Syntaks og semantik

Lektion 5

27 februar 2007

Regulære sprog

- 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 $w_1, w_2, \dots, w_k \in \Sigma$ og $r_0, r_1, \dots, r_k \in Q$ således at $w = w_1 w_2 \dots w_k$ og
  1. $r_0 = q_0$,
  2. $r_{i+1} = \delta(r_i, w_{i+1})$ for alle $i = 0, 1, \dots, k-1$, 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 $w_1, w_2, \dots, w_k \in \Sigma \cup \{\varepsilon\}$ og $r_0, r_1, \dots, r_k \in Q$ således at $w = w_1 w_2 \dots w_k$ og
  1. $r_0 = q_0$,
  2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for alle $i = 0, 1, \dots, k-1$, 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 *DFA*s
- Opgave 1.14: Mængden af regulære sprog er lukket under $^-$ (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.

- 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 $xy^iz \in A$ 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]

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

Jeres forståelse som *jeg* oplever den

- sprog ✓
- DFAs ✓
- NFAs ✓
- lukningsegenskaber ✓
- regulære udtryk ✓
  - konvertering DFA → regulært udtryk ?
- ikke-regulære sprog ?

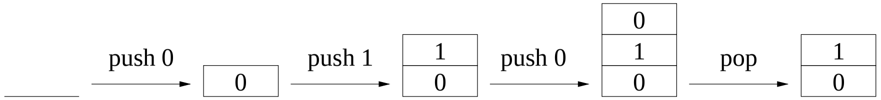
Opgaver som der specielt var problemer med:

- 1.21 a
- 1.29 a, c
- 1.46 a
- 1.53

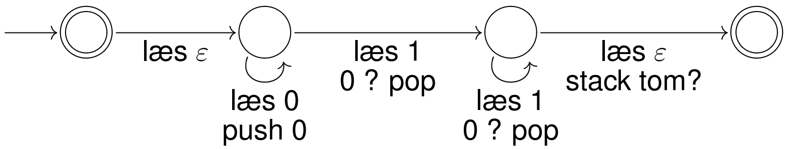
Push-down-automater

- 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

- Pushdown-automat: endelig automat plus *stack*
- Stack:



- kan pushe symboler på stacken og læse og poppe det *øverste* stacksymbol
- Eksempel:



- genkender sproget $\{0^n 1^n \mid n \in \mathbb{N}_0\}$

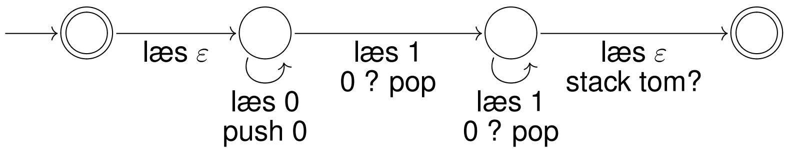
Definition 2.13: En pushdown-automat (PDA) er en 6-tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, hvor delene er

1. Q : en endelig mængde af tilstande
2. Σ : input-alfabetet
3. Γ : stack-alfabetet
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$: transitionsfunktionen
5. $q_0 \in Q$: starttilstanden
6. $F \subseteq Q$: mængden af accepttilstande

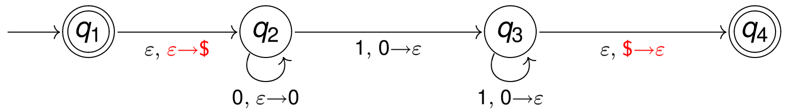
M siges at acceptere et ord $w \in \Sigma^*$ hvis der findes $m \in \mathbb{N}$ og $w_1, w_2, \dots, w_m \in \Sigma_\varepsilon$, $r_0, r_1, \dots, r_m \in Q$ og $s_0, s_1, \dots, s_m \in \Gamma^*$ således at $w = w_1 w_2 \dots w_m$ og

1. $r_0 = q_0$ og $s_0 = \varepsilon$,
2. for alle $i = 0, 1, \dots, m-1$ findes $a, b \in \Gamma_\varepsilon$ og $t \in \Gamma^*$ som opfylder $s_i = at$, $s_{i+1} = bt$ og $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, og
3. $r_m \in F$.

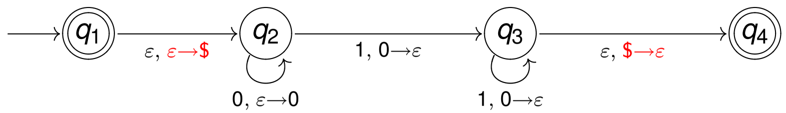
Eksempel 2.14:



At finde ud af om stacken er tom: Introducér et specielt end-of-stack-symbol \$



Eksempel 2.14:



$Q = \{q_1, q_2, q_3, q_4\}$ $\Sigma = \{0, 1\}$ $\Gamma = \{0, \$\}$ $F = \{q_1, q_4\}$

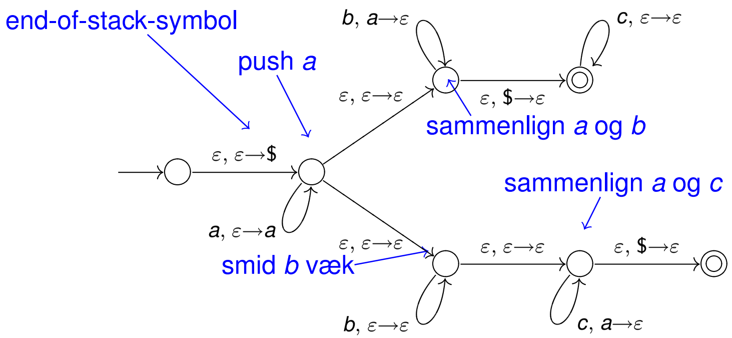
δ : Input:	0			1			ε		
Stack:	0	\$	ε	0	\$	ε	0	\$	ε
q_1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(q_2, \$)\}$
q_2	$\emptyset$	$\emptyset$	$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
q_3	$\emptyset$	$\emptyset$	$\emptyset$	$\{(q_3, \varepsilon)\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(q_4, \varepsilon)\}$	$\emptyset$
q_4	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Opsummering: PDA:

- endelig automat med *stack*
- stacken kan gemme på *vilkårligt mange* symboler, men kun det *øverste* kan læses (og poppes)
- *(first-in, last-out)*
- *nondeterministiske*
- der findes deterministiske PDAs, ja. Men
  - vi skal ikke se på dem her, og
  - de genkender *færre* sprog end de nondeterministiske PDAs!

Eksempel 2.16: En PDA der genkender sproget

$$\{a^i b^j c^k \mid i, j, k \in \mathbb{N}_0 \text{ og } i = j \text{ eller } i = k\}$$



– det kan vises at man *skal* bruge en *nondeterministisk* PDA for at genkende det sprog