INSTITUT FOR MATEMATIK OG DATALOGI

ODENSE UNIVERSITET

Skriftlig eksamen

D M 1 5

Fredag den 3. juni 1994 kl. 9.00 - 13.00

Opgavesættet består af 8 nummererede sider og indeholder 4 opgaver.
Opgaverne vægtes ens.
Alle skriftlige hjælpemidler samt brug af lommeregner er tilladt.

Opgave 1

Denne opgave handler om regulære sprog. Husk, at selv om man ikke har fået regnet alle spørgsmålene, må man gerne benytte resultaterne fra tidligere spørgsmål til at løse de senere.

spørgsmål a

Tegn tilstandsgrafen for en deterministisk endelig automat (DFA), der accepterer sproget

$$\{baaa\}\{a^*b^n \mid 3 \text{ går op i } n,\ n \geq 0\}.$$

spørgsmål b

Vis, at hvis L_1L_2 er et regulært sprog, og L_1 indeholder præcis én streng, så er L_2 også et regulært sprog.

spørgsmål c

Vis, at hvis L_1L_2 er et regulært sprog, og L_1 er endeligt, så er L_2 også et regulært sprog.

spørgsmål d

Vis, at hvis L_1L_2 er et regulært sprog, og L_1 også er et regulært sprog, så er L_2 ikke nødvendigvis et regulært sprog. Vis dette ved at angive et eksempel, hvor både L_1 og L_2 er defineret over alfabetet $\Sigma = \{a\}$.

Opgave 2

Denne opgave handler om kontekstfri sprog.

spørgsmål a

Lad $L_1 = \{xayb \;\mid\; x, y \in \{a,b\}^*,\ \mathit{length}(x) = \mathit{length}(y)\}$.
Vis, at L_1 er kontekstfrit.

spørgsmål b

Lad $L_2 = \{a^i b^j \;\mid\; j = i^2,\ i, j \geq 0\}$.
Vis, at L_2 ikke er kontekstfrit.

spørgsmål c

En kontekstfri grammatik er *lineær*, hvis og kun hvis ingen regler har mere end én non-terminal på højresiden. Et kontekstfrit sprog er *lineært*, hvis der eksisterer en lineær grammatik for det. Vis, at der eksisterer et lineært kontekstfrit sprog L, som ikke er regulært.

spørgsmål d

Find et kontekstfrit sprog, som ikke er lineært. Vis, at det er kontekstfrit og gør rede for, at det ikke er lineært (der kræves ikke et formelt bevis for, at sproget ikke er lineært).

Opgave 3

Denne opgave handler om Turing-maskiner, afgørlighed og MIRANDA. I de første to spørgsmål i denne opgave skal man implementere en simulator for deterministiske Turing-maskiner i MIRANDA. De sidste to spørgsmål handler om afgørlighed.

En Turing-maskine-simulator tager to argumenter: en liste af transitioner for en Turing-maskine og et input til Turing-maskinen. Simulatoren skal så simulere Turing-maskinen's opførsel på den givne input-streng. Hvis Turing-maskinen standser, så skal simulatoren returnere det, der står på båndet, når Turing-maskinen standser.

Antag, at Turing-maskinen's tilstande er af type num, og at båndsymbolerne er af type char. Antag, at 0 er starttilstanden. Så kan vi benytte følgende typedefinitioner i MIRANDA:

- dir ::= L | R
- transition == (num,char,num,char,dir)
- transitions == [transition]
- instdesc == (num,[char],[char],transitions)

En transition er et tupel, hvor man, givet en tilstand og et symbol under Turing-maskinen's læsehoved, kan se, hvad den næste tilstand skal være, hvilket symbol Turing-maskinen skal skrive, og hvilken retning Turing-maskinen skal bevæge sig. F.eks. angiver $(3,'a',7,'b',R)$, at Turing-maskinen i tilstand 3 med $'a'$ under læsehovedet skal skifte til tilstand 7, skrive et $'b'$ og gå til højre.

En øjebliksbeskrivelse (instantaneous description) består af den nuværende tilstand, en liste af symboler til venstre for læsehovedet, en liste af symbolerne under og til højre for læsehovedet samt listen af transitioner.

Vi benytter også følgende funktionsdefinitioner:

- first5 (a,b,c,d,e) = a
- second5 (a,b,c,d,e) = b
- third5 (a,b,c,d,e) = c
- fourth5 (a,b,c,d,e) = d
- fifth5 (a,b,c,d,e) = e
- third4 (a,b,c,d) = c

spørgsmål a

Skriv en funktion *move*, sådan at resultatet af *move* x er den næste øjebliksbeskrivelse efter x. Typen er

- move :: instdesc → instdesc

Hvis Turing-maskinen standser eller går til venstre fra den første plads på båndet, så skal tilstanden ikke ændres, men hele strengen på båndet skal stå i den første af de to lister med symbolerne på båndet, og den anden skal være tom.

spørgsmål b

Skriv en funktion *run*, sådan at resultatet af *run x trans* er hele den streng, der repræsenterer indholdet af båndet, idet Turing-maskinen standser eller går til venstre fra den første plads på båndet, når Turing-maskinen startes med x som input. Typen er

- run :: [char] $\rightarrow$ transitions $\rightarrow$ [char]

Husk, at der skal være et 'B' til venstre for x, når Turing-maskinen starter. Hvis Turing-maskinen ikke standser, så skal *run x trans* heller ikke standse.

spørgsmål c

Vis, at det generelt (dvs. for vilkårlige M og c) er uafgørligt, om en Turing-maskine M nogensinde skriver båndsymbolet c, hvis den startes på et tomt bånd.

spørgsmål d

Vis, at det er afgørligt, om en Turing-maskine M nogensinde skriver et symbol, som ikke er blank, hvis den startes på et tomt bånd. Man skal altså beskrive en algoritme, der tager en Turing-maskine M som input, og som så svarer på, om den nogensinde skriver et symbol, som ikke er blank, hvis den startes på et tomt bånd.

Opgave 4

Denne opgave har to versioner. Den første version er beregnet for de studerende, der tog DM15 i løbet af 93/94, og den anden version er beregnet for omgængere fra 92/93. Alle må besvare den version, de helst vil. **Man skal kun besvare én version, og det skal klart markeres, hvilken version, man besvarer.**

Version 1 (93/94)

Denne opgave handler om PCF, og hvordan man kan implementere et binært træ i PCF. Antag, at man har defineret følgende funktioner i PCF:

- En funktion *less* af type nat→nat→bool, sådan at *less* x y er *true*, hvis og kun hvis $x < y$. Ellers er resultatet *false*.
- En funktion *pairnums* af type nat→nat→nat, sådan at *pairnums* er en injektiv afbildning fra $\boldsymbol{N} \times \boldsymbol{N}$ til $\boldsymbol{N} - \{0\}$. (En mulig definition er *pairnums* n $m = (n+m)*(n+m+1)/2+m+1$.)
- En funktion $projnum_1$ af type nat→nat, sådan at $projnum_1$ *(pairnums n m)* $= n$.
- En funktion $projnum_2$ af type nat→nat, sådan at $projnum_2$ *(pairnums n m)* $= m$.

Nu kan man implementere et binært søgetræ med nøgler, der er naturlige tal. Et tomt træ repræsenteres af tallet 0. Et binært træ med nøglen x i roden, med venstre undertræ repræsenteret af l, og med højre undertræ repræsenteret af r, kan repræsenteres af *pairnums (pairnums l r) x*.

spørgsmål a

Hvordan repræsenterer man træet, der består af en rod med to sønner, hvor roden har nøglen 2, den venstre søn har nøglen 1, og den højre søn har nøglen 3?

spørgsmål b

Definer en funktion *find* af type nat → nat → bool. Det første argument er en repræsentation af et binært søgetræ, og det andet er en nøgle. Hvis nøglen findes i træet, så skal resultatet være *true*. Ellers skal det være *false*.

spørgsmål c

Definer en funktion *insert* af type nat → nat → nat. Det første argument er en repræsentation af et binært søgetræ, og det andet er en nøgle. Hvis nøglen findes i træet, så skal resultatet være førsteargumentet. Ellers skal det være repræsentationen af træet med den nye nøgle indsat, sådan at træet stædigvæk er et binært søgetræ.

Version 2 (92/93)

Denne opgave handler om lambda-kalkylen, og hvordan man kan implementere et binært træ i denne. Vi får brug for at operere med naturlige tal og sandhedsværdier. Da de ikke er med i den basale lambda-kalkyle, benyttes indkodning fra lærebogen. F.eks. er c_0 indkodningen af tallet 0. Al manipulation af tal foregår gennem følgende lambda-udtryk, som vi antager, vi allerede har:

- En funktion *less*, sådan at *less* c_x c_y er **T**, hvis og kun hvis $x < y$. Ellers er resultatet **F**.
- En funktion *equal*, sådan at *equal* c_x c_y er **T**, hvis og kun hvis $x = y$. Ellers er resultatet **F**.
- En funktion *pairnums*, sådan at *pairnums* er en injektiv afbildning fra $\boldsymbol{N} \times \boldsymbol{N}$ til $\boldsymbol{N} - \{0\}$. (En mulig definition er $\mathit{pairnums}\ c_n\ c_m = c_{(n+m)*(n+m+1)/2+m+1}$.)
- En funktion $\mathit{projnum}_1$, sådan at $\mathit{projnum}_1\ (\mathit{pairnums}\ c_n\ c_m) = c_n$.
- En funktion $\mathit{projnum}_2$, sådan at $\mathit{projnum}_2\ (\mathit{pairnums}\ c_n\ c_m) = c_m$.

Nu kan man implementere et binært søgetræ med nøgler, der er naturlige tal. Et tomt træ repræsenteres af c_0. Et binært træ med nøglen x i roden, med venstre undertræ repræsenteret af l, og med højre undertræ repræsenteret af r, kan repræsenteres af $\mathit{pairnums}\ (\mathit{pairnums}\ l\ r)\ c_x$.

Man må gerne benytte sig af udtryk, man allerede kender fra lærebogen. Dvs. **pair**, **h**, **t**, **T**, **F**, **if-then-else** og **Y**.

spørgsmål a

Hvordan repræsenterer man træet, der består af en rod med to sønner, hvor roden har nøglen 2, den venstre søn har nøglen 1, og den højre søn har nøglen 3?

spørgsmål b

Definer en funktion *find*. Det første argument er en repræsentation af et binært søgetræ, og den anden er en nøgle. Hvis nøglen findes i træet, så skal resultatet være **T**. Ellers skal det være **F**.

spørgsmål c

Definer en funktion *insert*. Det første argument er en repræsentation af et binært søgetræ, og det andet er en nøgle. Hvis nøglen findes i træet, så skal resultatet være førsteargumentet. Ellers skal det være repræsentationen af træet med den nye nøgle indsat, sådan at træet stædigvæk er et binært søgetræ.