Oblig 3 INF4130, Høst 2007 – dnebdal

2

**a.a**) Nei. Som definert i dette kurset er et formelt språk "De av alle mulige kombinasjoner av språkets tegn som en bestemt turingmaskin svarer 'Ja' for". Riktignok kan man lage en turingmaskin som godkjenner kun grammatisk riktig engelsk[1], og dermed bestemmer "grammatisk riktig engelsk" som et formelt språk over det latinske alfabetet, men det er som en bivirkning å regne.

**a.b**) Ja. Et desisjonsproblem kan sees på som "Gitt input x, finnes det et svar" for alle mulige x for et optimaliseringsproblem. Hvis det fantes en enkel måte å få svar på desisjonsproblemet, kunne det utnyttes til å få raske svar på optimaliseringsproblemet. Wikipedia sitt eksempel er illustrerende: Hvis vi kunne få svar i polynomisk tid på om et TSP-problem hadde et svar med størrelse <= x, ville det være mulig å først finne den minste x på polynomisk tid [2], og deretter endre problemet ved å øke kostnaden på en edge om gangen for å se om den var med i det beste svaret eller ikke. Da har man redusert kostnaden til (antall søk + antall mulige edges)*kostnaden av desisjonsalgoritmen.

Med andre ord: Hvis desisjonsproblemet for TSP tar polynomisk tid, så gjør optimaliseringsproblemet det også. Altså må de ligge i samme kompleksitetsklasse, med mindre desisjonsproblemet er enda verre. Det kan det ikke være, siden man kan få svaret ved å løse optimaliseringsproblemet.

**a.c**) Nei. For å være NP-komplett kreves det to ting: Problemet må være i NP (altså må svar kunne verifiseres på polynomisk tid), og det må henge sammen med de andre kjente NP-komplette problemene slik at en polynomisk-tid løsning for ett kan brukes på alle.

Det finnes problemer som er i NP og tar superpolynomisk tid som ikke er NP-komplette, slik som faktorisering av (store) heltall. Det finnes også problemer som tar superpolynomisk tid å løse som heller ikke er i NP, slik som de EXPTIME- og PSPACE-complete. Disse inkluderer bl.a. desisjonsproblemer rundt en del spill [3].

Det er derimot riktig at alle kjente måter å løse NP-komplette problemer er superpolynomiske, men det er ikke bevist at det er umulig å finne en polynomisk metode.

**b.a**) $L_1$ kan avgjøres av en ganske enkel turingmaskin. Gitt at input ikke inneholder Y, N, eller *b*, har en endelig lengde, er omgitt av *b*, og markøren starter helt til venstre, skal den under alltid gi et svar på maksimalt to tikk.

| **Tilstander** |
|---|
| Start |
| Funnet |
| Halt |

| **Tilstand** | **Symbol** | **Ny tilstand** | **Flytt** | **Skriv** |
|---|---|---|---|---|
| Start | Ikke 0 | Halt | - | N |
| Start | 0 | Funnet | R | *b* |
| Funnet | Ikke *b* | Halt | - | N |
| Funnet | *b* | Halt | - | Y |

Alfabetet er altså {*, 0, Y, N, *b*}, hvor * står for "hva som helst".

1 Noe som forsåvidt er forsøkt gjort, se f.eks. automatisk gramatikk-kontroll i tekstbehandlingsprogrammer

2 F.eks. ved å gjøre et binært søk mellom "den lengste avstanden mellom to noder * antall noder", og "den minste avstanden mellom to noder * antall noder", altså worst- og best-case.

3 For eksempel: "Gitt perfekt spill fra begge sider og brett av ubegrenset størrelse, kan hvit fremtvinge seier i sjakk?", som er EXPTIME-complete.

**b.b**)
*$L_2$ = {M | M gjenkjenner $L_1$ }*

$L_2$ består av "de av alle mulige maskiner som gjenkjenner $L_1$ ".
Hvis vi kaller alle mulige maskiner M*, finnes det et veldig enkelt argument for at $L_2$ er uavgjørbart:

Minst en maskin i M* er umulig å avgjøre om stopper[4]. Altså er det umulig å si for hver M i M* om den vil stoppe (uavhengig av input). Siden kriteriet for $L_2$ er "stopper med riktig output", må det nødvendigvis være uavgjørbart.

Alternativt:
Definisjonen av "avgjørlig" for formelle språk er "Det finnes en turing-maskin som svarer Y eller N for alle kandidatmedlemmer[5]". Hvis vi nå sier at det fantes en slik maskin, $M_2$:

Vi tenker oss en annen maskin $M_H$ som gjør følgende:
Leser inn en vilkårlig input I.
Konstruer koden for en turingmaskin $M_{TMP}$ som:
1) Sjekker om *sin* input er i $L_1$
2) Simulerer I
3) Skriver Y eller N avhengig av resultatet fra 1.

Kjører $M_2$ med $M_{TMP}$ som input.
Hvis nå $M_2$ sier at $M_{TMP}$ vil avgjøre $L_1$ , betyr det at I må stoppe.
Altså kan $M_H$ bruke $M_2$ til å løse halting-problemet, men halting-problemet er bevist uløselig.
Alle andre elementer i $M_H$ kan konstrueres, så $M_2$ må være umulig.

Altså finnes det ingen turingmaskin som svarer Y eller N på spørsmålet "kan denne maskinen gjenkjenne $L_1$" for alle mulige maskiner, og $L_2$ er uavgjørbart.

**b.c**)
*$L_3$ = {M | M gjenkjenner $L_2$}*

$L_3$ består altså av de maskinene som gjenkjenner de maskinene som gjenkjenner $L_1$.
Per argumentet over finnes det ingen slike, så $L_3$ er faktisk avgjørbart. Turingmaskinen som gjenkjenner $L_3$ svarer 'N' og stopper for alle inndata.

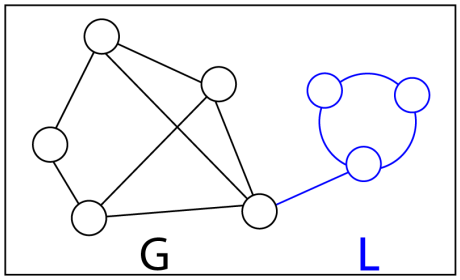
Såvidt jeg kan se vil språkene være vekselvis avgjørlige eller ikke for hvert nivå oppover:
$L_4$={M | M gjenkjenner $L_3$} er uavgjørlig fordi maskinen som gjenkjenner det må se om M stopper.
$L_5$={M | M gjenkjenner $L_4$} er avgjørlig fordi det ikke har noen medlemmer.
Og så videre.

4 Fordi det allerede er bevist at det ikke er mulig å algoritmisk avgjøre om enhver mulig maskin vil stoppe eller ikke
5 Altså for alle medlemmer i $\Sigma^*$ , hvis $\Sigma$ er alfabetet.

**c)**
Den første egenskapen ved NP-komplette problemer[6] er at løsningene kan verifiseres på polynomisk tid av en deterministisk turingmaskin. Det er her temmelig enkelt: Først sjekker man at hver node er i akkurat en av de to løkkene (løp gjennom alle noder i løkkene, marker hvilken løkke i en tabell; sjekk tabellen), deretter sjekker man at det faktisk finnes kanter fra hver node til den neste i løkken (løp gjennom nodene, slå opp i kant-tabellen). Ingen av disse behøver å ta mer enn lineær tid.

**Det bihamiltonske desisjonsproblemet (BHDP) er altså i NP.**

For å bevise at BHDP er NP-*komplett* kan vi redusere det vanlige hamiltonske desisjonsproblemet til BHDP. La oss tenke oss at vi har en turingmaskin $M_b$ som løser BHDP, og en vilkårlig graf G som vi vil sjekke om har en hamiltonsk løkke. Først lager vi G' ved å legge en kjent hamiltonsk løkke L til G, hvor akkurat en av de nye nodene har akkurat en kant som forbinder den med G. Deretter kjører vi $M_b$ på G'. Hvis den sier at G' er bi-hamiltonsk, må den ene løkken være L [7], og den andre må bestå av resten av G'. Og tilsvarende, den andre veien: Hvis den sier at G' ikke er bi-hamiltonsk, kan det ikke finnes en hamiltonsk løkke i G.



G′

Operasjonen med å legge til Ler enkel, forutsatt at grafen er lagret på et rasjonelt format. Man kan altså, i polynomisk tid, gjøre om et hamiltonian-desisjonsproblem til et BHDP.

**BHDP er altså NP-komplett.**

6 Og forsåvidt alle NP-problemer, siden det er definisjonen på NP

7 Ingen større løkke kan inneholde hele eller deler av L, siden den er nøyaktig en kant fra å være utilkoblet G, og en slik løkke nødvendigvis ville trenge minst to.