Lukkeheds- og afgørligheds egenskaber

- 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 og L_2,

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

' betyder *komplement* som i bogen...

Ja! – det beviste vi første seminar

En kontraponering

- Lukkethedsegenskaber kan bl.a. bruges til at vise, at visse sprog er **ikke-regulære**

- Eksempel:
  - 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

[Martin, opg.4.46]

- 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 $h(L) = \{\ h(x) \mid x \in L\ \}$ for alle $L \subseteq \Sigma_1^*$
- og $h^{-1}(L) = \{\ x \mid h(x) \in L\ \}$ for alle $L \subseteq \Sigma_2^*$

Eksempel

- $\Sigma_1=\{0,1\}$, $\Sigma_2=\{a,b\}$
- Lad g: $\Sigma_1 \rightarrow \Sigma_2^*$ være defineret ved
  - $g(0)=ab$
  - $g(1)=\Lambda$
- Lad h: $\Sigma_1^* \rightarrow \Sigma_2^*$ være homomorfien defineret fra g
- Der gælder f.eks.:
  - $h(0011) = abab\Lambda\Lambda = abab$
  - $h(\{1\}\{0\}^*\{1\}) = \{\Lambda\}\{ab\}^*\{\Lambda\} = \{ab\}^*$
  - $h^{-1}(\{ab\}^*) = \{0,1\}^*$

Regulære sprog og homomorfier

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

- 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

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

Eksempel på anvendelse

- Er følgende sprog over alfabetet $\Sigma=\{0,1,2\}$ regulært?

  $$L = \{\ x2y \mid y=reverse(x),\ x,y\in\{0,1\}^*\ \}$$

- Vi ved (fra første seminar) at sproget

  $$pal = \{\ x\in\{0,1\}^* \mid x=reverse(x)\ \}$$

  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 g_1,g_2,g_3: $\{0,1,2\}\to\{0,1\}^*$ ved

$$
\begin{array}{lll}
g_1(0)=0 & g_2(0)=0 & g_3(0)=0 \\
g_1(1)=1 & g_2(1)=1 & g_3(1)=1 \\
g_1(2)=\Lambda & g_2(2)=\mathbf{0} & g_3(2)=\mathbf{1}
\end{array}
$$

- og lad h_1,h_2,h_3 være de tilhørende homomorfier

- $h_1(L) \cup h_2(L) \cup h_3(L) = pal$

- så L er **ikke** regulært, idet *pal* ikke er regulært og klassen af regulære sprog er lukket under forening og homomorfi

Bevis, del 1

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

Bevis:

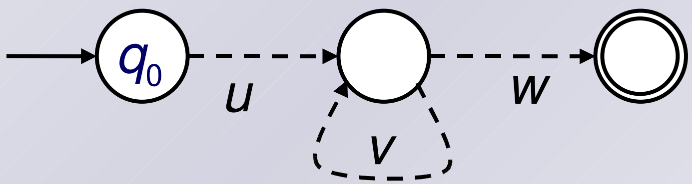
Givet en FA $M=(Q, \Sigma_2, q_0, A, \delta)$ hvor $L(M)=L$, definer en ny FA $M'=(Q, \Sigma_1, q_0, A, \delta')$ ved

$$\delta'(q, a) = \delta^*(q, h(a))$$

Påstand: $L(M') = h^{-1}(L)$ (bevises ved induktion)

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 én gang



$x = uvw$

- Hvis vi betragter den *første* af disse tilstande kan vi konkludere:

$$\exists u,v,w \in \Sigma^*: \; x=uvw \;\wedge\; |uv| \leq |Q| \;\wedge\; |v|>0 \;\wedge\; \delta^*(q_0, u)=\delta^*(q_0, uv)$$

"Pumping"-lemmaet for regulære sprog

Hvis L er et regulært sprog, så gælder flg.:

$\exists n > 0$:

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

$\exists u, v, w \in \Sigma^*$: $x = uvw \;\wedge$

$|uv| \leq n \;\wedge$

$|v| > 0 \;\wedge$

$\forall m \geq 0$: $uv^m w \in L$

Bevis: vælg n som antal tilstande i en FA, der genkender L, og "kør m gange rundt i løkken"...

Pumping-lemmaet og ikke-regulære sprog

Hvis

$\forall n > 0$:

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

$\forall u, v, w \in \Sigma^*$ hvor $x = uvw$, $|uv| \leq n$ og $|v| > 0$:

$\exists m \geq 0$: $uv^m w \notin L$

så er L **ikke** regulært

Bevis: kontraponering af pumping-lemmaet

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 $L = \{ 0^i 1^i \mid i \geq 0 \}$

Vi vil vise vha. pumping-lemmaet 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

$$v=0^k \text{ for et } k>0$$

- dvs. $uv^m w = uv^2 w = 0^{n+k} 1^n \notin L$
- så L er ikke regulært

Eksempel 2

Lad *pal* = $\{ x \in \{0,1\}^* \mid x = reverse(x) \}$ (som uge 15)

Vi vil vise vha. pumping-lemmaet 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

$$v=0^k \text{ for et } k>0$$

- dvs. $uv^mw = uv^2w = 0^{n+k}10^n \notin pal$
- så *pal* er ikke regulært

Eksempel 3

Lad $L = \{ 0^i \mid i$ er et primtal $\}$

Vi vil vise vha. pumping-lemmaet at L ikke er regulært

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

- $|uv^m w| = |uv^{p-k}w| = |uw| + (p-k)\cdot|v| = p-k + (p-k)\cdot k$
  $= (k+1)\cdot(p-k)$ og begge disse led er >1,
  dvs. $|uv^m w|$ er ikke et primtal
- så L er ikke regulært

Mere om pumping-lemmaet

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

- Eksempel:

$$L = \{ a^i b^j c^j \mid i \geq 1 \text{ og } j \geq 0 \} \cup \{ b^j c^k \mid j,k \geq 0 \}$$

  - L er *ikke* regulært, men
  - L har pumping-egenskaben

    (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)

Program

- MyHill-Nerode-sætningen (resume)
- En algoritme til minimering af FA'er
- Lukkethedsegenskaber
- **Afgørlighedsegenskaber**
- Kontekstfri grammatikker

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 $x \in L(M)$?)

Algoritme:

Kør x på M, startende i starttilstanden, og se om den ender i en accepttilstand

Emptiness-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~~

Algoritme 1:

Afprøv for alle *x* **hvor $|x|<|Q|$** om $x\in L(M)$ ved hjælp af algoritmen fra *membership*-problemet

en ***reduktion*** til *membership*-problemet

~~**Algoritme 2:**~~

~~Undersøg om der findes en accepttilstand~~

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?)

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...)

Algoritme 2:

Ide: Udnyt at $L(M)$ er uendeligt hvis og kun hvis der i 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 $L(M_1) \subseteq L(M_2)$?)

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 $L(M_3) = \varnothing$

(Bevis for korrekthed: $L(M_1) \subseteq L(M_2) \iff L(M_1) - L(M_2) = \varnothing$)

Equality-problemet

Givet to FA'er, M_1 og M_2, er sprogene for M_1 og M_2 ens?

(Dvs. er $L(M_1)=L(M_2)$?)

Algoritme:

Afgør med algoritmen til *subset*-problemet om $L(M_1) \subseteq L(M_2)$ og $L(M_2) \subseteq L(M_1)$

Øvelser

- [Martin] 5.26 (a+e)
- [Martin] 5.28 (a+b+d+g)

dRegAut Java-pakken

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

beslutningsprocedurer for de nævnte beslutningsproblemer

- `FA.getAShortestExample()`
  – finder en korteste sti fra starttilstanden til en accepttilstand (hvis sproget er ikke-tomt)

getAShortestExample

```
String getAShortestExample() {
    pending = [ q0 ]              // queue of states that need to be visited
    paths = [ "" ]                // paths[i] is a shortest path from q0 to pending[i]
    visited = { q0 }              // set of states that have been visited
    while pending≠Ø do
        q = pending.removeFirst()
        path = paths.removeFirst()
        if q∈A then return path
        else
            for each c∈Σ do
                p = δ(q, c)
                if p∉visited
                    pending.addToEnd(p)
                    paths.addToEnd(path++c)
                    visited = visited ∪ {p}
    return null      // return null if no accept state is found
}
```

"bredde-først gennemløb" af automaten

Status

- Regulære udtryk og endelige automater ✓
- Kontekstfri grammatikker