker
6. Pushdown-automater
7. Sprog der *ikke* er kontekstfrie

Kursets indhold

Semantik:

8. Introduktion til operationel semantik
9. Operationelle semantikker for **Bims**
10. Udvidelser af **Bims**
11. Blokke og procedurer
12. Parametermekanismer
13. Objektorientede sprog

Teoretisk grundlag:

14. Rekursive definitioner

Hvad kan jeg bruge det til?

- Vil jeg lære et nyt programmeringssprog?
  Nej.
- Skal vi se nogen smarte algoritmer?
  Nej.
- Vil jeg blive bedre til at programmere?
  Forhåbentlig.
- Vil jeg opnå større forståelse for hvordan programmeringssprog er opbygget?
  Ja.
- Vil jeg opnå større forståelse for hvilke problemer computere kan løse?
  Til dels.
- Vil jeg blive bedre til at forstå teorien bag programmering?
  Ja.

Kursets form

- 8:15 – 10:00: Forelæsning
- 10:10 – 12:00: Opgaveregning

Kursets form

- 8:15 – 10:00: Forelæsning
  - i B3-104 og i NOVIs auditorium
  - de sidste 5 minutter bruges på quiz
  - Læs stoffet *hurtigt* inden forelæsningen, så I ved hvad det handler om, og læs det *grundigt* igen bagefter, så I er sikre på at have forstået det.
  - Kursets emner bygger ovenpå hinanden, så hvis der er noget man misser, er det svært at finde tilbage igen!
- 10:10 – 12:00: Opgaveregning
  - i grupperum
  - to større afleveringsopgaver
  - Forvent ikke at kunne forstå stoffet uden at regne opgaver.
  - Studerende der ikke regner opgaver, kan ikke opholde sig i grupperummet under opgaveregningen.

Quiz

- sidste 5 minutter af hver forelæsning
- små opgaver, multiple choice
- afleveres anonymt
- I kan se hvad dagens vigtige pointer var, og om I har forstået dem
- Jeg kan se om jeg formår at formidle de vigtige pointer

Afleveringsopgaver

- to gennemgående opgaver som I skal bruge en del af opgaveregningen på, *hver gang*
- afleveres til mig, kommenteres bagefter først af jeres kolleger og til sidst af mig
- vil være del af eksamenspensum
- kan for PE-studerende erstattes af tilsvarende opgaver der har *relation til projektet*
- Syntaksopgave
- Semantikopgave

Afleveringsopgaver

- to gennemgående opgaver som I skal bruge en del af opgaveregningen på, *hver gang*
- afleveres til mig, kommenteres bagefter først af jeres kolleger og til sidst af mig
- vil være del af eksamenspensum
- kan for PE-studerende erstattes af tilsvarende opgaver der har *relation til projektet*
- Syntaksopgave
  - tilgængelig nu
  - afleveres senest 12 marts
  - evt. erstatnings*opgave* skal indleveres senest 16 februar
- Semantikopgave
  - vil blive offentliggjort senest 6 marts

Bøger

- Michael Sipser: *Introduction to the Theory of Computation*, Second Edition, PWS Publishing Co. 2005.
  Brug ikke ældre udgaver, der er lavet for meget om!
- Hans Hüttel: *Pilen ved træets rod*, Aalborg Universitet 2007.
- begge klar i boghandelen nu
- Sipser skal vi bruge *nu*
- Hüttel først i marts

Hjemmeside

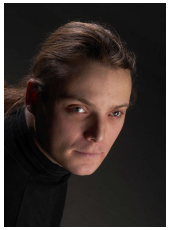
http://sands07.twoday.net

- slides
- opgaver
- andet materiale
- interessante links
- RSS-feed
- kommentarfunktion!

Eksamen

- mundtlig, 20min
- et antal spørgsmål kendt på forhånd – træk ét af dem
- 20min forberedelsestid
- ekstern censor, karakter
- pensum og spørgsmål fastlægges ved tredjesidste kursusgang
- afleveringsopgaver indgår som hver deres spørgsmål

Uli Fahrenberg
underviser
uli@cs.aau.dk



Jens Alsted
hjælpelærer
alsted@cs.aau.dk

Hans Hüttel
lærebogsforfatter
hans@cs.aau.dk

Ord og sprog

- Σ – en endelig mængde af bogstaver eller symboler – et alfabet
- et ord: en endelig følge af bogstaver – normalt skrevet uden parenteser eller kommaer
- eksempel: $\Sigma = \{0, 1\}$
  ord over Σ: f.x. $0, 1, 00, 01, 1001010110101$
- eksempel: $\Sigma = \{a, b, c, d, r\}$
  ord over Σ: f.x. $a, b, c, d, r, abba, abracadabra$
- eksempel: $\Sigma = \{\text{gik}, \text{jeg}, \text{land}, \text{mig}, \text{og}, \text{over}, \text{sø}, \text{to}, \text{vi}\}$
  ord over Σ: f.x. “jeg og mig og vi to”
  eller “jeg gik mig over sø og land”
- eksempel: $\Sigma = \{\text{else}, \text{if}, \text{then}, \mathit{Exp}, \mathit{Stm}\}$
  ord over Σ: f.x. “if *Exp* then *Stm* else *Stm*”

- et sprog: en mængde af ord (endelig eller uendelig)
- mængden af *alle* ord over et alfabet Σ skrives Σ^*
  (den er altid uendelig (medmindre Σ er tom . . .))
- eksempel: $\Sigma = \{0, 1\}$
  $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \dots\}$
- ε – det *tomme* ord; ordet af længden 0
- længden af et ord: $|w|$ = antallet af bogstaver i ordet
- $\emptyset$ – det *tomme sprog*; mængden uden indhold
- Bemærk: ε er et *ord*, $\emptyset$ er et *sprog*. Og $\{\varepsilon\} \neq \emptyset$

- at sammensætte ord: $abe \circ kat = abekat$
  (svarer til at gange tal sammen, men *ikke kommutativt!*)
  ($\circ$-tegnet udelades de fleste gange)
- ε er identiteten: $w \circ \varepsilon = w$ og $\varepsilon \circ w = w$ for alle ord w.
  (ligesom tallet 1 er identiteten for multiplikation)
- gentagen sammensættelse skrives som potenser: $a^2 = aa$, $a^3 = aaa$, $a^9 = aaaaaaaaa$ etc.

Hvis L_1 og L_2 er sprog over et alfabet Σ, kan vi danne

- foreningsmængden $L_1 \cup L_2 = \{w \mid w \in L_1 \text{ eller } w \in L_2\}$
  – sproget med alle de ord der er i L_1 eller L_2
- fællesmængden $L_1 \cap L_2 = \{w \mid w \in L_1 \text{ og } w \in L_2\}$
  – sproget med alle de ord der er i L_1 og L_2
- sammensætningen
  $L_1 \circ L_2 = \{w_1 \circ w_2 \mid w_1 \in L_1 \text{ og } w_2 \in L_2\}$
  – sproget med alle de ord der er sammensætninger af et ord fra L_1 efterfulgt af et ord fra L_2
- stjernen $L_1^* = \{w_1 \circ w_2 \circ \cdots \circ w_k \mid \text{alle } w_i \in L_1\}$
  – sproget med alle de ord der er sammensætninger af vilkårligt mange ord fra L_1
  – indeholder ε: det tomme ord = sammensætningen af 0 ord fra L_1 ...

Regulære udtryk

Vi kan beskrive sprog som mængder: (her lader vi $\Sigma = \{a, b\}$)

- $L_1 = \{a, b, ab\}$ (et *endeligt* sprog)
- $L_2 = \{a^n \mid n \in \mathbb{N}\}$ – alle ord der indeholder kun a, af vilkårlig længde
- $L_3 = \{a^n b a^m \mid n, m \in \mathbb{N}\}$ – alle ord der indeholder præcist ét b
- $L_4 = \{a^n b^n\}$ – alle ord der indeholder et antal a og så *samme* antal b

eller ved hjælp af regulære udtryk:

- $L_1 = a \cup b \cup ab$
- $L_2 = a^*$
- $L_3 = a^* \circ b \circ a^*$
- $L_4 = ???$
  (vi skal senere se at L_4 *ikke* kan beskrives ved regulære udtryk!)

Definition 1.52: Et regulært udtryk over et alfabet Σ er et udtryk af formen

1. a for et $a \in \Sigma$,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, hvor R_1 og R_2 er regulære udtryk,
5. $(R_1 \circ R_2)$, hvor R_1 og R_2 er regulære udtryk, eller
6. (R_1^*), hvor R_1 er et regulært udtryk.

– en induktiv (eller rekursiv) definition: 1. til 3. giver de basale *byggesten*, og 4. til 6. giver *byggeregler* til hvordan man kan sætte ting sammen.
– parenteserne udelades ofte

Definition 1.52: Et regulært udtryk over et alfabet Σ er et udtryk af formen

1. a for et $a \in \Sigma$,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, hvor R_1 og R_2 er regulære udtryk,
5. $(R_1 \circ R_2)$, hvor R_1 og R_2 er regulære udtryk, eller
6. (R_1^*), hvor R_1 er et regulært udtryk.

Eksempler (med $\Sigma = \{a, b\}$):
a, b, $a \cup b$, $(a \cup b)^*$, $(a \cup b)^* \circ b$, $((a \cup b)^* \circ b)^*$

Definition 1.52, fortsat: Sproget, som et regulært udtryk R beskriver, betegnes $[\![R]\!]$ og er defineret som følger:

1. $[\![a]\!] = \{a\}$
2. $[\![\varepsilon]\!] = \{\varepsilon\}$
3. $[\![\emptyset]\!] = \emptyset$
4. $[\![R_1 \cup R_2]\!] = [\![R_1]\!] \cup [\![R_2]\!]$
5. $[\![R_1 \circ R_2]\!] = [\![R_1]\!] \circ [\![R_2]\!]$
6. $[\![R_1^*]\!] = [\![R_1]\!]^*$

Definition 1.52, fortsat: Sproget, som et regulært udtryk R beskriver, betegnes $[\![R]\!]$ og er defineret som følger:

1. $[\![a]\!] = \{a\}$
2. $[\![\varepsilon]\!] = \{\varepsilon\}$
3. $[\![\emptyset]\!] = \emptyset$
4. $[\![R_1 \cup R_2]\!] = [\![R_1]\!] \cup [\![R_2]\!]$
5. $[\![R_1 \circ R_2]\!] = [\![R_1]\!] \circ [\![R_2]\!]$
6. $[\![R_1^*]\!] = [\![R_1]\!]^*$

– Sipser skriver $L(R)$ i stedet for $[\![R]\!]$. Jeg vil bruge begge notationer

Udvidelser:

- $\Sigma = a_1 \cup a_2 \cup \cdots \cup a_n$ (hvis sproget er $\Sigma = \{a_1, a_2, \ldots, a_n\}$)
- $R^+ = R \circ R^*$

Eksempler (1.53): (for $\Sigma = \{0,1\}$)

1. $[\![0^*10^*]\!] =$ sproget med alle ord der indeholder symbolet 1 *præcist* én gang
2. $[\![\Sigma^*0\Sigma^*]\!] =$ sproget med alle ord der indeholder symbolet 1 *mindst* én gang
4. $[\![(01^+)^*]\!] =$ sproget af alle ord hvori ethvert 0 efterfølges af mindst ét 1
5. $[\![(\Sigma\Sigma)^*]\!] = \{w \mid |w| \text{ er et lige tal}\}$
8. $[\![0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1]\!] = \{w \mid$ start- og slutsymbolet i w er ens$\}$
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
X. $(0 \cup 1)^* = (0^*1^*)^*$

- regulære udtryk bruges i praksis bl.a. til analyse og editering af tekst
- i dag ser vi på `grep` og `sed`
- senere skal vi også bruge `flex`
- ellers bruges de i `ed`, `vi`, `emacs` og stort set alle skriptsprog (`perl`, `PHP`, `bash` etc.)
- Men pas på! Der er store forskelle i syntaksen mellem de forskellige programmers "regulære udtryk," og nogen gange dækker "regulære udtryk" også over ting der faktisk *ikke* er regulære i teknisk forstand. (Fy, `perl`!)

Sipser	`grep`, `sed` etc.	kommentarer
a	`a`	
Σ	`.`	
$\varepsilon, \emptyset$		findes ikke
$R_1 \cup R_2$	R_1 `|` R_2	
$R_1 \circ R_2$	$R_1 R_2$	
R^*	R`*`	
R^+	R`+`	
$(,)$	`\(,\)`	
	`[abcd]`	svarer til `a` $\cup$ `b` $\cup$ `c` $\cup$ `d`
	`[:alpha:]`	matcher alle bogstaver
	`[:digit:]`	matcher alle cifre
	`^`	negation. Rigtige regexps indeholder *ikke* negation!

se også `man sed` eller `man 7 regex`

Eksempler:

- `grep 'In' regexp-ex.txt`
- `grep 'In\(d\|t\)' regexp-ex.txt`
- `grep 'In\([dt]\)' regexp-ex.txt #det samme`
- `grep '[HJ][ae]ns' regexp-ex.txt`
- `grep 'a.*e.*e' regexp-ex.txt`
- `grep 'a[^ ]*e[^ ]*e' regexp-ex.txt`
- `sed 's:a:u:g' regexp-ex.txt`
- `sed 's:  *: :g' regexp-ex.txt`
- `sed 's: :\n:g' regexp-ex.txt`