Laskennan vaativuus **Ratkaisuehdotuksia**
Harjoitus 5, 30.4.2007

1. (a) MIN-FORMULA -kieli määritellään

$$\text{MIN-FORMULA} = \{\varphi \mid \varphi \text{ kaava ja } \forall\theta : (\varphi \Leftrightarrow \theta) \implies |\varphi| \leq |\theta|\}.$$

Muodostetaan Turingin kone, joka toimii seuraavan algoritmin tavoin:

if φ ei ole kaava **then**
    Hylkää φ
end if
for $\theta \in \{x \mid x \in \Sigma^*, |x| < |\varphi|\}$ **do**
    if θ on kaava **then**
        $M \leftarrow$ kaavoissa φ ja θ esiintyvien muuttujien nimet
        for $s \leftarrow \{f \mid f : M^{|M|} \to \{\mathtt{0}, \mathtt{1}\}^{|M|}$ sijoitus$\}$ **do**
            if $s(\varphi) \neq s(\theta)$ **then**
                Jatka seuraavaan θ:an
            end if
        end for
        Hylkää φ, sillä $\varphi \Leftrightarrow \theta$ ja $|\theta| < |\varphi|$
    end if
end for
Hyväksy φ

Jokaisen testattavan merkkijonon θ pituus on polynominen. Merkkijonon tunnistaminen kaavaksi onnistuu varmasti polynomisessa tilassa. Kaikissa kaavoissa φ ja θ on korkeintaan $|\varphi|+|\theta|$ muuttujaa, joten jokainen sijoitus on kooltaan polynominen. Koska kaavan kuuluminen joukkoon MIN-FORMULA voidaan ratkaista polynomisessa tilassa, on MIN-FORMULA$\in$ **PSPACE**.

(b) Jos $\varphi \notin$ MIN-FORMULA, niin tosiaan on olemassa jokin kaava θ, jolla $\varphi \Leftrightarrow \theta$ ja $|\theta| < |\varphi|$. Epädeterminististinen Turingin kone voi arvata jonkin lyhyemmän kaavan θ', mutta se joutuisi myös osoittamaan, että $\varphi \Leftrightarrow \theta'$. Siksi todistus $\overline{\text{MIN-FORMULA}} \in$ **NP** eli MIN-FORMULA $\in$ **coNP** ei ole aivan helppo.

2. Olkoon MULT $= \{abc \mid a, b, c$ binäärilukuja, $a \cdot b = c\}$. Oletetaan että binääriluvuilla tarkoitetaan binäärikokonaislukuja. Muuten joudumme käyttämään uutta symbolia esittämään desimaalipilkkua tai koodaamaan sen muulla tavalla. Desimaalilukujen käyttö ei kuitenkaan vaikuttane ongelman vaativuuteen.

Muodostetaan Turingin kone M, joka testaa syötteen x kuuluvuuden kieleen MULT. Koneen syötenauhaa käytetään read-only -nauhana. Ensin kone tarkistaa, että syöte x koostuu pelkistä `0` ja `1` symboleista. Koska x:llä ei ole tarkempaa rakennetta, seuraavaksi koneen täytyy itse jakaa se osiin a, b ja c. Ylläpidetään työnauhalla binäärilukuja i_b ja i_c, jotka kertovat kertovat, mistä luvut b ja c alkavat syötteessä. Näiden muuttujien avulla käydään kaikki mahdolliset kolmikot (a, b, c) läpi.

Jokaisella kolmikolla (a, b, c) tulee testata, päteekö $a \cdot b = c$. Jos laskisimme $a \cdot b$, niin tuloksen tallentaminen vaatisi tilan $\mathcal{O}(|x|)$. Koska koko kertolaskun tulosta ei voi tallettaa muistiin, suoritetaan vertailu c:hen bitti kerrallaan laskennan aikana. Esimerkkikuvan avulla alla olevaa algoritmia on helpompi seurata.

$$
\begin{array}{cccccccc}
 & & & 1 & 0 & 1 & 1 & = a \\
\cdot & & & & 1 & 0 & 1 & = b \\
\hline
 & & & 1 & 0 & 1 & 1 & \\
 & & 0 & 0 & 0 & 0 & & \\
+ & 1 & 0 & 1 & 1 & & & \\
\hline
 & 1 & 1 & 0 & 1 & 1 & 1 & = c
\end{array}
$$

```
1: for (a, b, c) = x do
2:     m ← 0    // Muistinumero
3:     for j ← 0 ... |c| − 1 do
4:         for k ← 0 ... j do
5:             if a_{j−k} · b_k = 1 then    // a_j = 0, jos j ≥ |a|
6:                 m = m + 1
7:             end if
8:         end for
9:         if m_0 ≠ c_j then
10:            Testaa seuraava (a, b, c)
11:        end if
12:        m = m/2    // Bittisiirto
13:    end for
14:    if m = 0 then
15:        Hyväksy x    // Kaikki välitulokset vastasivat c:n bittejä
16:    end if
17: end for
18: Hylkää x    // Mikään kolmikko ei täyttänyt vaatimuksia
```

Muuttujat j ja k mahtuvat logaritmiseen tilaan. Muuttujan m maksimiarvo on suuruusluokkaa $|c|^2/2^{|c|} \leq |x|^2$. Tilaa se vie $\mathcal{O}(\log(|x|^2)) = \mathcal{O}(\log(|x|))$. Mahdolliset lisämuuttujat, joita ei tarvittu algoritmin esittämiseen mutta tarvitaan Turingin koneen toteuttamiseeen, vievät vakiotilan tai koko on $\mathcal{O}(\log(|x|))$. Näin on siis saatu MULT$\in$ **L**.

3. Tässä tehtävässä syklinä pidetään sellaista kulkua verkossa solmusta itseensä, missä mitään solmua ei käytetä montaa kertaa. Muuten on mahdollista esittää vastaesimerkkejä.

   Väite: Verkko on kaksijakoinen jos ja vain jos se ei sisällä parittoman mittaista sykliä. Todistus:

   "$\Rightarrow$"

   Olkoon verkko $G = (V, E)$ kaksijakoinen, missä $A, B \subseteq V$, joilla $A \cap B = \emptyset$, $A \cup B = V$ ja $\forall (v, w) \in E : (v \in A \wedge w \in B) \vee (v \in B \wedge w \in A)$. Vaikka verkko onkin suuntaamaton, esitetään sykli suunnattuna selkeyden vuoksi. Olkoon verkossa sykli, johon kuuluu solmu $a \in A$. Jos syklissä solmusta a lähtevä ja siihen tuleva särmä ovat sama, on syklissä vain kaksi solmua. Suuremmissa sykleissä siis lähtevä ja tuleva särmä ovat eri särmiä. Jos suureen sykliin kuuluu n kappaletta joukon A solmuja, niin siihen kuuluu myös jokaista lähtevää särmää kohden yksi joukon B solmu. Siis syklissä on parillinen määrä $2n$ solmua.

   "$\Leftarrow$"

Oletetaan, että verkossa $G = (V, E)$ ei ole parittoman solmumäärän mittaista sykliä. Jos verkossa ei ole lainkaan syklejä, on jako helppo toteuttaa. Oletetaan sitten että verkossa on parillisen mittainen sykli ja että verkko on yhtenäinen. Epäyhtenäisessä verkossa jokainen yhtenäinen osa voidaan jakaa erikseen ja yhdistää osatulokset.

Muodostetaan kaksijakoisen verkon solmujoukot A ja B siten, että jokaisessa syklissä joka toinen solmu kuuluu joukkon A ja joka toinen joukkoon B. Jako suoritetaan siten, että jos solmu on jossakin syklissä joukossa A, on se samassa joukossa myös toisessa syklissä. Solmut jotka eivät ole osa mitään sykliä sijoitetaan joukkoonsa sen mukaan, mihin joukkoon kyseisen solmun naapuri kuuluu.

Oletetaan nyt että kaikki kaikki solmut on numeroitu siten että parittomat kuuluvat joukkoon A ja parilliset joukkoon B. Jos parillisen mittaisessa syklissä on osa $v_1 \to v_2 \to v_3$, niin ei ole olemassa särmää $(v_1, v_3) \in E$, sillä tällöin olisi olemassa parittoman mittainen sykli. Edelleen jokaisessa syklin osassa $v_1 \to v_2 \to \cdots \to v_{2n+1}$ ei ole olemassa särmää $(v_1, v_{2n+1}) \in E$, sillä tällöinkin verkossa olisi parittoman mittainen sykli. Siis joukoissa A ja B ei ole sisäisiä särmiä ja verkko G on kaksijakoinen.

Näytetään lopuksi, että verkon kaksijakoisuus kuuluu **NL**:ään. Muodostetaan epädeterministinen Turingin kone M, joka valitsee syötteestään $G = (V, E)$ jonkin solmun $v \in V$ aloituskohdaksi. Kone ylläpitää laskuria k siitä, monessako solmussa se on vieraillut, ja toista muuttujaa w, mistä solmusta tähän solmuun on tultu. Alussa $k = 0$ ja $w = v$. Jokaisessa solmussa valitaan epädeterministisesti, mihin naapurisolmuun edetään. Jos valittu solmu on eri kuin mistä solmuun tultiin, edetään siihen ja päivitetään k ja w. Jos sopivaa naapurisolmua ei ole olemassa, hylätään syöte. Jos $k > |V|$, niin hylätään. Jos saavuttiin aloitussolmuun v, tutkitaan onko laskuri k parillinen vai pariton. Jos k on pariton, hyväksytään syöte. Muuten syöte hylätään. Nyt kone M hyväksyy vain sellaiset verkot, joissa on parittoman solmun mittainen sykli. Aikaisemmin todistetun mukaan kone M hyväksyy vain ei-kaksijakoisia verkkoja. Koneen tilantarpeesta nähdään että $L(M) \in \mathbf{NL}$. Koska $\mathbf{NL} = \mathbf{coNL}$, niin $\overline{L(M)} \in \mathbf{NL}$.

4. Väite: $\mathbf{BPP} \subseteq \mathbf{PSPACE}$.
   Todistus: Olkoon $A \in \mathbf{BPP}$ kieli ja M sen virhetodennäköisyydellä korkeintaan $\frac{1}{3}$ tunnistava probabilistinen Turingin kone. Muodostetaan TM N, joka simuloi koneen M syötteellä x eri suoritukset syvyyssuuntaisella läpikäynnillä. Jokaisen suorituksen päätteeksi kirjataan, hyväksyttiinkö vain hylättiinkö syöte. Kun kaikki suoritukset on käyty läpi, vertaillaan tuloksia. Jos hyväksyvien haarojen osuus on vähintään $\frac{2}{3}$ kaikista tuloksista, kone N hyväksyy syötteen. Jos hylkäävien haarojen osuus on vähintään $\frac{2}{3}$ kaikista tuloksista, kone N hylkää syötteen. Muita vaihtoehtoja ei ole johtuen koneen M ominaisuuksista.

   Koneen N läpikäymän suorituspuun syvyys on polynominen suhteessa syötteeseen ja koneen M konfiguraatio jokaisessa haarassa on polynomisen kokoinen, joten koneen N muistissa pitämän polun koko on polynominen. Eri suorituksia voi olla eksponentiaalinen määrä, mutta suoritusten tulokset kirjataan binäärilukuina, jolloin luvut saadaan kooltaan polynomisiksi. Siten koneen N tilavaativuus on polynominen ja $L(N) \in \mathbf{PSPACE}$. Koska N onnistui simuloimaan kielen A tunnistavaa konetta M, on oltava $A \in \mathbf{PSPACE}$. Yleisemmin $\mathbf{BPP} \subseteq \mathbf{PSPACE}$.

5. Olkoon $\mathbf{NP} \subseteq \mathbf{BPP}$ (Huomaa että tätä ei oikeasti ole onnistuttu todistamaan).
   Väite: $\mathbf{NP} = \mathbf{RP}$.
   Todistus: Luennoista tiedetään ja suoraan määritelmistä voidaan havaita, että $\mathbf{RP} \subseteq \mathbf{NP}$, joten todistetaan vain $\mathbf{NP} \subseteq \mathbf{RP}$. Koska $\mathbf{NP} \subseteq \mathbf{BPP}$, niin SAT$\in \mathbf{BPP}$. On siis olemassa probabilistinen kone M, joka ratkaisee $\varphi(x_1, \ldots, x_n) \overset{?}{\in}$ SAT virhetodennäköisyydellä $\frac{1}{3}$. Vahvistuslemman nojalla virhetodennäköisyys voidaan pienentää todennäköisyyteen $\frac{1}{n^2}$.

   Koska luokka **RP** ei salli kieleen kuulumattomien syötteiden hyväksymistä millään todennäköisyydellä, on ainoa mahdollisuutemme käyttää käytössämme olevaa probabilistista konetta M ratkaisemaan jokin sijoitus kaavalle $\varphi(x_1, \ldots, x_n)$. Muodostetaan seuraavanlainen kone N. Aloitetaan testaamalla $M(\varphi(x_1, \ldots, x_n))$. Jos kone hylkää, N hylkää. Muuten kokeillaan $M(\varphi(0, x_2, \ldots, x_n))$. Jos tämä hylättiin muodostetaan sijoitus $s(1/x_1)$. Hyväksyttäessä asetetaan sijoitukseksi $s(0/x_1)$. Jatketaan sijoitusten kokeilua kunnes meillä on sijoitus $s(a_1/x_1, \ldots, a_n/x_n)$ kaikille muuttujille. Kone N nyt joko hylkää syötteen tai palauttaa sijoituksen s. Todennäköisyys että N palauttaa virheellisen sijoituksen eli M vastasi vähintään kerran väärin on korkeintaan $\frac{n+1}{n^2} \leq \frac{1}{2}$.

   SAT voidaan palauttaa luokan **RP** ongelmaksi käyttäen konetta N. Ajetaan N syötteelle φ. Jos N hylkäsi, hylätään. Jos N palautti sijoituksen s, tarkistetaan päteekö $s(\varphi)$. Jos sijoitus ei päde, hylätään syöte. Muuten syöte hyväksytään. Näin muodostettu kone ratkaisee SAT-ongelman. Se ei koskaan hyväksy sellaista syötettä, joka ei kuulu SAT:iin. Jos $\varphi \in$ SAT, niin todennäköisyys vastata oikein on yli $\frac{1}{2}$. Siis SAT$\in \mathbf{RP}$. Koska SAT on **NP**-täydellinen ongelma, on $\mathbf{NP} \subseteq \mathbf{RP}$. Näin siis $\mathbf{NP} = \mathbf{RP}$.