INSTITUT FOR MATEMATIK OG DATALOGI

ODENSE UNIVERSITET

Skriftlig eksamen

D M 1 5

Fredag den 9. juni 1995 kl. 15.00 - 19.00

Opgavesættet består af 6 nummererede sider og indeholder 4 opgaver.

Opgaverne vægtes sådan:

- Opgave 1: 25%
- Opgave 2: 35%
- Opgave 3: 25%
- Opgave 4: 15%

Alle skriftlige hjælpemidler samt brug af lommeregner er tilladt.

Opgave 1

Denne opgave handler om regulære sprog.

Lad $f : \Sigma^* \to \Sigma^*$ være den funktion, der duplerer hvert eneste tegn. D.v.s. at

$$f(y_1 y_2 \dots y_n) = y_1 y_1 y_2 y_2 \dots y_n y_n$$

hvor $y_i \in \Sigma$ for $1 \leq i \leq n$.

Lad $g : \Sigma^* \to \Sigma^*$ være den funktion, der laver n kopier af hvert tegn, hvor n er længden af argumentstrengen. D.v.s. at

$$g(y_1 y_2 \dots y_n) = \underbrace{y_1 y_1 \dots y_1}_{n \text{ gange}} \ \underbrace{y_2 y_2 \dots y_2}_{n \text{ gange}} \ \dots \ \underbrace{y_n y_n \dots y_n}_{n \text{ gange}}$$

hvor $y_i \in \Sigma$ for $1 \leq i \leq n$.

For eksempel er $f(aaba) = aaaabbaa$ og $g(aaba) = aaaaaaaabbbbaaaa$.

spørgsmål a

Tegn tilstandsgrafen for den nondeterministiske endelige automat med λ-overgange (NFA-λ), der er defineret sådan: $M_1 = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b, c, d\}, \delta, q_0, \{q_5\})$, hvor

$$\begin{array}{lll}
\delta(q_0, a) = q_1 & \delta(q_0, \lambda) = q_2 & \\
\delta(q_1, a) = q_2 & & \\
\delta(q_2, d) = q_3 & \delta(q_2, b) = q_4 & \delta(q_2, \lambda) = q_5 \\
\delta(q_3, d) = q_2 & & \\
\delta(q_4, b) = q_5 & & \\
\delta(q_5, c) = q_5 & &
\end{array}$$

Angiv en regulær grammatik for det sprog, der accepteres af M_1.

spørgsmål b

Lad $M_2 = (Q, \Sigma, \delta, q_0, F)$ være en DFA. Definer v.h.a. M_2 en NFA-λ, der accepterer

$$\{x \in \Sigma^* \mid x = f(y) \text{ for mindst ét } y \in L(M_2)\}.$$

spørgsmål c

Vis, at selv om L er et regulært sprog, så er $L' = \{g(x) \mid x \in L\}$ ikke nødvendigvis et regulært sprog.

spørgsmål d

Find et regulært sprog L, sådan at $L' = \{g(x) \mid x \in L\}$ også er et regulært sprog.

Opgave 2

Denne opgave handler om deterministiske PDA'er og MIRANDA. I de første to spørgsmål i denne opgave skal man implementere en simulator for deterministiske PDA'er i MIRANDA. Simulatoren skal virke for deterministiske PDA'er, der ikke har nogen transitioner af formen $\delta(q_i, \lambda, x) = (q_j, y)$, hvor $x, y \in \Gamma \cup \{\lambda\}$. D.v.s. at hver eneste transition bruger ét symbol fra input.

Antag, at PDA'ernes tilstande er af typen num (og at de ikke er negative), og at både inputsymbolerne og staksymbolerne er af typen char. Antag, at symbolet '!' ikke bruges af PDA'erne. Vi kan så bruge '!' til at repræsentere λ på stakken. Antag, at 0 er starttilstanden. Så kan vi benytte følgende typedefinitioner i MIRANDA for transitioner, mængder af sluttilstande, PDA'er o.s.v:

- state == num
- input == [char]
- stack == [char]
- transition == (state,char,char,state,char)
- transitions == [transition]
- final == [state]
- pda_type == (transitions,final)

Så en transition $\delta(q_i, a, A) = (q_j, B)$ skrives som $(i, 'a', 'A', j, 'B')$, og $\delta(q_i, a, \lambda) = (q_j, B)$ skrives som $(i, 'a', '!', j, 'B')$.

spørgsmål a

Skriv en funktion *move* med typen

- move :: transitions $\rightarrow$ state $\rightarrow$ input $\rightarrow$ stack $\rightarrow$ (state, input, stack)

Resultatet af *move trans i (x:str) stack1* skal være *(j,str,stack2)*, hvis der findes en transition i *trans*, der specificerer, at når PDA'en er i tilstand i, læser symbolet x på input og har stakken *stack1*, så kan der skiftes til tilstand j og stakken har udseendet *stack2*.

Hvis der gøres forsøg på at fjerne et element fra en tom stak, eller hvis der ikke er en transition, der kan anvendes (givet den øjeblikkelige tilstand, første symbol på input samt staktoppen), så skal der skiftes til tilstand -1.

spørgsmål b

Skriv en funktion *accept* med typen

- accept :: pda_type → input → bool

Resultatet af *accept pda str* skal være *True*, hvis PDA'en *pda* accepterer strengen *str* og *False* ellers. Der skal benyttes sluttilstandsaccept (acceptance by final state). D.v.s. det er ikke nødvendigt, at stakken tømmes. Det er tilstrækkeligt, at alt input læses, hvorefter der standses i en sluttilstand.

Det vil sikkert være nyttigt først at skrive en funktion *run* med typen

- run :: pda_type → input → (state,input,stack)

som kører PDA'en på inputstrengen, indtil den standser. Bemærk, at hvis hele inputstrengen er læst, så vil tuplens midterste komponent være en tom liste.

spørgsmål c

Antag, at M er en deterministisk PDA (nu som defineret i lærebogen af Sudkamp) og overvej følgende spørgsmål i forbindelse med en kørsel af M på input w.

- Er det muligt, at PDA'en kan køre uendeligt? Forklar.
- Kan antallet af symboler på stakken vokse efter alt input er blevet læst? Forklar.
- Kan antallet af symboler på stakken aftage efter alt input er blevet læst? Forklar.

spørgsmål d

Beskriv en algoritme, der afgør, om en deterministisk PDA M standser på input w.

Opgave 3

Denne opgave drejer sig om den talteoretiske funktion $h(n) = \sum_{i=0}^{n} i^2$.
For eksempel er $h(4) = 0^2 + 1^2 + 2^2 + 3^2 + 4^2 = 0 + 1 + 4 + 9 + 16 = 30$.

spørgsmål a

Bevis, at sproget $\{1^{h(n)} \mid n \geq 0\}$ ikke er kontekst-frit.

spørgsmål b

Vis, at h er μ-rekursiv. Man må gerne benytte funktionerne *add(x,y)* and *mult(x,y)*, som er defineret i lærebogen af Sudkamp.

spørgsmål c

Skriv en funktion i MIRANDA, der beregner $h(n)$.

spørgsmål d

Skriv en funktion i PCF, der beregner $h(n)$. Man må gerne benytte funktionerne *mult* og *pred*, som er defineret i noten af Mitchell.

spørgsmål e

Reducér $h\,1$, hvor h er den PCF funktion, du definerede i spørgsmål d. Anvend venstre-reduktion (left-most reduction strategy) og vis alle skridtene. Du må dog gerne reducere *pred 1* til *0* i ét skridt og *mult (x,y)* til *x*y* i et skridt. For eksempel må man erstatte *mult (3,7)* med *21*.

Opgave 4

Denne opgave drejer sig om afgørlighed.

spørgsmål a

Bevis, at man *kan* afgøre, om fællesmængden mellem to regulære sprog (givet ved grammatikker) er tom eller ej.

spørgsmål b

Bevis, at man *ikke* kan afgøre, om fællesmængden mellem to sprog defineret ved urestringerede (unrestricted) grammatikker er tom eller ej.