Syntaks og semantik

Lektion 5

19 februar 2007

Regulære sprog

1. Bogstaver, ord og sprog (13f, 44)
2. Deterministiske endelige automater (35f, 40)
3. Nondeterministiske endelige automater (53–56)
4. Lukningsegenskaber (45f, 58–63, 85)
5. Regulære udtryk (64, 67, 69–74)
6. Oversigt
7. Ikke-regulære sprog (77–80)
8. Anvendelser
9. At forstå
10. En anden bog

- alfabet: en endelig mængde, normalt betegnet Σ
- bogstav / tegn / symbol: et element i Σ
- ord / streng: en endelig følge $(a_1, a_2, \ldots, a_k)$ af bogstaver. Normalt skrevet uden parenteser og komma: $a_1 a_2 \ldots a_k$
- ε: det tomme ord (med 0 bogstaver)
- at sammensætte ord: $abe \circ kat = abekat$
- ε er identiteten for $\circ$: $w \circ \varepsilon = \varepsilon \circ w = w$ for alle ord w

- Sprog (over Σ): en mængde af ord med bogstaver fra Σ
- $\emptyset$: det tomme sprog
- Σ^*: sproget bestående af *alle* ord over Σ

$\Rightarrow$ L er et sprog over Σ hvis og kun hvis $L \subseteq \Sigma^*$

- Givet sprog $L_1, L_2 \subseteq \Sigma^*$, da kan vi danne sprogene
  - $L_1 \cup L_2 = \{w \mid w \in L_1 \text{ eller } w \in L_2\}$ foreningsmængden
  - $L_1 \circ L_2 = \{w_1 \circ w_2 \mid w_1 \in L_1 \text{ og } w_2 \in L_2\}$ sammensætningen
  - $L_1^* = \{w_1 \circ w_2 \circ \cdots \circ w_k \mid \text{alle } w_i \in L_1\}$ stjernen
- Disse 3 operationer kaldes de regulære operationer på sprog.
- Vi kan også danne andre sprog; de vigtigste andre operationer:
  - $L_1 \cap L_2 = \{w \mid w \in L_1 \text{ og } w \in L_2\}$ fællesmængden
  - $\bar{L}_1 = \Sigma^* \setminus L_1 = \{w \in \Sigma^* \mid w \notin L\}$ komplementet

- Definition 1.5: En deterministisk endelig automat (DFA) er en 5-tupel $M = (Q, \Sigma, \delta, q_0, F)$, hvor delene er
  1. Q : en endelig mængde af tilstande
  2. Σ : input-alfabetet
  3. $\delta : Q \times \Sigma \to Q$: transitionsfunktionen
  4. $q_0 \in Q$: starttilstanden
  5. $F \subseteq Q$: mængden af accepttilstande
- M siges at acceptere et ord $w \in \Sigma^*$ hvis der findes $w_1, w_2, \ldots, w_k \in \Sigma$ og $r_0, r_1, \ldots, r_k \in Q$ således at $w = w_1 w_2 \ldots w_k$ og
  1. $r_0 = q_0$,
  2. $r_{i+1} = \delta(r_i, w_{i+1})$ for alle $i = 0, 1, \ldots, k-1$, og
  3. $r_k \in F$.
- Sproget som genkendes af M er $[\![M]\!] = \{w \in \Sigma^* \mid M \text{ accepterer } w\}$.
- Definition 1.16: Et sprog siges at være regulært hvis der findes en DFA der genkender det.

- Definition 1.37: En nondeterministisk endelig automat (NFA) er en 5-tupel $M = (Q, \Sigma, \delta, q_0, F)$, hvor delene er
  1. Q : en endelig mængde af tilstande
  2. Σ : input-alfabetet
  3. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$: transitionsfunktionen
  4. $q_0 \in Q$: starttilstanden
  5. $F \subseteq Q$: mængden af accepttilstande
- M siges at acceptere et ord $w \in \Sigma^*$ hvis der findes $w_1, w_2, \ldots, w_k \in \Sigma \cup \{\varepsilon\}$ og $r_0, r_1, \ldots, r_k \in Q$ således at $w = w_1 w_2 \ldots w_k$ og
  1. $r_0 = q_0$,
  2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for alle $i = 0, 1, \ldots, k-1$, og
  3. $r_k \in F$.
- Sproget som genkendes af M er $[\![M]\!] = \{w \in \Sigma^* \mid M \text{ accepterer } w\}$.

- Enhver DFA er også en NFA.
- Sætning 1.39: Til enhver NFA findes der en DFA der genkender samme sprog.
- Bevis ved brug af
  - delmængdekonstruktionen: Hvis NFAen har tilstandsmængde Q, skal DFAens tilstandsmængde være $\mathcal{P}(Q)$
  - og ε-aflukningen: Den nye transitionsfunktion skal være

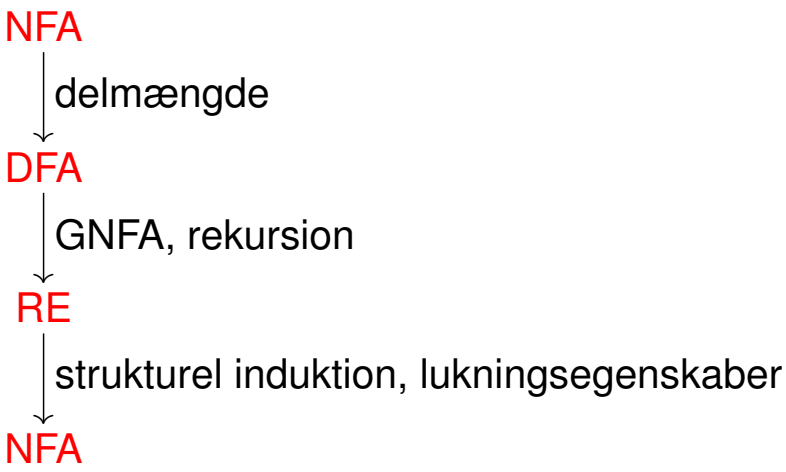
$$\delta'(R, a) = \{q \in Q \mid q \text{ kan nås fra } R \text{ ved en } a\text{-transition efterfulgt af 0 eller flere } \varepsilon\text{-transitioner}\}$$

- Sætning 1.45, 1.47, 1.49: Mængden af regulære sprog er lukket under de regulære operationer. Dvs. $A_1, A_2 \in \Sigma^*$ regulære $\Rightarrow A_1 \cup A_2$, $A_1 \circ A_2$, A_1^* regulære
- Bevis ved at sammensætte NFAs på en meget intuitiv måde
- Sætning 1.25 (fodnote): Mængden af regulære sprog er lukket under $\cap$.
- Bevis ved at konstruere produktet af to DFAs
- Opgave 1.14: Mængden af regulære sprog er lukket under $\bar{}$ (komplement)
- Bevis ved at bytte om på accept- og reject-tilstandene i en DFA

- Definition 1.52: Et regulært udtryk over et alfabet Σ er et udtryk af formen
  1. a for et $a \in \Sigma$, ε eller $\emptyset$,
  2. $(R_1 \cup R_2)$, $(R_1 \circ R_2)$ eller (R_1^*), hvor R_1 og R_2 er regulære udtryk.
- Sproget, som et regulært udtryk R beskriver, betegnes $[\![R]\!]$ og er defineret som følger:
  1. $[\![a]\!] = \{a\}$, $[\![\varepsilon]\!] = \{\varepsilon\}$ og $[\![\emptyset]\!] = \emptyset$,
  2. $[\![R_1 \cup R_2]\!] = [\![R_1]\!] \cup [\![R_2]\!]$, $[\![R_1 \circ R_2]\!] = [\![R_1]\!] \circ [\![R_2]\!]$ og $[\![R_1^*]\!] = [\![R_1]\!]^*$
- Sætning 1.54: Et sprog er regulært hvis og kun hvis det kan beskrives ved et regulært udtryk.
- (følger af Lemma 1.55 og Lemma 1.60)

- Lemma 1.55: Hvis et sprog genereres af et regulært udtryk, da er det regulært.
- Bevis ved brug af strukturel induktion:
  1. Vis at de basale regulære udtryk a, ε og $\emptyset$ kan konverteres til NFAs
  2. Konvertér sammensætninger af regulære udtryk til sammensætninger af NFAs
- Lemma 1.60: Hvis et sprog er regulært, da kan det beskrives ved et regulært udtryk.
- Bevis ved brug af
  - generaliserede NFAs: Konvertér en DFA til en GNFA, der har regulære udtryk på transitionerne (i stedet for bare bogstaver)
  - og rekursion: Konvertér en GNFA til en ny med én tilstand mindre, ved at fjerne en tilstand og lave tilsvarende ændringer på transitionerne.

Tre formalismer der beskriver den samme klasse af sprog:



– Findes der endnu andre formalismer til det? Jep, f.x. regulære grammatikker.

- Sætning 1.70 (Pumpelemmaet): For ethvert regulært sprog A findes der et (naturligt) tal p (pumpelængden) således at ethvert ord $s \in A$ der har længde mindst p kan pumpes, dvs. opsplittes i tre stykker, $s = xyz$, med
  - $|y| \geq 1$ og $|xy| \leq p$,
  - og således at ordene $xy^i z \in A$ for alle $i \in \mathbb{N}_0$.
- Bevis ved at tage en DFA for A og lade p være antallet af dens tilstande
- Anvendelse: At vise at et givet sprog B ikke er regulært:
  1. *antag* at B er regulært
  2. så må der findes en pumpelængde p for B
  3. tag et velegnet ord s som
     - har længde $|s| \geq p$, dvs. bør kunne pumpes,
     - men som *ikke kan pumpes*.
  4. Modstrid!

- `grep`, `sed`, teksteditorer etc.: konverterer et givet regulært udtryk til en NFA for at søge og erstatte
- `lex`, `flex` etc.: konverterer et eller flere givne regulære udtryk til en DFA der kan bruges til leksikalsk analyse

[sok.lex]

Vigtige emner indtil nu

- Sprog ✓
- DFA, NFA ✓
- Konvertering NFA $\longrightarrow$ DFA ✓
- Lukningsegenskaber ($+$ beviser) ✓
- Regulære udtryk ✓
- Konvertering RE $\longrightarrow$ NFA ✓
- Konvertering DFA $\longrightarrow$ RE ?
- Pumpelemma ($+$ bevis) ✓
- Anvendelse af pumpelemma ?

At forstå (på forskellige måder)

	formelt	grafisk	ved eksempel
Sætninger	✓		✓
Beviser	✓	✓	✓
Anvendelser	✓	✓	✓

Hvis I synes at *Sipser* er for blød, eller hvis I vil vide mere end hvad *Sipser* skriver om, prøv at kigge i

> Hopcroft, Motwani, Ullman: Introduction to automata theory, languages, and computation. 2nd ed. Addison-Wesley, 2001

Kontekst-frie grammatikker

11 Kontekst-frie sprog
12 Eksempel
13 Definition
14 Parse-træer
15 Opsummering
16 **Sok**

- Problem: Mange interessante sprog er ikke regulære. F.x.
  - sproget *ADD* fra opgave 1.53
  - sproget L_3 fra syntaksopgaven
  - programmeringssprog generelt
- Brug for "stærkere" værktøjer til at beskrive dem:
  - kontekst-frie grammatikker (CFG) for at *generere* dem
  - push-down-automater (PDA) for at *genkende* dem
- sprog genereret af CFGs $=$ sprog genkendt af PDAs $=$ kontekst-frie sprog
- Er alle sprog kontekst-frie? *Nej.*
- Anvendelse: parsere

En kontekstfri grammatik:

$$S \xrightarrow{1} ASB$$
$$S \xrightarrow{2} \varepsilon$$
$$A \xrightarrow{3} 0$$
$$B \xrightarrow{4} 1$$

- S, A, B: variable
- 0, 1: terminaler
- startvariablen: S

At *generere ord*:

- $S \overset{2}{\Longrightarrow} \varepsilon$ ✓
- $S \overset{1}{\Longrightarrow} ASB \overset{2}{\Longrightarrow} A\varepsilon B \overset{3}{\Longrightarrow} 0B \overset{4}{\Longrightarrow} 01$ ✓
- $S \overset{1}{\Longrightarrow} ASB \overset{1}{\Longrightarrow} AASBB \overset{2}{\Longrightarrow} AA\varepsilon BB \overset{3,3,4,4}{\Longrightarrow} 0011$ ✓
- $S \overset{1,\ldots,1}{\Longrightarrow} A^nSB^n \overset{2}{\Longrightarrow} A^n\varepsilon B^n \overset{3,4}{\Longrightarrow} 0^n1^n$
- grammatikken genererer sproget $\{0^n1^n \mid n \in \mathbb{N}_0\}$

Definition 2.2: En kontekstfri grammatik (CFG) er en 4-tupel $G = (V, \Sigma, R, S)$, hvor delene er

1. V : en endelig mængde af variable
2. Σ : en endelig mængde af terminaler, med $V \cap \Sigma = \emptyset$
3. $R : V \to \mathcal{P}\big((V \cup \Sigma)^*\big)$: produktioner / regler
4. $S \in V$: startvariablen

– produktioner skrives $A \to w$ i stedet for $w \in R(A)$

- Hvis $u, v, w \in (V \cup \Sigma)^*$ er ord og $A \to w$ er en produktion, siges uAv at frembringe uwv: $uAv \Rightarrow uwv$.
- Hvis $u, v \in (V \cup \Sigma)^*$ er ord, siges u at derivere v: $u \overset{*}{\Rightarrow} v$, hvis $u = v$ eller der findes en følge $u_1, u_2, \ldots, u_k$ af ord således at $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$.
- Sproget som G genererer er $[\![G]\!] = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$.

– dvs. et ord $w \in \Sigma^*$ genereres af G hvis og kun hvis der findes en derivation $S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_k \Rightarrow w$, hvor alle $w_i \in (V \cup \Sigma)^*$.

Eksempel 2.3: $G_3 = (\{S\}, \{a, b\}, R, S)$ med produktioner

$$S \to aSb \mid SS \mid \varepsilon$$

Et par derivationer:

- $S \Rightarrow \varepsilon$
- $S \Rightarrow aSb \Rightarrow ab$
- $S \Rightarrow aSb \Rightarrow aSSb \Rightarrow aaSbSb \Rightarrow aaSbaSbb \Rightarrow aababb$

Generelt er det nok at opskrive *produktionerne* for at specificere en kontekstfri grammatik:

- de variable er venstresiderne
- terminalerne er alle andre bogstaver
- startvariablen er venstresiden af den *øverste* produktion

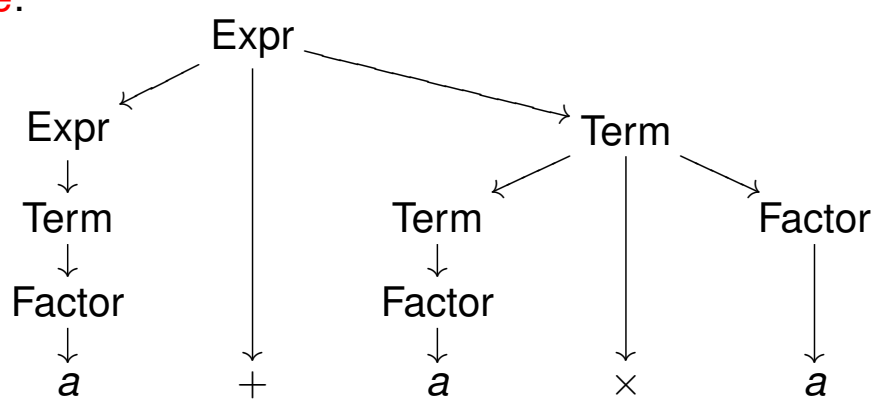
Eksempel 2.4: Aritmetiske udtryk

$$\begin{aligned} \text{Expr} &\to \text{Expr} + \text{Term} \mid \text{Term} \\ \text{Term} &\to \text{Term} \times \text{Factor} \mid \text{Factor} \\ \text{Factor} &\to (\text{Expr}) \mid a \end{aligned}$$

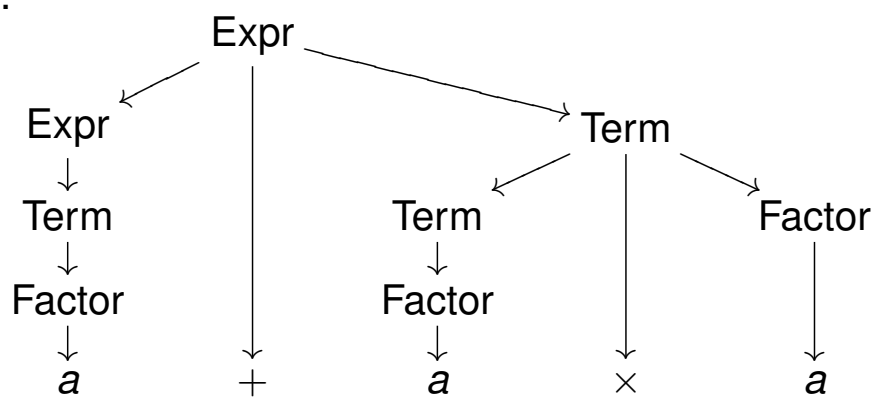
En derivation:

$$\begin{aligned} \text{Expr} \Rightarrow \text{Expr} + \text{Term} \Rightarrow \text{Term} + \text{Term} &\overset{*}{\Rightarrow} \text{Factor} + \text{Term} \times \text{Factor} \\ &\Rightarrow \text{Factor} + \text{Factor} \times \text{Factor} \overset{*}{\Rightarrow} a + a \times a \end{aligned}$$

Et parse-træ:



Et parsetræ:



- Parsetræer udtrykker betydningen af et ord
- At parse: programkode $\rightsquigarrow$ parsetræ $\rightsquigarrow$ ...
- En kontekstfri grammatik i hvilken der er et ord der har to *forskellige* parsetræer kaldes tvetydig.
- to forskellige parsetræer $\Rightarrow$ to forskellige *betydninger* $\Rightarrow$ BAD

Opsummering:

- CFG: et (endeligt) antal produktioner af formen $A \to s_1 \mid s_2 \mid \ldots s_k$ for symboler A og strenge $s_1, s_2, \ldots, s_k$.
- "|" kendetegner *alternativer* (nondeterminisme!)
- symboler på venstre side af produktionerne: variable (eller non-terminaler)
- alle andre symboler: terminaler
- venstre side af *første* produktion: startsymbolet
- at frembringe: $uAv \Rightarrow uwv$ hvis $A \to w$ er en produktion
- hvis w er en streng af *terminaler*: grammatikken genererer w hvis der findes en derivation $S \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_k \Rightarrow w$, hvor alle w_i er strenge af terminaler og variable.
- vigtigt: parsetræer
- Definition: Et sprog siges at være kontekstfrit hvis det kan genereres af en CFG.

Eksempel: En CFG til sproget

$$\{w \in \{a, b\}^* \mid \text{antallet af } a \text{ i } w = \text{antallet af } b \text{ i } w\}$$

Idé: Variable som *tilstande*:

- S : Jeg har set lige mange a og b
- A : Jeg mangler et a
- B : Jeg mangler et b

$$S \to aB \mid bA \mid \varepsilon$$
$$A \to aS \mid bAA$$
$$B \to bS \mid aBB$$

(opgave 2.21!)

Eksempel: En (ufuldstændig og ikke helt rigtig) CFG for **Sok**

$$\begin{aligned}
\text{ProGram} &\to \text{VarErkList}\,;\,\text{MetErkList}\\
\text{VarErkList} &\to \text{VarErk}\,;\,\text{VarErkList} \mid \varepsilon\\
\text{VarErk} &\to \textit{var}\ \text{VarNavn} = \text{HelTal}\\
\text{MetErkList} &\to \text{MetErk}\,;\,\text{MetErkList} \mid \varepsilon\\
\text{MetErk} &\to \textit{metode}\ \text{MetNavn}\ \text{StateMentList}\ \textit{end}\\
\text{StateMentList} &\to \text{StateMent}\,;\,\text{StateMentList} \mid \varepsilon\\
\text{StateMent} &\to \text{MetKald} \mid \text{TilSkriv}\\
\text{TilSkriv} &\to \text{VarNavn} := \text{ArUdtryk}\\
\text{MetKald} &\to \textit{kald}\ \text{MetNavn}
\end{aligned}$$