IN2080 Beregninger og kompleksitet

Løsningsforslag, hjemmeeksamen våren 2020
Oppgave 11 og 14-20

Teorem 1 (Pumpelemma for bulgarske språk) *La L være et bulgarsk språk. Da finnes det et tall p (pumpelengden) med følgende egenskap: For enhver streng s i L som er ekte lengre enn p, så finnes det strenger u, v, x, y, z som tilfredstiller*

1. $s = uvxyz$
2. $uv^i xy^i z \in L$ *for enhver* $i \geq 0$
3. $|vy| > 0$
4. $|uv| \leq p$ *og* $|yz| \leq p$.

Oppgave 11. Vis teorem 1.

Bevis av pumpelemma. La L være et bulgarsk språk. Da finnes det en bulgarsk grammatikk $G = (V, \Sigma, R, S)$ slik at $L = L(G)$. Vi kan uten tap av generalitet anta at ingen av reglene i R er på formen $X_1 \to X_2$ der $X_1, X_2 \in V$ (se normalformteorem).

Enhver utledning i G er på formen

$$S \Rightarrow s_1 A_1 t_1 \Rightarrow s_2 A_2 t_2 \Rightarrow \ldots \Rightarrow s_m A_m t_m \Rightarrow w$$

der $A_1, \ldots A_m \in V$ og $s_1, \ldots s_m, t_1, \ldots t_m, w \in \Sigma^*$. La oss si at en utledning er *enkel* dersom ingen av variablene i V forekommer (ekte) mer enn to ganger i sekvensen $S, A_1, A_2, \ldots A_m$. Mengden av strenger som har en enkel utledning er opplagt endelig. La p være lengden til den lengste strengen som har en enkel utledning.

La $w \in L$ og $|w| > p$. Da har ikke w en enkel utledning. Dermed må det finnes $B \in V$ slik at

$$\begin{aligned} S \Rightarrow s_1 A_1 t_1 \Rightarrow \ldots \Rightarrow s_m A_m t_m \Rightarrow s_{m+1} B t_{m+1} \\ \Rightarrow s_{m+1} v_1 C_1 y_1 t_{m+1} \Rightarrow \ldots \Rightarrow s_{m+1} v_k C_k y_k t_{m+1} \\ \Rightarrow s_{m+1} v_{k+1} B y_{k+1} t_{m+1} \Rightarrow_* s_{m+1} v_{k+1} x y_{k+1} t_{m+1} = w \end{aligned} \tag{1}$$

der

- $A_1, \ldots A_m, B, C_1, \ldots C_k$ er forskjellige variabler
- $|v_{k+1} y_{k+1}| > 0$
- det finnes en streng x_0 slik at $s_{m+1} v_{k+1} x_0 y_{k+1} t_{m+1}$ har en enkel utledning.

Siden $s_{m+1} v_{k+1} x_0 y_{k+1} t_{m+1}$ har en enkel utledning, så har vi

$$|s_{m+1} v_{k+1} x_0 y_{k+1} t_{m+1}| \leq p \ .$$

Dermed har vi også

$$|s_{m+1} v_{k+1}| \leq p \quad \text{og} \quad |y_{k+1} t_{m+1}| \leq p \ .$$

Dette viser at teorem 1 holder når $u = s_{m+1}$ og $v = v_{k+1}$ og $y = y_{k+1}$ og $z = t_{m+1}$ og x er gitt ved utledningen (1) (slutt på bevis av pumpelemma).

Teorem 2 (Normalform for bulgarske gramatikker) *For enhver bulgarsk grammatikk G finnes en bulgarsk grammatikk G' på normalform slik at $L(G) = L(G')$.*

Kompleksitetsteori

Språket A er gitt ved

$$A \ = \ \{ \ G \mid G \text{ er en regulær grammatikk og } L(G) \neq \emptyset \ \} \ .$$

Husk at $\overline{A}$ betegner komplementspråket til A.

Oppgave 14. Vis at A er NL-komplett.

Løsningsforslag oppgave 14.

Vi viser

- (I) at A er i NL
- (II) at $PATH \leq_L A$.

Siden $PATH$ er NL-komplett, så følger det at A er NL-komplett.

Vi ser på en algoritme som avjør A:

`INPUT:` en regulær grammatikk $G = (V, \Sigma, R, S)$
– $N := \langle$antall regler i $R\rangle - 1$
      (reglene i R må telles)
– Velg nondeterministisk en regel på formen $S \to w$
      (S er startvariabelen)
– Hvis $w = \varepsilon$, så `AKSEPTER INPUT`
`while true do`
  `begin`
  – La B være variabelen som forekommer i w
      (w er høyre side av sist valgte regel)
  – Velg nondeterministisk en regel fra R på formen $B \to w$
  – Hvis det ikke finnes en regel på denne formen, så `AVVIS INPUT`
  – Hvis $w = \varepsilon$, så `AKSEPTER INPUT`
  – $N := N - 1$
  – Hvis $N = 0$, så `AVVIS INPUT`
  `end`

Det er ganske opplagt at finnes en eksekvering av algoritmen over som aksepterer G hvis og bare hvis det finnes en streng w som er med i $L(G)$. Det er også opplagt at algoritmen alltid terminerer. For å konkludere at (I) holder, må vi vise at algoritmen kan utføres av en nondeterministisk TM som arbeider i rom $O(\log n)$ der n er lengden av input.

Algoritmen teller antall regeler N i R. Vi representerer N binært, og N vil være mindre enn n. Dermed kan dette gjøres av en TM som arbeider i rom $O(\log n)$. Algoritmen vil (nondeterministisk) velge regler fra R på en viss form. Dette kan gjøres ved hjelp av et endelig antall pekere inn i input. Dermed kan algoritmen utføres av en TM som arbeider i rom $O(\log n)$. Dette avslutter vårt bevis av (I).

Nå skal vi vise (II). Vi har

$$
\begin{aligned}
\mathit{PATH} \;=\; \{\, & \langle H, s, t\rangle \mid \\
& H \text{ er en rettet graf hvor det går en rettet sti fra } s \text{ til } t \,\} \,.
\end{aligned}
$$

Vi definererer en reduksjon f slik at

$$\langle H, s, t\rangle \in \mathit{PATH} \;\Leftrightarrow f(\langle H, s, t\rangle) \in A \,.$$

Funksjonen f skal være beregnbar av en "log space transducer".
La $f(\langle H, s, t\rangle) = G = (V, \Sigma, R, S)$ der

- $$V = \{A_i \mid i \text{ er en node i } H\}$$

- $$\Sigma = \emptyset$$

- $$R = \{A_i \to A_j \mid (i, j) \text{ er en kant i } H\} \cup \{A_t \to \varepsilon\}$$

- $$S = A_s \,.$$

Grammatikken G genererer ε (den tomme strengen) hvis det finnes en rettet sti fra s til t i H. Hvis det ikke finnes en slik sti, så vil ikke G generere noe som helst (vi har $L(G) = \emptyset$). Det er lett å se at en "log space transducer" kan bergne f:

- Ved å løpe gjennom nodene i H kan transduseren skrive V til outputteipen (den trenger ikke bruke arbeidsteip).

- Ved å løpe gjennom kantene i H kan transduseren skrive regler i R til outputteipen (den trenger ikke bruke arbeidsteip).

- I tillegg trenger transduser kun å skrive regelen $A_t \to \varepsilon$ og startsymbolet A_s til outputteipen.

Oppgave 15. Besvar følgende fire spørsmål:

1. Er A i P? **JA.** Det vises i Sipser's bok at NL $\subseteq$ P, og ved oppgave 14 har vi $A \in$ NL.

2. Er $\overline{A}$ i coNL? **JA.** Ved oppgave 14 har vi $A \in$ NL. Dermed har vi $\overline{A} \in$ coNL ved definisjonen av coNL.

3. Er $\overline{A}$ i NL? **JA.** Det vises i Sipser's bok at at NL = coNL, og vi har $\overline{A} \in$ coNL.

4. Er A i L? Ved oppgave 14 har vi at A er NL-komplett. Dermed, har vi $A \notin$ L hvis NL $\neq$ L. Hvis NL = L, så har vi selvsagt $A \in$ L. Den almindelige oppfatningen er at NL $\neq$ L, men ingen har klart å vise dette.

Svar kort, men begrunn svarene. Du må gjerne referere til teoremer og lignende i Sipsers bok [1]. Det er ikke mulig å besvare hvert eneste spørmål med et definitivt JA eller et definitivt NEI.

Oppgave 16. Vis at ethvert bulgarsk språk er i NL. Hint: Bruk teorem 2.

Løsningsforslag oppgave 16. La L være et bulgarsk språk. La $L = L(G)$ der $G = (V, \Sigma, R, S)$ er en bulgarsk grammatikk på normalform. En slik grammatikk finnes ved teorem 2. Vi skal snart beskrive en nondeterministisk algoritme som er basert på G. Algoritmen avgjør L. Algoritmen benytter en inputteip og en arbeidsteip. Merk at grammatikken G ikke er input til algoritmen. Det er rimelig lett å se at algoritmen kan utføres av en turingmaskinen arbeider i rom $O(\log n)$ der n er lengden av input.

Notasjon: Algoritmen vil operere med to tellere V (venstre) og H (høyre) på arbeidsteipen. Disse vil tjene som pekere inn i input w der $w = w_1 w_2 \dots w_n$ og $w_1, \dots, w_n \in \Sigma$. Av bekvemlighetshensyn antar vi $1 \leq n$, dvs., vi antar $w \neq \varepsilon$. Vi bruker notasjonen w_V for inputsymbolet V peker på og notasjonen w_H for inputsymbolet H peker på.

Her er algoritmen:

```
INPUT:  w = w_1, ..., w_n der w_1, ..., w_n ∈ Σ
– V := 1 og H := n
– Velg nondet. en regel i R på formen S → u der u ∈ (V ∪ Σ)*
            (S er G's startvariabel)
while V ≤ H do
  begin
  – Hvis u = ε, så AVVIS INPUT
  if ⟨ u er på formen aA der a ∈ Σ og A ∈ V ⟩ then
    begin
    – Hvis w_V ≠ a, så AVVIS INPUT
    – V := V + 1
    – Velg nondet. en regel i R på formen A → u der u ∈ (V ∪ Σ)*
    – Hvis en slik regel ikke finnes, så AVVIS INPUT
    end
  else if ⟨ u er på formen Aa der a ∈ Σ og A ∈ V ⟩ then
    begin
    – Hvis w_H ≠ a, så AVVIS INPUT
    – H := H − 1
    – Velg nondet. en regel i R på formen A → u der u ∈ (V ∪ Σ)*
    – Hvis en slik regel ikke finnes, så AVVIS INPUT
    end
  end
– Hvis u = ε, så AKSEPTER INPUT
– Hvis u ≠ ε, så AVVIS INPUT
```

Beregnbarhetsteori (bonusoppgave)

La Σ være et alfabet. Språket $H \subseteq \Sigma^*$ er gitt ved

$$H = \{ \langle M, w, n \rangle \mid M \text{ er en turingmaskin som bruker færre enn } n \text{ trinn på å akseptere eller avise input } w \}.$$

Vi holder oss til konvensjonen om at et (naturlig) tall skal representeres binært på en turingmaskins inputteip.

Oppgave 17. Er H et gjenkjennbart[1] språk? Er H et avgjørbart[2] språk? Er H et språk i kompleksitetsklassen P? Svar kort, men begrunn svarene.

Løsningsforslag oppgave 17. La oss føst slå fast at H er avgjøbart: *En turingmaskin M_0 kan simulere eksekveringen av M på input w. Hvis eksekveringen terminerer innen n tinn, så aksepterer M_0 input $\langle M, w, n\rangle$ (en eksekvering av M vil alltid terminere ved at M enten aksepterer eller aviser input). Hvis eksekveringen ikke terminerer innen n tinn, så avviser M_0 input $\langle M, w, n\rangle$.* Dermed er H et avgjørbart språk.

Alle avgjørbare språk er gjenkjennbare. Dermed er H et gjenkjennbart språk.

Språket H er ikke i P. For å avjøre medlemskap i H må en turingmaskin M_0 simulere n trinn av eksekvering av M på input w. Siden n er representert binært, så vil antall trinn som M_0 simulerer være eksponestelt i lengden av input. Dermed vil ikke M_0 arbeide i polynom tid. (Dette er en god begrunnelse for at H ikke er i P, men det er intet bevis. Det kan bevises at H ikke er i P, men det er meget vanskelig...og det forventes overhodet ikke at en eksamenskandidat skal bevise noe slikt.)

Vi definerer funksjonen f ved

$$f(\langle M, w\rangle) \;=\; \begin{cases} n+1 & \text{der } n \text{ er den minste } n \text{ slik at } \langle M, w, n\rangle \in H \\ 0 & \text{hvis en slik } n \text{ ikke finnes.} \end{cases}$$

Språket H' er gitt ved

$$H' \;=\; \{\ \langle M, w\rangle \mid \langle M, w, f(M, w)\rangle \in H\ \}\ .$$

Oppgave 18. Er H' et gjenkjennbart språk? Er H' et avgjørbart språk? Er H' et språk i kompleksitetsklassen P? Svar kort, men begrunn svarene.

Løsningsforslag oppgave 18. Språket H' er gjenkjennbart: *En turingmaskin M_1 kan simulere eksekveringen av M på input w. Hvis eksekveringen terminerer, så aksepterer M_1 input $\langle M, w\rangle$. Hvis eksekveringen ikke terminerer, så vil simuleringen fortsette til evig tid.* Språket H' er ikke avgjørbart: *Hvis H' hadde vært avgjørt, så ville også språket A_{TM} ha vært*

[1] Et *gjenkjennbart språk* kalles *a Turing-recognizable language* i Sipsers bok [1], se Def. 3.5 side 170.

[2] Et *avgjørbart språk* kalles *a Turing-decidable language* i Sipsers bok [1], se Def. 3.6 side 170.

avgjørbart. Språket A_{TM} er mye omtalt i Sipsers bok, og det vises at A_{TM} ikke er avgjørbart. Alle språk i P er avgjørbare. Dermed er ikke H' i P.

La $g : \Sigma^* \to \Sigma^*$ være en total og beregnbar[3] funksjon. (At g skal være total betyr at $g(w)$ er definert for enhver $w \in \Sigma^*$.)

Oppgave 19. Vis at det finnes en turingmaskin M slik at $|g(w)| < f(M, w)$ (for alle $w \in \Sigma^*$).

Løsningsforslag oppgave 19. La M utføre følgende operasjoner:

`INPUT:` w
−Beregn $g(w)$ og skriv resultatet tegn for tegn på arbeidsteipen
    (antall trinn i denne prosessen må nødvendigvis være større enn $|g(w)|$)
    (prosessen terminerer siden g er en total)
− `AKSEPTER INPUT`
    (så M akepterer alle input, men dette er av underordnet betydning)
    (det viktige er at M stopper på ehnver input)

Da har vi $|g(w)| < f(M, w)$.

Oppgave 20. Forklar kort hvorfor f ikke er en beregnbar funksjon.

Løsningsforslag oppgave 20. Et svar a la det følgende vil være godt nok: Hvis f hadde vært en beregnbar funskjon, så hadde A_{TM} vært et avgjørbart språk.

Her er et mer utførlig svar: Anta (i den hensikt å utlede en selvmotsigelse) at f er beregnbar. La M_2 være en TM som gjør følgende:

`INPUT:` $\langle M, w\rangle$
−Beregn $f(M, w)$
− Hvis resultatet av beregningen ble 0, så `AVVIS INPUT`
    (hvis resultatet ikke ble 0, så vil M terminere på input w)
− Simuler M på input w
− Hvis M aksepterer w, så `AKSEPTER INPUT`
− Hvis M avviser w, så `AVVIS INPUT`

Vi ser at M_2 avgjør A_{TM}. Dermed er A_{TM} et avgjørbart språk. Det vises i Sipsers bok at A_{TM} ikke er et avgjørbart språk. Selvmotsigelse!

[3] Se Def. 5.17 side 234 i Sipser [1].

References

[1] Michael Sipser: *Introduction to the Theory of Computation.* 3rd Edition, International Edition, Cengage Learning, 2013.