3. Seminar EVU RegAut

Sigurd Meldgaard

28. februar 2009

Plan

Lukketheds og afgørlighedsegenskaber

- Lukkethed under $\cup, \cap, ', \cdot, ^*$
- Lukkethed under homomorfi og invers homomorfi
- "Pumping"-lemmaet
- Beslutningsproblemer: membership, emptiness,finiteness subset, equality
- Beslutningsprocedurer i Java-pakken

Lukkethedsegenskaber

Givet to regulære sprog L_1, L_2

- er $L_1 \cap L_2$ regulært

Ja, det beviste vi første seminar (produktkonstruktionen)

Lukkethedsegenskaber

Givet to regulære sprog L_1, L_2

- er $L_1 \cap L_2$ regulært
- er $L_1 \cup L_2$ regulært

Ja, det beviste vi første seminar (produktkonstruktionen)

Lukkethedsegenskaber

Givet to regulære sprog L_1, L_2

- er $L_1 \cap L_2$ regulært
- er $L_1 \cup L_2$ regulært
- er L_1' regulært

Ja, det beviste vi første seminar (produktkonstruktionen)

Lukkethedsegenskaber

Givet to regulære sprog L_1, L_2

- er $L_1 \cap L_2$ regulært
- er $L_1 \cup L_2$ regulært
- er L_1' regulært
- er $L_1 L_2$ regulært

Ja, det beviste vi første seminar (produktkonstruktionen)

Lukkethedsegenskaber

Givet to regulære sprog L_1, L_2

- er $L_1 \cap L_2$ regulært
- er $L_1 \cup L_2$ regulært
- er L_1' regulært
- er $L_1 L_2$ regulært
- er L_1^* regulært

Ja, det beviste vi første seminar (produktkonstruktionen)

Kontraponering, (Contraposition)

- Lukkethedsegenskaber kan vise at sprog *ikke* er regulære.

Kontraponering, (Contraposition)

- Lukkethedsegenskaber kan vise at sprog *ikke* er regulære.
- Fx: Klassen af regulære sprog er lukket under $\cap$

Kontraponering, (Contraposition)

- Lukkethedsegenskaber kan vise at sprog *ikke* er regulære.
- Fx: Klassen af regulære sprog er lukket under $\cap$
- Antag vi har bevist, at sproget S ikke er regulært

Kontraponering, (Contraposition)

- Lukkethedsegenskaber kan vise at sprog *ikke* er regulære.
- Fx: Klassen af regulære sprog er lukket under $\cap$
- Antag vi har bevist, at sproget S ikke er regulært
- Hvis $S = P \cap R$ og R er regulært, så kan P ikke være regulært.

Homomorfier

- Antag $g : \Sigma_1 \rightarrow \Sigma_2^*$ hvor Σ_1 og Σ_2 er alfabeter

Homomorfier

- Antag $g : \Sigma_1 \to \Sigma_2^*$ hvor Σ_1 og Σ_2 er alfabeter
- Definer $h : \Sigma_1^* \to \Sigma_2^*$ ved

$$h(x) = \begin{cases} \Lambda & \text{hvis } x = \Lambda \\ h(y)g(a) & \text{hvis } x = ya, y \in \Sigma_1^*, a \in \Sigma_1 \end{cases}$$

Homomorfier

- Antag $g : \Sigma_1 \rightarrow \Sigma_2^*$ hvor Σ_1 og Σ_2 er alfabeter
- Definer $h : \Sigma_1^* \rightarrow \Sigma_2^*$ ved

$$h(x) = \begin{cases} \Lambda & \text{hvis } x = \Lambda \\ h(y)g(a) & \text{hvis } x = ya, y \in \Sigma_1^*, a \in \Sigma_1 \end{cases}$$

- h opfylder at h(xy)=h(x)h(y) og kaldes en homomorfi

Homomorfier

- Antag $g : \Sigma_1 \to \Sigma_2^*$ hvor Σ_1 og Σ_2 er alfabeter
- Definer $h : \Sigma_1^* \to \Sigma_2^*$ ved

$$h(x) = \begin{cases} \Lambda & \text{hvis } x = \Lambda \\ h(y)g(a) & \text{hvis } x = ya, y \in \Sigma_1^*, a \in \Sigma_1 \end{cases}$$

- h opfylder at h(xy)=h(x)h(y) og kaldes en homomorfi
- Definer $h(L) = \{h(x) | x \in L\}$ for alle $L \subseteq \Sigma_1^*$

Homomorfier

- Antag $g : \Sigma_1 \rightarrow \Sigma_2^*$ hvor Σ_1 og Σ_2 er alfabeter
- Definer $h : \Sigma_1^* \rightarrow \Sigma_2^*$ ved

$$h(x) = \begin{cases} \Lambda & \text{hvis } x = \Lambda \\ h(y)g(a) & \text{hvis } x = ya, y \in \Sigma_1^*, a \in \Sigma_1 \end{cases}$$

- h opfylder at h(xy)=h(x)h(y) og kaldes en homomorfi
- Definer $h(L) = \{h(x) | x \in L\}$ for alle $L \subseteq \Sigma_1^*$
- og $h - 1(L) = \{x | h(x) \in L\}$ for alle $L \subseteq \Sigma_2^*$

Homomorfier

- Antag $g : \Sigma_1 \to \Sigma_2^*$ hvor Σ_1 og Σ_2 er alfabeter
- Definer $h : \Sigma_1^* \to \Sigma_2^*$ ved

$$h(x) = \begin{cases} \Lambda & \text{hvis } x = \Lambda \\ h(y)g(a) & \text{hvis } x = ya, y \in \Sigma_1^*, a \in \Sigma_1 \end{cases}$$

- h opfylder at h(xy)=h(x)h(y) og kaldes en homomorfi
- Definer $h(L) = \{h(x) | x \in L\}$ for alle $L \subseteq \Sigma_1^*$
- og $h - 1(L) = \{x | h(x) \in L\}$ for alle $L \subseteq \Sigma_2^*$
- Se opg. [Martin, opg.4.46] p. 166.

Regularitet og homomorphier

- Hvis $h : \Sigma_1^* \to \Sigma_2^*$ er en homomorphi og $L \subseteq \Sigma_1^*$ er et regulært sprog, så er $h(L)$ også regulært.

Dvs. klassen af regulære sprog er lukket både under homomorfi og invers homomorfi.

Regularitet og homomorphier

- Hvis $h : \Sigma_1^* \to \Sigma_2^*$ er en homomorphi og $L \subseteq \Sigma_1^*$ er et regulært sprog, så er $h(L)$ også regulært.
- Hvis $h : \Sigma_1^* \to \Sigma_2^*$ er en homomorphi og $L \subseteq \Sigma_2^*$ er et regulært sprog, så er $h^{-1}(L)$ også regulært.

Dvs. klassen af regulære sprog er lukket både under homomorfi og invers homomorfi.

Eksempel

- Er følgende sprog over alfabetet $\Sigma = \{0, 1, 2\}$ regulært? $L = \{x2y | y = \mathit{reverse}(x), x, y \in \{0, 1\}^*\}$

Eksempel

- Er følgende sprog over alfabetet $\Sigma = \{0, 1, 2\}$ regulært? $L = \{x2y | y = reverse(x), x, y \in \{0, 1\}^*\}$
- Vi ved (fra første seminar) at sproget $pal = \{x \in \{0, 1\}^* | x = reverse(x)\}$ ikke er regulært

Eksempel

- Er følgende sprog over alfabetet $\Sigma = \{0, 1, 2\}$ regulært? $L = \{x2y | y = reverse(x), x, y \in \{0, 1\}^*\}$
- Vi ved (fra første seminar) at sproget $pal = \{x \in \{0, 1\}^* | x = reverse(x)\}$ ikke er regulært
- En (utilstrækkelig) intuition: L minder om pal, men måske symbolet 2, der markerer midten af strengen, gør, at vi kan lave en FA for L?

Eksempel, fortsat

- Definer tre funktioner $g1, g2, g3 : \{0,1,2\} \rightarrow \{0,1\}^*$ ved

$$g_1(0) = 0,\ g_2(0) = 0,\ g_3(0) = 0$$

$$g_1(1) = 1,\ g_2(1) = 1,\ g_3(1) = 1$$

$$g_1(2) = \Lambda,\ g_2(2) = 0,\ g_3(2) = 1$$

Eksempel, fortsat

- Definer tre funktioner $g1, g2, g3 : \{0,1,2\} \to \{0,1\}^*$ ved

$$g_1(0) = 0,\ g_2(0) = 0,\ g_3(0) = 0$$

$$g_1(1) = 1,\ g_2(1) = 1,\ g_3(1) = 1$$

$$g_1(2) = \Lambda,\ g_2(2) = 0,\ g_3(2) = 1$$

- Og lad $h1, h2, h3$ være de tilhørende homomorfier

Eksempel, fortsat

- Definer tre funktioner $g1, g2, g3 : \{0, 1, 2\} \to \{0, 1\}^*$ ved

$$g_1(0) = 0,\ g_2(0) = 0,\ g_3(0) = 0$$

$$g_1(1) = 1,\ g_2(1) = 1,\ g_3(1) = 1$$

$$g_1(2) = \Lambda,\ g_2(2) = 0,\ g_3(2) = 1$$

- Og lad $h1, h2, h3$ være de tilhørende homomorfier
- $h1(L) \cup h2(L) \cup h3(L) = pal$

Eksempel, fortsat

- Definer tre funktioner $g1, g2, g3 : \{0,1,2\} \rightarrow \{0,1\}^*$ ved

$$g_1(0) = 0,\; g_2(0) = 0,\; g_3(0) = 0$$

$$g_1(1) = 1,\; g_2(1) = 1,\; g_3(1) = 1$$

$$g_1(2) = \Lambda,\; g_2(2) = 0,\; g_3(2) = 1$$

- Og lad $h1, h2, h3$ være de tilhørende homomorfier
- $h1(L) \cup h2(L) \cup h3(L) = pal$
- Så L er *ikke* regulært, idet pal ikke er regulært og klassen af regulære sprog er lukket under forening og homomorfi

Bevis, del 1 (Øvelse)

- Hvis h: $\Sigma_1^* \rightarrow \Sigma_2^*$ er en homomorfi og $L \subseteq \Sigma_1^*$ er et regulært sprog, så er $h(L)$ også regulært

Bevis, del 1 (Øvelse)

- Hvis h: $\Sigma_1^* \rightarrow \Sigma_2^*$ er en homomorfi og $L \subseteq \Sigma_1^*$ er et regulært sprog, så er $h(L)$ også regulært
- Bevis: Strukturel induktion i regulære udtryk... (erstat hver $a \in \Sigma_1$ i udtrykket med $h(a)$).

Bevis, del 2

- Hvis h: $\Sigma_1^* \longrightarrow \Sigma_2^*$ er en homomorfi og $L \subseteq \Sigma_2^*$ er et regulært sprog, så er $h^{-1}(L)$ også regulært

Bevis, del 2

- Hvis h: $\Sigma_1^* \to \Sigma_2^*$ er en homomorfi og $L \subseteq \Sigma_2^*$ er et regulært sprog, så er $h^{-1}(L)$ også regulært
- Bevis: givet en FA $M = (Q, \Sigma_2, q_0, A, \delta)$ så $L(M) = L$. Definer en ny FA $M' = (Q, \Sigma_1, q_0, A, \delta')$ hvor $\delta'(q, a) = \delta^*(q, h(a))$

Bevis, del 2

- Hvis h: $\Sigma_1^* \rightarrow \Sigma_2^*$ er en homomorfi og $L \subseteq \Sigma_2^*$ er et regulært sprog, så er $h^{-1}(L)$ også regulært
- Bevis: givet en FA $M = (Q, \Sigma_2, q_0, A, \delta)$ så $L(M) = L$. Definer en ny FA $M' = (Q, \Sigma_1, q_0, A, \delta')$ hvor $\delta'(q, a) = \delta^*(q, h(a))$
- Bevis nu at $L(M') = h^{-1}(L) = h^{-1}(L(M))$.

Bevis, del 2

- Hvis h: $\Sigma_1^* \to \Sigma_2^*$ er en homomorfi og $L \subseteq \Sigma_2^*$ er et regulært sprog, så er $h^{-1}(L)$ også regulært
- Bevis: givet en FA $M = (Q, \Sigma_2, q_0, A, \delta)$ så $L(M) = L$. Definer en ny FA $M' = (Q, \Sigma_1, q_0, A, \delta')$ hvor $\delta'(q, a) = \delta^*(q, h(a))$
- Bevis nu at $L(M') = h^{-1}(L) = h^{-1}(L(M))$.
- Brug induktion i en streng $x \in L(M)$ og vis at $\delta'^*(q_0, x) = \delta^*(q_0, h(x))$.

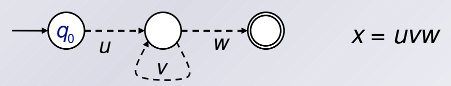
Plan

Endnu en egenskab ved regulære sprog

- Antag $M = (Q, \Sigma, q_0, A, \delta)$ er en FA og $\exists x \in L(M) : |x| \geq |Q|$

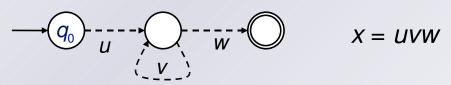
Endnu en egenskab ved regulære sprog

- Antag $M = (Q, \Sigma, q_0, A, \delta)$ er en FA og $\exists x \in L(M) : |x| \geq |Q|$
- Ved en kørsel af x på M vil mindst én af tilstandene blive besøgt mere end en gang.



Endnu en egenskab ved regulære sprog

- Antag $M = (Q, \Sigma, q_0, A, \delta)$ er en FA og $\exists x \in L(M) : |x| \geq |Q|$
- Ved en kørsel af x på M vil mindst én af tilstandene blive besøgt mere end en gang.



- Se på den første af disse tilstande, nu kan vi se at: $\exists u, v, w \in \Sigma^* : x = uvw \wedge |uv| \leq |Q| \wedge |v| > 0 \wedge \delta^*(q_0, u) = \delta^*(q_0, uv)$

"Pumping"-lemmaet for regulære sprog

Hvis L er et regulært sprog så gælder:

$\exists n > 0 :$

"Pumping"-lemmaet for regulære sprog

Hvis L er et regulært sprog så gælder:

$\exists n > 0 :$

$\forall x \in L$ hvor $|x| \geq n :$

"Pumping"-lemmaet for regulære sprog

Hvis L er et regulært sprog så gælder:

$$\exists n > 0 :$$
$$\forall x \in L \text{ hvor } |x| \geq n :$$
$$\exists u, v, w \in \Sigma^* :$$

"Pumping"-lemmaet for regulære sprog

Hvis L er et regulært sprog så gælder:

$$\exists n > 0 :$$
$$\forall x \in L \text{ hvor } |x| \geq n :$$
$$\exists u, v, w \in \Sigma^* :$$
$$(x = uvw)$$

"Pumping"-lemmaet for regulære sprog

Hvis L er et regulært sprog så gælder:

$$\exists n > 0 :$$
$$\forall x \in L \text{ hvor } |x| \geq n :$$
$$\exists u, v, w \in \Sigma^* :$$
$$(x = uvw) \wedge (|uv| \leq n)$$

"Pumping"-lemmaet for regulære sprog

Hvis L er et regulært sprog så gælder:

$$
\begin{aligned}
&\exists n > 0 : \\
&\quad \forall x \in L \text{ hvor } |x| \geq n : \\
&\quad\quad \exists u, v, w \in \Sigma^* : \\
&\quad\quad\quad (x = uvw) \land (|uv| \leq n) \land (|v| > 0) \land
\end{aligned}
$$

"Pumping"-lemmaet for regulære sprog

Hvis L er et regulært sprog så gælder:

$$\begin{aligned}&\exists n > 0 :\\ &\quad \forall x \in L \text{ hvor } |x| \geq n :\\ &\quad\quad \exists u, v, w \in \Sigma^* :\\ &\quad\quad\quad (x = uvw) \wedge (|uv| \leq n) \wedge (|v| > 0) \wedge\\ &\quad\quad\quad (\forall m \geq 0 : uv^m w \in L)\end{aligned}$$

"Pumping"-lemmaet for regulære sprog

Hvis L er et regulært sprog så gælder:

$$
\begin{aligned}
&\exists n > 0 : \\
&\quad \forall x \in L \text{ hvor } |x| \geq n : \\
&\quad\quad \exists u, v, w \in \Sigma^* : \\
&\quad\quad\quad (x = uvw) \wedge (|uv| \leq n) \wedge (|v| > 0) \wedge \\
&\quad\quad\quad (\forall m \geq 0 : uv^m w \in L)
\end{aligned}
$$

"Pumping"-lemmaet for ikke-regulære sprog

Dette resultat kan kontraponeres:

"Pumping"-lemmaet for ikke-regulære sprog

Dette resultat kan kontraponeres:

Hvis det gælder om L

"Pumping"-lemmaet for ikke-regulære sprog

Dette resultat kan kontraponeres:

Hvis det gælder om L

$\forall n > 0$:

"Pumping"-lemmaet for ikke-regulære sprog

Dette resultat kan kontraponeres:

Hvis det gælder om L

$\forall n > 0$:

$\exists x \in L$ hvor $|x| \geq n$:

"Pumping"-lemmaet for ikke-regulære sprog

Dette resultat kan kontraponeres:

Hvis det gælder om L

$\forall n > 0$:

$\exists x \in L$ hvor $|x| \geq n$:

$\forall u, v, w \in \Sigma^*$:

"Pumping"-lemmaet for ikke-regulære sprog

Dette resultat kan kontraponeres:

Hvis det gælder om L

$$
\begin{aligned}
&\forall n > 0 : \\
&\quad \exists x \in L \text{ hvor } |x| \geq n : \\
&\quad\quad \forall u, v, w \in \Sigma^* : \\
&\quad\quad\quad (x = uvw) \land (|uv| \leq n) \land (|v| > 0) \land
\end{aligned}
$$

“Pumping”-lemmaet for ikke-regulære sprog

Dette resultat kan kontraponeres:

Hvis det gælder om L

$$\forall n > 0 :$$
$$\exists x \in L \text{ hvor } |x| \geq n :$$
$$\forall u, v, w \in \Sigma^* :$$
$$(x = uvw) \wedge (|uv| \leq n) \wedge (|v| > 0) \wedge$$
$$(\exists m \geq 0 : uv^m w \notin L)$$

"Pumping"-lemmaet for ikke-regulære sprog

Dette resultat kan kontraponeres:

Hvis det gælder om L

$$\begin{aligned}
&\forall n > 0 : \\
&\quad \exists x \in L \text{ hvor } |x| \geq n : \\
&\quad\quad \forall u, v, w \in \Sigma^* : \\
&\quad\quad\quad (x = uvw) \wedge (|uv| \leq n) \wedge (|v| > 0) \wedge \\
&\quad\quad\quad (\exists m \geq 0 : uv^m w \notin L)
\end{aligned}$$

Så kan L ikke være regulært.

Pumping-lemmaet som "kvantor-spil"

- Antag vi prøver at vise, at L er ikke-regulært

Pumping-lemmaet som "kvantor-spil"

- Antag vi prøver at vise, at L er ikke-regulært
- Vi skal vise noget på form
  $\forall n \ldots : \exists x \ldots : \forall u, v, w \ldots : \exists m \ldots : \ldots$

Pumping-lemmaet som “kvantor-spil”

- Antag vi prøver at vise, at L er ikke-regulært
- Vi skal vise noget på form
  $\forall n \ldots : \exists x \ldots : \forall u, v, w \ldots : \exists m \ldots : \ldots$
- “Fjenden” vil prøve at modarbejde os

Pumping-lemmaet som “kvantor-spil”

- Antag vi prøver at vise, at L er ikke-regulært
- Vi skal vise noget på form
  $\forall n \ldots : \exists x \ldots : \forall u, v, w \ldots : \exists m \ldots : \ldots$
- “Fjenden” vil prøve at modarbejde os
  1. Fjenden vælger n

Pumping-lemmaet som “kvantor-spil”

- Antag vi prøver at vise, at L er ikke-regulært
- Vi skal vise noget på form
  $\forall n \ldots : \exists x \ldots : \forall u, v, w \ldots : \exists m \ldots : \ldots$
- “Fjenden” vil prøve at modarbejde os
  1. Fjenden vælger n
  2. Vi vælger x (efter reglerne, dvs. så $x \in L$ og $|x| \geq n$)

Pumping-lemmaet som "kvantor-spil"

- Antag vi prøver at vise, at L er ikke-regulært
- Vi skal vise noget på form
  $\forall n \ldots : \exists x \ldots : \forall u, v, w \ldots : \exists m \ldots : \ldots$
- "Fjenden" vil prøve at modarbejde os
  1. Fjenden vælger n
  2. Vi vælger x (efter reglerne, dvs. så $x \in L$ og $|x| \geq n$)
  3. Fjenden vælger u, v, w (efter reglerne...)

Pumping-lemmaet som "kvantor-spil"

- Antag vi prøver at vise, at L er ikke-regulært
- Vi skal vise noget på form
  $\forall n \ldots : \exists x \ldots : \forall u, v, w \ldots : \exists m \ldots : \ldots$
- "Fjenden" vil prøve at modarbejde os
  1. Fjenden vælger n
  2. Vi vælger x (efter reglerne, dvs. så $x \in L$ og $|x| \geq n$)
  3. Fjenden vælger u, v, w (efter reglerne...)
  4. Vi vælger m

Pumping-lemmaet som "kvantor-spil"

- Antag vi prøver at vise, at L er ikke-regulært
- Vi skal vise noget på form
  $\forall n \ldots : \exists x \ldots : \forall u, v, w \ldots : \exists m \ldots : \ldots$
- "Fjenden" vil prøve at modarbejde os
  1. Fjenden vælger n
  2. Vi vælger x (efter reglerne, dvs. så $x \in L$ og $|x| \geq n$)
  3. Fjenden vælger u, v, w (efter reglerne...)
  4. Vi vælger m
- Hvis vi uanset fjendens valg kan opnå at $uv^m w \notin L$, så har vi vundet, dvs. bevist at L er ikke-regulært.

Eksempel 1

Lad $L = \{0^i 1^i | i \geq 0\}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$

Eksempel 1

Lad $L = \{0^i 1^i | i \geq 0\}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n 1^n$ som opfylder $x \in L$ og $|x| \geq n$

Eksempel 1

Lad $L = \{0^i 1^i | i \geq 0\}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n 1^n$ som opfylder $x \in L$ og $|x| \geq n$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$

Eksempel 1

Lad $L = \{0^i 1^i | i \geq 0\}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n 1^n$ som opfylder $x \in L$ og $|x| \geq n$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$
- Vi vælger $m = 2$

Eksempel 1

Lad $L = \{0^i 1^i | i \geq 0\}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n 1^n$ som opfylder $x \in L$ og $|x| \geq n$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$
- Vi vælger $m = 2$
- Da $x = uvw = 0^n 1^n, |uv| \leq n$ og $|v| > 0$ så gælder at $v = 0^k$ for et $k > 0$

Eksempel 1

Lad $L = \{0^i 1^i | i \geq 0\}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n 1^n$ som opfylder $x \in L$ og $|x| \geq n$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$
- Vi vælger $m = 2$
- Da $x = uvw = 0^n 1^n, |uv| \leq n$ og $|v| > 0$ så gælder at $v = 0^k$ for et $k > 0$
- D.v.s. $uv^m w = uv^2 w = 0^{n+k} 1^n \notin L$

Eksempel 1

Lad $L = \{0^i 1^i | i \geq 0\}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n 1^n$ som opfylder $x \in L$ og $|x| \geq n$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$
- Vi vælger $m = 2$
- Da $x = uvw = 0^n 1^n, |uv| \leq n$ og $|v| > 0$ så gælder at $v = 0^k$ for et $k > 0$
- D.v.s. $uv^m w = uv^2 w = 0^{n+k} 1^n \notin L$
- Så L er ikke regulært.

Eksempel 2

Lad $L = pal = \{x \in \Sigma^* | x = reverse(x)\}$
V.h.a. pumping-lemmaet vil vi vise at *pal* ikke er regulært.

- Fjenden vælger et $n > 0$

Eksempel 2

Lad $L = pal = \{x \in \Sigma^* | x = reverse(x)\}$
V.h.a. pumping-lemmaet vil vi vise at *pal* ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n 1 0^n$ som opfylder $x \in pal$ og $|x| \geq n$

Eksempel 2

Lad $L = pal = \{x \in \Sigma^* | x = reverse(x)\}$
V.h.a. pumping-lemmaet vil vi vise at *pal* ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n 1 0^n$ som opfylder $x \in pal$ og $|x| \geq n$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$

Eksempel 2

Lad $L = pal = \{x \in \Sigma^* | x = reverse(x)\}$
V.h.a. pumping-lemmaet vil vi vise at *pal* ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n 1 0^n$ som opfylder $x \in pal$ og $|x| \geq n$
- Fjenden vælger u, v, w så $x = uvw$, $|uv| \leq n$ og $|v| > 0$
- Vi vælger $m = 2$

Eksempel 2

Lad $L = pal = \{x \in \Sigma^* | x = reverse(x)\}$
V.h.a. pumping-lemmaet vil vi vise at *pal* ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n10^n$ som opfylder $x \in pal$ og $|x| \geq n$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$
- Vi vælger $m = 2$
- Da $x = uvw = 0^n10^n, |uv| \leq n$ og $|v| > 0$ så gælder at $v = 0^k$ for et $k > 0$

Eksempel 2

Lad $L = pal = \{x \in \Sigma^* | x = reverse(x)\}$
V.h.a. pumping-lemmaet vil vi vise at *pal* ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n10^n$ som opfylder $x \in pal$ og $|x| \geq n$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$
- Vi vælger $m = 2$
- Da $x = uvw = 0^n10^n, |uv| \leq n$ og $|v| > 0$ så gælder at $v = 0^k$ for et $k > 0$
- D.v.s. $uv^mw = uv^2w = 0^{n+k}10^n \notin pal$

Eksempel 2

Lad $L = pal = \{x \in \Sigma^* | x = reverse(x)\}$
V.h.a. pumping-lemmaet vil vi vise at *pal* ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi vælger $x = 0^n 1 0^n$ som opfylder $x \in pal$ og $|x| \geq n$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$
- Vi vælger $m = 2$
- Da $x = uvw = 0^n 1 0^n, |uv| \leq n$ og $|v| > 0$ så gælder at $v = 0^k$ for et $k > 0$
- D.v.s. $uv^m w = uv^2 w = 0^{n+k} 1 0^n \notin pal$
- Så *pal* er ikke regulært.

Eksempel 3

Lad $L = \{0^p | p \text{ er et primtal }\}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$

Eksempel 3

Lad $L = \{0^p | p \text{ er et primtal } \}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi finder et primtal $> n + 1$ og vælger $x = 0^p$

Eksempel 3

Lad $L = \{0^p | p \text{ er et primtal } \}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi finder et primtal $> n + 1$ og vælger $x = 0^p$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$

Eksempel 3

Lad $L = \{0^p | p \text{ er et primtal } \}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi finder et primtal $> n+1$ og vælger $x = 0^p$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$
- Vi vælger $m = p - |v|$

Eksempel 3

Lad $L = \{0^p | p \text{ er et primtal }\}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi finder et primtal $> n + 1$ og vælger $x = 0^p$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$
- Vi vælger $m = p - |v|$
- $|uv^m w| = |uv^{p-|v|} w| = |uw| + (p - |v|) \cdot |v| = (p - |v|) + (p - |v|) \cdot |v| = (p - |v|) \cdot (p - |v|)$ Begge disse led > 1, dvs. $|uv^m w|$ ikke er et primtal

Eksempel 3

Lad $L = \{0^p | p \text{ er et primtal } \}$
V.h.a. pumping-lemmaet vil vi vise at L ikke er regulært.

- Fjenden vælger et $n > 0$
- Vi finder et primtal $> n+1$ og vælger $x = 0^p$
- Fjenden vælger u, v, w så $x = uvw, |uv| \leq n$ og $|v| > 0$
- Vi vælger $m = p - |v|$
- $|uv^m w| = |uv^{p-|v|}w| = |uw| + (p - |v|) \cdot |v| = (p - |v|) + (p - |v|) \cdot |v| = (p - |v|) \cdot (p - |v|)$ Begge disse led > 1, dvs. $|uv^m w|$ ikke er et primtal
- Så L er ikke regulært.

Advarsel

- Pumping-lemmaet kan ikke bruges til at vise, at et givet regulært sprog er regulært

Advarsel

- Pumping-lemmaet kan ikke bruges til at vise, at et givet regulært sprog er regulært
- Eksempel: $L = \{a^i b^j c^j | i \geq 1 \text{ og } j \geq 0\} \cup \{b^j c^k | j, k \geq 0\}$

Advarsel

- Pumping-lemmaet kan ikke bruges til at vise, at et givet regulært sprog er regulært
- Eksempel: $L = \{a^i b^j c^j | i \geq 1 \text{ og } j \geq 0\} \cup \{b^j c^k | j, k \geq 0\}$
- L er ikke regulært, men L har pumping-egenskaben (bevis p. 185). (dvs. $\exists n \ldots : \forall x \ldots : \exists u, v, w \ldots : \forall m \geq 0 : uv^m w \in L$)

Øvelser

Martin 5.23 (a+b+e) p. 195.

Plan

Beslutningsproblemer

- Membership: Givet en FA M og en streng x, tilhører x sproget af M?

Beslutningsproblemer

- Membership: Givet en FA M og en streng x, tilhører x sproget af M?
- Emptiness: Givet en FA M, er sproget for M tomt?

Beslutningsproblemer

- Membership: Givet en FA M og en streng x, tilhører x sproget af M?
- Emptiness: Givet en FA M, er sproget for M tomt?
- Finiteness: Givet en FA M, er sproget for M endeligt?

Beslutningsproblemer

- Membership: Givet en FA M og en streng x, tilhører x sproget af M?
- Emptiness: Givet en FA M, er sproget for M tomt?
- Finiteness: Givet en FA M, er sproget for M endeligt?
- Subset: Givet to FA'er, M_1 og M_2, er sproget for M_1 en delmængde af sproget for M_2?

Beslutningsproblemer

- Membership: Givet en FA M og en streng x, tilhører x sproget af M?
- Emptiness: Givet en FA M, er sproget for M tomt?
- Finiteness: Givet en FA M, er sproget for M endeligt?
- Subset: Givet to FA'er, M_1 og M_2, er sproget for M_1 en delmængde af sproget for M_2?
- Equality: Givet to FA'er, M_1 og M_2, er sprogene for M_1 og M_2 ens?

Beslutningsproblemer

- Membership: Givet en FA M og en streng x, tilhører x sproget af M?
- Emptiness: Givet en FA M, er sproget for M tomt?
- Finiteness: Givet en FA M, er sproget for M endeligt?
- Subset: Givet to FA'er, M_1 og M_2, er sproget for M_1 en delmængde af sproget for M_2?
- Equality: Givet to FA'er, M_1 og M_2, er sprogene for M_1 og M_2 ens?
- Alle disse problemer er afgørlige!

Membership-problemet

Givet en FA M og en streng x, tilhører x sproget af M? (Dvs. er $x \in L(M)$?)

Membership-problemet

Givet en FA M og en streng x, tilhører x sproget af M? (Dvs. er $x \in L(M)$?)
Algoritme: Kør x på M, startende i starttilstanden, og se om den ender i en accepttilstand

Emptyness-problemet

- Givet en FA M, er sproget for M tomt? (Dvs. er $L(M) = \emptyset$?)

Emptyness-problemet

- Givet en FA M, er sproget for M tomt? (Dvs. er $L(M) = \emptyset$?)
- Algoritme 1: Afprøv for alle $x \in \Sigma^*$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet

Emptyness-problemet

- Givet en FA M, er sproget for M tomt? (Dvs. er $L(M) = \emptyset$?)
- Algoritme 1: Afprøv for alle $x \in \Sigma^*$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet
- Duer ikke (uendelig mange strenge at prøve med)

Emptyness-problemet

- Givet en FA M, er sproget for M tomt? (Dvs. er $L(M) = \emptyset$?)
- Algoritme 1: Afprøv for alle $x \in \Sigma^*$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet
- Duer ikke (uendelig mange strenge at prøve med)
- Algoritme 1': Afprøv for alle x hvor $|x| < |Q|$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet (dette er en reduktion af det emptyness-problemet til membership-problemet)

Emptyness-problemet

- Givet en FA M, er sproget for M tomt? (Dvs. er $L(M) = \emptyset$?)
- Algoritme 1: Afprøv for alle $x \in \Sigma^*$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet
- Duer ikke (uendelig mange strenge at prøve med)
- Algoritme 1': Afprøv for alle x hvor $|x| < |Q|$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet (dette er en reduktion af det emptyness-problemet til membership-problemet)
- Algoritme 2: Undersøg om der findes en accepttilstand

Emptyness-problemet

- Givet en FA M, er sproget for M tomt? (Dvs. er $L(M) = \emptyset$?)
- Algoritme 1: Afprøv for alle $x \in \Sigma^*$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet
- Duer ikke (uendelig mange strenge at prøve med)
- Algoritme 1': Afprøv for alle x hvor $|x| < |Q|$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet (dette er en reduktion af det emptyness-problemet til membership-problemet)
- Algoritme 2: Undersøg om der findes en accepttilstand
- Duer ikke hvis accepttilstanden ikke kan opnås fra starttilstanden

Emptyness-problemet

- Givet en FA M, er sproget for M tomt? (Dvs. er $L(M) = \emptyset$?)
- Algoritme 1: Afprøv for alle $x \in \Sigma^*$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet
- Duer ikke (uendelig mange strenge at prøve med)
- Algoritme 1': Afprøv for alle x hvor $|x| < |Q|$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet (dette er en reduktion af det emptyness-problemet til membership-problemet)
- Algoritme 2: Undersøg om der findes en accepttilstand
- Duer ikke hvis accepttilstanden ikke kan opnås fra starttilstanden
- Algoritme 2': Undersøg om der findes en accepttilstand, som er opnåelig fra starttilstanden

Finiteness-problemet

- Givet en FA M, er sproget for M endeligt? (Dvs. er L(M) en endelig mængde?)

Finiteness-problemet

- Givet en FA M, er sproget for M endeligt? (Dvs. er L(M) en endelig mængde?)
- Algoritme 1: Afprøv for alle x hvor $|Q| \leq |x| < 2 \cdot |Q|$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet. $L(M)$ er endeligt hvis og kun hvis der ikke eksisterer en sådan streng (Bevis for korrekthed: se bogen p. 188.)

Finiteness-problemet

- Givet en FA M, er sproget for M endeligt? (Dvs. er L(M) en endelig mængde?)
- Algoritme 1: Afprøv for alle x hvor $|Q| \leq |x| < 2 \cdot |Q|$ om $x \in L(M)$ ved hjælp af algoritmen fra membership-problemet. $L(M)$ er endeligt hvis og kun hvis der ikke eksisterer en sådan streng (Bevis for korrekthed: se bogen p. 188.)
- Algoritme 2: Ide: Udnyt at $L(M)$ er uendeligt hvis og kun hvis der i tilstandsgrafen for M eksisterer en cykel, der kan nås fra starttilstanden, og som kan nå til en accepttilstand.

Subset-problemet

- Givet to FA'er, M_1 og M_2, er sproget for M_1 en delmængde af sproget for M_2? (Dvs. er $L(M_1) \subseteq L(M_2)$?)

Subset-problemet

- Givet to FA'er, M_1 og M_2, er sproget for M_1 en delmængde af sproget for M_2? (Dvs. er $L(M_1) \subseteq L(M_2)$?)
- Algoritme: Lav med produktkonstruktionen en FA M_3 som opfylder $L(M_3) = L(M_1) - L(M_2)$ og afgør med en algoritme til emptiness-problemet om $L(M_3) = \emptyset$

Subset-problemet

- Givet to FA'er, M_1 og M_2, er sproget for M_1 en delmængde af sproget for M_2? (Dvs. er $L(M_1) \subseteq L(M_2)$?)
- Algoritme: Lav med produktkonstruktionen en FA M_3 som opfylder $L(M_3) = L(M_1) - L(M_2)$ og afgør med en algoritme til emptiness-problemet om $L(M_3) = \emptyset$
- (Bevis for korrekthed: $L(M_1) \subseteq L(M_2) \Leftrightarrow L(M1) - L(M2) = \emptyset$).

Equality-problemet

- Givet to FA'er, M_1 og M_2, er sproget for M_1 det samme som M_2? (Dvs. er $L(M_1) = L(M_2)$?)

Equality-problemet

- Givet to FA'er, M_1 og M_2, er sproget for M_1 det samme som M_2? (Dvs. er $L(M_1) = L(M_2)$?)
- Algoritme: Test med subset-problemet om $L(M_1) \subseteq L(M_2)$ og $L(M_2) \subseteq L(M_1)$

Øvelser

Martin 5.26 (a+e)

Øvelser

Martin 5.26 (a+e)

Martin 5.28 (a+b+d+g)

dRegAut java pakken

beslutningsprocedurer for de nævnte beslutningsproblemer

- FA.accepts(String)
- FA.isEmpty()
- FA.isFinite()
- FA.subsetOf(FA)
- FA.equals(FA)

dRegAut java pakken

beslutningsprocedurer for de nævnte beslutningsproblemer

- FA.accepts(String)
- FA.isEmpty()
- FA.isFinite()
- FA.subsetOf(FA)
- FA.equals(FA)
- FA.getAShortestExample() — finder en korteste sti fra starttilstanden til en accepttilstand (hvis sproget er ikke-tomt)

getAShortestExample

getAShortestExample() // laver bredde-først gennemløb.

getAShortestExample

getAShortestExample() // laver bredde-først gennemløb.
  $pending = [q_0]$ // queue of states that need to be visited
  $paths = []$ // $paths[i]$ is a shortest path from q_0 to $pending[i]$
  $visited = \{q0\}$ // set of states that have been visited

getAShortestExample

getAShortestExample() // laver bredde-først gennemløb.
  $pending = [q_0]$ // queue of states that need to be visited
  $paths = []$ // $paths[i]$ is a shortest path from q_0 to $pending[i]$
  $visited = \{q0\}$ // set of states that have been visited
  while $pending \neq \emptyset$ do
    $q = pending.removeFirst()$
    $path = paths.removeFirst()$
    if $q \in A$ then return path
    else

getAShortestExample

getAShortestExample() // laver bredde-først gennemløb.
  $pending = [q_0]$ // queue of states that need to be visited
  $paths = []$ // $paths[i]$ is a shortest path from q_0 to $pending[i]$
  $visited = \{q0\}$ // set of states that have been visited
  while $pending \neq \emptyset$ do
    $q = pending.removeFirst()$
    $path = paths.removeFirst()$
    if $q \in A$ then return path
    else
      for each $c \in \Sigma$ do
        $p = \delta(q, c)$

getAShortestExample

getAShortestExample() // laver bredde-først gennemløb.
  $pending = [q_0]$ // queue of states that need to be visited
  $paths = []$ // $paths[i]$ is a shortest path from q_0 to $pending[i]$
  $visited = \{q0\}$ // set of states that have been visited
  while $pending \neq \emptyset$ do
    $q = pending.removeFirst()$
    $path = paths.removeFirst()$
    if $q \in A$ then return path
    else
      for each $c \in \Sigma$ do
        $p = \delta(q, c)$
        if $p \notin visited$
          pending.addToEnd(p)
          paths.addToEnd($path \cdot c$)
          $visited = visited \cup \{p\}$

getAShortestExample

getAShortestExample() // laver bredde-først gennemløb.
  $pending = [q_0]$ // queue of states that need to be visited
  $paths = []$ // $paths[i]$ is a shortest path from q_0 to $pending[i]$
  $visited = \{q0\}$ // set of states that have been visited
  while $pending \neq \emptyset$ do
    $q = pending.removeFirst()$
    $path = paths.removeFirst()$
    if $q \in A$ then return path
    else
      for each $c \in \Sigma$ do
        $p = \delta(q, c)$
        if $p \notin visited$
          pending.addToEnd(p)
          paths.addToEnd($path \cdot c$)
          $visited = visited \cup \{p\}$
  return null // return null if no accept state is found

Plan

Eksempel

- *sentence* $\longrightarrow$ *subject verb object*
- *subject* $\longrightarrow$ *person*
- *person* $\longrightarrow$ **Morten** | **Ole** | **Henrik**
- *verb* $\longrightarrow$ **spurgte** | **sparkede**
- *object* $\longrightarrow$ *thing* | *person*
- *thing* $\longrightarrow$ **fodbolden** | **computeren**

Eksempel

- *sentence* $\longrightarrow$ *subject verb object*
- *subject* $\longrightarrow$ *person*
- *person* $\longrightarrow$ **Morten** | **Ole** | **Henrik**
- *verb* $\longrightarrow$ **spurgte** | **sparkede**
- *object* $\longrightarrow$ *thing* | *person*
- *thing* $\longrightarrow$ **fodbolden** | **computeren**

- Nonterminal-symboler: *sentence*, *subject*, *person*, *verb*, *object*, *thing*

Eksempel

- *sentence* $\rightarrow$ *subject verb object*
- *subject* $\rightarrow$ *person*
- *person* $\rightarrow$ **Morten** | **Ole** | **Henrik**
- *verb* $\rightarrow$ **spurgte** | **sparkede**
- *object* $\rightarrow$ *thing* | *person*
- *thing* $\rightarrow$ **fodbolden** | **computeren**

- Nonterminal-symboler: *sentence*, *subject*, *person*, *verb*, *object*, *thing*
- Terminal-symboler: **Morten**, **Ole**, **Henrik**, **spurgte**, **sparkede**, **fodbolden**, **computeren**

Eksempel

- *sentence* → *subject verb object*
- *subject* → *person*
- *person* → **Morten** | **Ole** | **Henrik**
- *verb* → **spurgte** | **sparkede**
- *object* → *thing* | *person*
- *thing* → **fodbolden** | **computeren**

- Nonterminal-symboler: *sentence*, *subject*, *person*, *verb*, *object*, *thing*
- Terminal-symboler: **Morten**, **Ole**, **Henrik**, **spurgte**, **sparkede**, **fodbolden**, **computeren**
- Start-symbol: *sentence*

Eksempel

- *sentence* $\rightarrow$ *subject verb object*
- *subject* $\rightarrow$ *person*
- *person* $\rightarrow$ **Morten** | **Ole** | **Henrik**
- *verb* $\rightarrow$ **spurgte** | **sparkede**
- *object* $\rightarrow$ *thing* | *person*
- *thing* $\rightarrow$ **fodbolden** | **computeren**

- Nonterminal-symboler: *sentence*, *subject*, *person*, *verb*, *object*, *thing*
- Terminal-symboler: **Morten**, **Ole**, **Henrik**, **spurgte**, **sparkede**, **fodbolden**, **computeren**
- Start-symbol: *sentence*
- Eksempel på derivation: *sentence* $\Rightarrow$ *subject verb object* $\Rightarrow \ldots \Rightarrow$ **Ole spurgte computeren**

Formel definition af CFG

En kontekstfri grammatik (CFG) er et 4-tuppel $G = (V, \Sigma, S, P)$

Formel definition af CFG

En kontekstfri grammatik (CFG) er et 4-tuppel $G = (V, \Sigma, S, P)$

- V er en endelig mængde af non-terminal-symboler

Formel definition af CFG

En kontekstfri grammatik (CFG) er et 4-tuppel $G = (V, \Sigma, S, P)$

- V er en endelig mængde af non-terminal-symboler
- Σ er en endelig mængde af terminal-symboler ($V \cap \Sigma = \emptyset$)

Formel definition af CFG

En kontekstfri grammatik (CFG) er et 4-tuppel $G = (V, \Sigma, S, P)$

- V er en endelig mængde af non-terminal-symboler
- Σ er en endelig mængde af terminal-symboler ($V \cap \Sigma = \emptyset$)
- $S \in V$ er startsymbol.

Formel definition af CFG

En kontekstfri grammatik (CFG) er et 4-tuppel $G = (V, \Sigma, S, P)$

- V er en endelig mængde af non-terminal-symboler
- Σ er en endelig mængde af terminal-symboler ($V \cap \Sigma = \emptyset$)
- $S \in V$ er startsymbol.
- P er en endelig mængde af produktioner på form $A \to \alpha$ hvor $A \in V$ og $\alpha \in (V \cup \Sigma)^*$

Derivationer

- "$\Rightarrow$" repræsenterer ét derivations-trin, hvor en nonterminal erstattes ifølge en produktion

Derivationer

- "$\Rightarrow$" repræsenterer ét derivations-trin, hvor en nonterminal erstattes ifølge en produktion
- dvs. "$\Rightarrow$" er en relation over mængden $(V \cup \Sigma)^*$

Derivationer

- "$\Rightarrow$" repræsenterer ét derivations-trin, hvor en nonterminal erstattes ifølge en produktion
- dvs. "$\Rightarrow$" er en relation over mængden $(V \cup \Sigma)^*$
- Hvis $\alpha_1, \alpha_2 \in (V \cup \Sigma)^*$ og $(A \rightarrow \gamma) \in P$ (dvs. grammatikken indeholder produktionen $A \rightarrow \gamma$)

Derivationer

- "$\Rightarrow$" repræsenterer ét derivations-trin, hvor en nonterminal erstattes ifølge en produktion
- dvs. "$\Rightarrow$" er en relation over mængden $(V \cup \Sigma)^*$
- Hvis $\alpha_1, \alpha_2 \in (V \cup \Sigma)^*$ og $(A \rightarrow \gamma) \in P$ (dvs. grammatikken indeholder produktionen $A \rightarrow \gamma$)
- så gælder

$$\alpha_1 A \alpha_2 \Rightarrow \alpha_1 \gamma \alpha_2$$

Derivationer

- "$\Rightarrow$" repræsenterer ét derivations-trin, hvor en nonterminal erstattes ifølge en produktion
- dvs. "$\Rightarrow$" er en relation over mængden $(V \cup \Sigma)^*$
- Hvis $\alpha_1, \alpha_2 \in (V \cup \Sigma)^*$ og $(A \rightarrow \gamma) \in P$ (dvs. grammatikken indeholder produktionen $A \rightarrow \gamma$)
- så gælder

$$\alpha_1 A \alpha_2 \Rightarrow \alpha_1 \gamma \alpha_2$$

- ("$\Rightarrow$" er i denne sammenhæng ikke et "logisk medfører" tegn)

Sproget for en CFG

- Definer relationen "$\Rightarrow^*$" som den refleksive transitive lukning af "$\Rightarrow$", dvs. $\alpha \Rightarrow^* \beta$ hvis og kun hvis $\alpha \underbrace{\Rightarrow \ldots \Rightarrow \ldots \Rightarrow}_{\text{0 eller flere derivationer}} \beta$

Sproget for en CFG

- Definer relationen "$\Rightarrow^*$" som den refleksive transitive lukning af "$\Rightarrow$", dvs. $\alpha \Rightarrow^* \beta$ hvis og kun hvis $\alpha \underbrace{\Rightarrow \ldots \Rightarrow \ldots \Rightarrow}_{\text{0 eller flere derivationer}} \beta$
- Sproget af G defineres som $L(G) = \{x \in \Sigma^* | S \Rightarrow^* x\}$

Sproget for en CFG

- Definer relationen "$\Rightarrow^*$" som den refleksive transitive lukning af "$\Rightarrow$", dvs. $\alpha \Rightarrow^* \beta$ hvis og kun hvis $\alpha \underbrace{\Rightarrow \ldots \Rightarrow \ldots \Rightarrow}_{\text{0 eller flere derivationer}} \beta$
- Sproget af G defineres som $L(G) = \{x \in \Sigma^* | S \Rightarrow^* x\}$
- Et sprog $L \subseteq \Sigma^*$ er kontekstfrit hvis og kun hvis der findes en CFG G hvor $L(G) = L$

Eksempel 1

- Sproget $A = \{a^n b^n | n \geq 0\}$ kan beskrives af en CFG $G = (V, \Sigma, S, P)$ hvor

Eksempel 1

- Sproget $A = \{a^n b^n | n \geq 0\}$ kan beskrives af en CFG $G = (V, \Sigma, S, P)$ hvor
- $V = \{S\}$

Eksempel 1

- Sproget $A = \{a^n b^n | n \geq 0\}$ kan beskrives af en CFG $G = (V, \Sigma, S, P)$ hvor
- $V = \{S\}$
- $\Sigma = \{a, b\}$

Eksempel 1

- Sproget $A = \{a^n b^n | n \geq 0\}$ kan beskrives af en CFG $G = (V, \Sigma, S, P)$ hvor
- $V = \{S\}$
- $\Sigma = \{a, b\}$
- $P = \{S \rightarrow aSb, S \rightarrow \Lambda\}$ dvs. $L(G) = A$ (bevis følger...)

Eksempel 1

- Sproget $A = \{a^n b^n | n \geq 0\}$ kan beskrives af en CFG $G = (V, \Sigma, S, P)$ hvor
- $V = \{S\}$
- $\Sigma = \{a, b\}$
- $P = \{S \rightarrow aSb, S \rightarrow \Lambda\}$ dvs. $L(G) = A$ (bevis følger...)
- Alternativ notation: $S \rightarrow aSb|\Lambda$

Bevis for korrekthed

Vi vil bevise at $L(G) = A$

- Bevisskitse: (udnyt at $x \in L \Leftrightarrow S \Rightarrow^* x$)

Bevis for korrekthed

Vi vil bevise at $L(G) = A$

- Bevisskitse: (udnyt at $x \in L \Leftrightarrow S \Rightarrow^* x$)
- $L(G) \subseteq A$: Givet $x \in L(G)$ lav induktion i strukturen af derivationen i $S \Rightarrow^* x$

Bevis for korrekthed

Vi vil bevise at $L(G) = A$

- Bevisskitse: (udnyt at $x \in L \Leftrightarrow S \Rightarrow^* x$)
- $L(G) \subseteq A$: Givet $x \in L(G)$ lav induktion i strukturen af derivationen i $S \Rightarrow^* x$
- $A \subseteq L(G)$: Givet $x \in A$, lav induktion i længden af x (eller i n) og påvis en derivation.

Eksempel 2

Sproget $pal = \{x \in \{0,1\}^* | x = reverse(x)\}$ kan beskrives af en CFG $G = (V, \Sigma, S, P)$

- $V = \{S\}$

Eksempel 2

Sproget *pal* $= \{x \in \{0,1\}^* | x = \mathit{reverse}(x)\}$ kan beskrives af en CFG $G = (V, \Sigma, S, P)$

- $V = \{S\}$
- P: $S \rightarrow \Lambda | 0 | 1 | 0S0 | 1S1$

Eksempel 2

Sproget $pal = \{x \in \{0,1\}^* | x = reverse(x)\}$ kan beskrives af en CFG $G = (V, \Sigma, S, P)$

- $V = \{S\}$
- P: $S \rightarrow \Lambda | 0 | 1 | 0S0 | 1S1$
- Øvelse: Bevis at $L(G) = pal$

Eksempel 2

Sproget $pal = \{x \in \{0,1\}^* | x = reverse(x)\}$ kan beskrives af en CFG $G = (V, \Sigma, S, P)$

- $V = \{S\}$
- P: $S \rightarrow \Lambda | 0 | 1 | 0S0 | 1S1$
- Øvelse: Bevis at $L(G) = pal$
- Beviset er efter helt samme mønster som eksempel 1, blot med lidt flere tilfælde.

Hvorfor hedder det kontekstfri?

- $\alpha_1 A \alpha_2 \Rightarrow \alpha_1 \gamma \alpha_2$ hvis vi har produktionen $A \rightarrow \gamma$

Hvorfor hedder det kontekstfri?

- $\alpha_1 A \alpha_2 \Rightarrow \alpha_1 \gamma \alpha_2$ hvis vi har produktionen $A \to \gamma$
- Dvs. γ kan substituere A uanset konteksten (α_1, α_2).

Anvendelser af kontekstfri grammatikker

- Praktisk: til beskrivelse (og genkendelse/parsning) af programmeringssprog (ofte med BNF-notation).

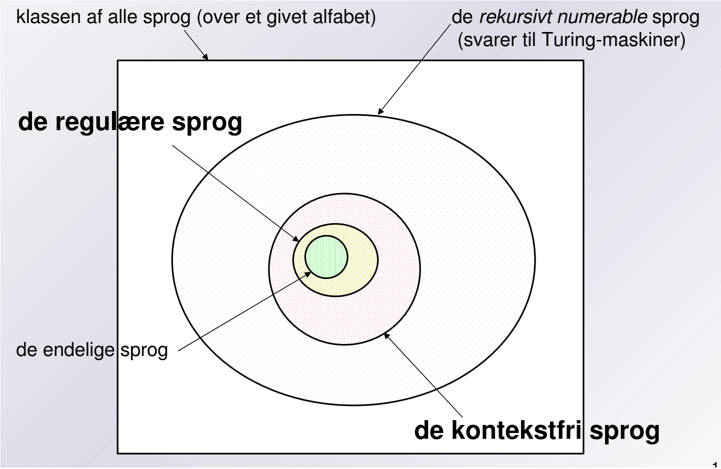
Anvendelser af kontekstfri grammatikker

- Praktisk: til beskrivelse (og genkendelse/parsning) af programmeringssprog (ofte med BNF-notation).
- Teoretisk: som karakteristik af en vigtig klasse af sprog.

Kontekstfri grammatik for Java

- `http://www.daimi.au.dk/dRegAut/JavaBNF.html`
- En tekst er et syntaktisk korrekt Java-program hvis den kan deriveres af denne grammatik

Klasser af formelle sprog



Øvelser

Martin 6.1 (a+b+e) (p. 240)

Øvelser

Martin 6.1 (a+b+e) (p. 240)

Martin 6.9 (a-c) (p. 243)

Regulære sprog er også kontekstfri 1/2

Der er simple konstruktioner for både $FA \rightarrow CFG$ og $regex \rightarrow CFG$

- Givet en et regulært udtryk r over alfabetet Σ kan vi konstruere en grammatik $G = (\{S\}, \Sigma, S, P)$ så $L(G) = L(r)$

Regulære sprog er også kontekstfri 1/2

Der er simple konstruktioner for både $FA \rightarrow CFG$ og $regex \rightarrow CFG$

- Givet en et regulært udtryk r over alfabetet Σ kan vi konstruere en grammatik $G = (\{S\}, \Sigma, S, P)$ så $L(G) = L(r)$
- $r = \emptyset : G = (\{S\}, \Sigma, S, \emptyset)$

Regulære sprog er også kontekstfri 1/2

Der er simple konstruktioner for både $FA \rightarrow CFG$ og $regex \rightarrow CFG$

- Givet en et regulært udtryk r over alfabetet Σ kan vi konstruere en grammatik $G = (\{S\}, \Sigma, S, P)$ så $L(G) = L(r)$
- $r = \emptyset : G = (\{S\}, \Sigma, S, \emptyset)$
- $r = \Lambda : G = (\{S\}, \Sigma, S, \{S \rightarrow \Lambda\})$

Regulære sprog er også kontekstfri 1/2

Der er simple konstruktioner for både $FA \rightarrow CFG$ og $regex \rightarrow CFG$

- Givet en et regulært udtryk r over alfabetet Σ kan vi konstruere en grammatik $G = (\{S\}, \Sigma, S, P)$ så $L(G) = L(r)$
- $r = \emptyset : G = (\{S\}, \Sigma, S, \emptyset)$
- $r = \Lambda : G = (\{S\}, \Sigma, S, \{S \rightarrow \Lambda\})$
- $r = a : G = (\{S\}, \Sigma, S, \{S \rightarrow a\})$

Regulære sprog er også kontekstfri 2/2

- $r = r_1 + r_2$:
  $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \rightarrow S_1, S' \rightarrow S_2\} \cup P_1 \cup P_2)$
  Hvor V_1, V_2 er nonterminalerne i G_1, G_2, for de mindre regulære udtryk, men omdøbt så der ikke er konflikter

Regulære sprog er også kontekstfri 2/2

- $r = r_1 + r_2$:
  $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \rightarrow S_1, S' \rightarrow S_2\} \cup P_1 \cup P_2)$
  Hvor V_1, V_2 er nonterminalerne i G_1, G_2, for de mindre regulære udtryk, men omdøbt så der ikke er konflikter
- $r = r_1 \cdot r_2$: $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \rightarrow S_1 S_2\} \cup P_1 \cup P_2)$

Regulære sprog er også kontekstfri 2/2

- $r = r_1 + r_2$:
  $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \to S_1, S' \to S_2\} \cup P_1 \cup P_2)$
  Hvor V_1, V_2 er nonterminalerne i G_1, G_2, for de mindre regulære udtryk, men omdøbt så der ikke er konflikter
- $r = r_1 \cdot r_2$: $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \to S_1 S_2\} \cup P_1 \cup P_2)$
- $r = r_1^*$: $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \to S_1, S' \to S' S_1, S' \to \Lambda\} \cup P_1 \cup P_2)$

Regulære sprog er også kontekstfri 2/2

- $r = r_1 + r_2$:
  $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \to S_1, S' \to S_2\} \cup P_1 \cup P_2)$
  Hvor V_1, V_2 er nonterminalerne i G_1, G_2, for de mindre regulære udtryk, men omdøbt så der ikke er konflikter
- $r = r_1 \cdot r_2$: $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \to S_1 S_2\} \cup P_1 \cup P_2)$
- $r = r_1^*$: $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \to S_1, S' \to S' S_1, S' \to \Lambda\} \cup P_1 \cup P_2)$
- Beviset for korrekthed er naturligvis induktion i strukturen af det regulære udtryk / strukturen af derivationen af $x \in G$.

Regulære sprog er også kontekstfri 2/2

- $r = r_1 + r_2$:
  $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \rightarrow S_1, S' \rightarrow S_2\} \cup P_1 \cup P_2)$
  Hvor V_1, V_2 er nonterminalerne i G_1, G_2, for de mindre regulære udtryk, men omdøbt så der ikke er konflikter
- $r = r_1 \cdot r_2$: $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \rightarrow S_1 S_2\} \cup P_1 \cup P_2)$
- $r = r_1^*$: $G = (\{S'\} \cup V_1 \cup V_2, \Sigma, S', \{S' \rightarrow S_1, S' \rightarrow S' S_1, S' \rightarrow \Lambda\} \cup P_1 \cup P_2)$
- Beviset for korrekthed er naturligvis induktion i strukturen af det regulære udtryk / strukturen af derivationen af $x \in G$.
- Bemærk dette viser også at Kontekstfri sprog er lukket under $\cup$, konkatenering og *

Plan

- Lukketheds- og afgorlighedsegenskaber
- Pumping lemmaet
- Afgørlighedsegenskaber
- Kontekstfrie Grammatikker
- Afrunding, og om eksamen

Eksamen

- Der er 6 eksamensspørgsmål, man trækker et når man kommer ind.
- Eksamen varer ca. 20 min inklusiv votering. Det er meget kort tid, så det er godt at have en konkret plan til hvert emne
- Karakterer på den nye 7-trinsskala.
- Brug endelig tavlen.

Eksamensspørgsmål

Her er forslag til hvilke emner man kunne komme ind på, bemærk dette er kun *forslag*, og ikke bud på en færdig plan.

- regulære udtryk (definition, skitse af Kleenes sætning, lav konstruktion $regex \rightarrow NFA - \Lambda$ og/eller $FA \rightarrow regex$)

Eksamensspørgsmål

Her er forslag til hvilke emner man kunne komme ind på, bemærk dette er kun *forslag*, og ikke bud på en færdig plan.

- regulære udtryk (definition, skitse af Kleenes sætning, lav konstruktion $regex \rightarrow NFA - \Lambda$ og/eller $FA \rightarrow regex$)
- endelige automater (definition, produktkonstruktionen, dele af Kleene's sætning)

Eksamensspørgsmål

Her er forslag til hvilke emner man kunne komme ind på, bemærk dette er kun *forslag*, og ikke bud på en færdig plan.

- regulære udtryk (definition, skitse af Kleenes sætning, lav konstruktion $regex \rightarrow NFA - \Lambda$ og/eller $FA \rightarrow regex$)
- endelige automater (definition, produktkonstruktionen, dele af Kleene's sætning)
- lukkethedsegenskaber (produktkonstruktionen, pumping-lemma, homomorfier)

Eksamensspørgsmål

Her er forslag til hvilke emner man kunne komme ind på, bemærk dette er kun *forslag*, og ikke bud på en færdig plan.

- regulære udtryk (definition, skitse af Kleenes sætning, lav konstruktion $regex \to NFA - \Lambda$ og/eller $FA \to regex$)
- endelige automater (definition, produktkonstruktionen, dele af Kleene's sætning)
- lukkethedsegenskaber (produktkonstruktionen, pumping-lemma, homomorfier)
- nondeterministiske automater (definition af NFA'er, determinisering, lambda-eliminering)

Eksamensspørgsmål

Her er forslag til hvilke emner man kunne komme ind på, bemærk dette er kun *forslag*, og ikke bud på en færdig plan.

- regulære udtryk (definition, skitse af Kleenes sætning, lav konstruktion $regex \rightarrow NFA - \Lambda$ og/eller $FA \rightarrow regex$)
- endelige automater (definition, produktkonstruktionen, dele af Kleene's sætning)
- lukkethedsegenskaber (produktkonstruktionen, pumping-lemma, homomorfier)
- nondeterministiske automater (definition af NFA'er, determinisering, lambda-eliminering)
- minimering af automater (Uskelnelighed, MyHill Nerode, Minimeringsalgoritmen, L_{42} har 2^{42} ækvivalensklasser)

Eksamensspørgsmål

Her er forslag til hvilke emner man kunne komme ind på, bemærk dette er kun *forslag*, og ikke bud på en færdig plan.

- regulære udtryk (definition, skitse af Kleenes sætning, lav konstruktion $regex \to NFA - \Lambda$ og/eller $FA \to regex$)
- endelige automater (definition, produktkonstruktionen, dele af Kleene's sætning)
- lukkethedsegenskaber (produktkonstruktionen, pumping-lemma, homomorfier)
- nondeterministiske automater (definition af NFA'er, determinisering, lambda-eliminering)
- minimering af automater (Uskelnelighed, MyHill Nerode, Minimeringsalgoritmen, L_{42} har 2^{42} ækvivalensklasser)
- begrænsninger af regulære sprog (pumping-lemma, eksempler på ikke-regulære sprog, kontekstfrie grammatikker, klasser af sprog)