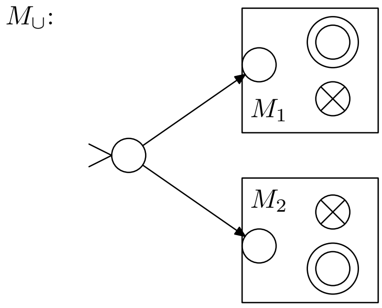
T-79.1001/2 **Syksy 2007**
Tietojenkäsittelyteorian perusteet (T)
Harjoitus 10
Demonstraatiotehtävien ratkaisut

4. **Tehtävä**: Osoita, että rekursiivisesti numeroituvien kielten luokka on suljettu yhdisteiden ja leikkausten suhteen. Miksi luokkaa ei voida osoittaa suljetuksi komplementtien suhteen samaan tapaan kuin rekursiivisten kielten luokkaa, yksinkertaisesti tunnistajakoneiden hyväksyvät ja hylkäävät lopputilat vaihtamalla?

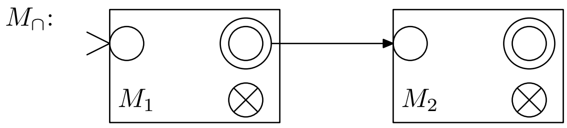
   Vastaus: Olkoon L_1 ja L_2 rekursiivisesti numeroituvia kieliä ($L_1, L_2 \in RE$). Tällöin on olemassa Turingin koneet M_1 ja M_2, jotka tunnistavat ne ($L(M_1) = L_1$ ja $L(M_2) = L_2$). Muodostetaan nyt koneet $M_\cup$ ja $M_\cap$, jotka tunnistavat kielten unionin $L_1 \cup L_2$ ja leikkauksen $L_1 \cap L_2$.

   Unioni: Muodostetaan kone $M_\cup$ yhdistämällä koneet M_1 ja M_2 seuraavaan tapaan:



   Kone on epädeterministinen, ja se aluksi valitsee epädeterministisesti suorittaako se laskennan koneen M_1 vai M_2 mukaisesti. Koska epädeterministiset ja deterministiset Turingin koneet ovat yhtä ilmaisuvoimaisia ja $M_\cup$ tunnistaa kielen $L_1 \cup L_2$, niin $L_1 \cup L_2 \in RE$.

   Leikkaus: Muodostetaan kone $M_\cap$ yhdistämällä koneet M_1 ja M_2 seuraavaan tapaan:



   Syötteellä x kone $M_\cap$ suorittaa ensin laskennan $M_1(x)$. Mikäli tämä pysähtyy, suoritetaan laskenta $M_2(x)$. Mikäli molemmat laskennat pysähtyvät, $x \in L_1 \cap L_2$ ja $M_\cap$ pysäytetään myös. Näin ollen $L_1 \cap L_2 \in RE$. (Tarkkaan ottaen $M_\cap$-koneen täytyy tallettaa syöte x ennen kuin M_1 käynnistetään, jotta se voitaisiin antaa myös M_2:lle).

   Komplementti:
   Kun kieli L kuuluu luokkaan $RE - R$ (= rekursiivisesti numeroituva, muttei rekursiivinen), voi kielen tunnistava Turingin kone M hylätä sanan x kahdella eri tapaa:

   (a) M pysähtyy tilaan q_{rej}; tai
   (b) M ei pysähdy koskaan.

   Muodostetaan nyt kone $\overline{M}$, jossa hyväksyvä ja hylkäävä tila on vaihdettu keskenään. Nyt myös $\overline{M}$ hylkää sanan x, mikäli laskenta ei pysähdy, joten $x \notin L(M) \cup L(\overline{M})$, joten $\overline{M}$:n tunnistama kieli ei voi olla $\overline{L(M)}$.
   Itseasiassa voidaan osoittaa (tehtävät 1d ja 2b), että jos $L \in RE - R$, niin $\overline{L} \notin RE$.

5. **Tehtävä**: Osoita, että Turingin koneilla, joilla on hyväksyvän ja hylkäävän lopputilan lisäksi vain *yksi* sisäinen tila, voidaan tunnistaa täsmälleen samat kielet kuin standardimallisillakin koneilla. Saat tarvittaessa olettaa, että koneet ovat moninauhaisia ja voivat pitää nauhapäänsä myös paikallaan (siirtosuunta $S \sim$ 'stationary'). Montako sisäistä tilaa tarvittaisiin simuloinnin toteuttamiseen yksinauhaisilla koneilla?

   Vastaus:

   Olkoon M Turingin kone, jossa on n tilaa. Tällaista konetta simuloida yksitilaisella kaksinauhaisella Turingin koneella M', jonka nauha-aakkostoon lisätään merkki q_i jokaista M:n tilaa q_i kohden. Koneen M' ensimmäistä nauhaa käytetään varsinaiseen laskentaan, ja toiselle nauhalle tallennetaan tila, jossa M olisi. Esimerkiksi koneen M tilanne $(q_2, a\underline{b}ba)$ esitettäisiin nauhalla seuraavasti:

$$\left\{\begin{array}{l}\begin{array}{|c|c|c|}\hline > & \underline{q_2} & < \\ \hline\end{array}\\ \begin{array}{|c|c|c|c|c|c|}\hline > & \mathrm{a} & \underline{\mathrm{b}} & \mathrm{b} & \mathrm{a} & < \\ \hline\end{array}\end{array}\right.$$

   Mikäli koneella M on siirtymä $\delta(q_i, a) = (q_j, b, \Delta)$, niin koneelle M' asetetaan siirtymä $\delta(q_0', (a, q_i)) = (q_0', (b, q_j), (\Delta, S))$, missä q_0' on M':n ainoa tila. Toisin sanoen, kone M' suorittaa täsmälleen samat laskenta-askeleet kuin M:kin, mutta laskennan tilaa ei talletetakaan koneen tiloihin vaan nauhalle. Koneen toinen lukupää pysyy koko ajan paikallaan, sillä muuten tila-informaatio hukkuisi.

   Formaalisti konstruktion voi esittää seuraavasti: Olkoon $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}})$ standardimallinen Turingin kone, missä $Q = \{q_0, \ldots, q_n\}$. Muodostetaan 2-nauhainen Turingin kone $M' = (\{q_0'\}, \Sigma, \Gamma \cup \{q_0, \ldots, q_n\}, \delta', q_0', q_{\mathrm{acc}}', q_{\mathrm{rej}}')$, missä:

$$\begin{aligned}\delta' =& \{(q_0', (a, q_i), q_0', (b, q_j), (\Delta, S)) \mid \delta(q_i, a) = (b, q_j, \Delta), q_j \notin \{q_{\mathrm{acc}}, q_{\mathrm{rej}}\}\} \\ &\cup \{(q_0', (a, q_i), q', (b, q)) \mid \delta(q_i, a) = (b, q, (\Delta, S)), q \in \{q_{\mathrm{acc}}, q_{\mathrm{rej}}\}\}\end{aligned}$$

   Äkkiseltään voisi ajatella, että saman tempun voisi tehdä myös 2-uraisella koneella, jolloin ei tarvittaisi toista nauhaa. Valitettavasti tämä onnistuu ainoastaan silloin, kun kone M ei koskaan liikuta lukupäätänsä. Heti kun M siirtää lukupäänsä joko oikealle tai vasemmalle, 2-uraisena toteutettu M' hukkaa tilansa. Mikäli M on sopivaa muotoa, voidaan osa sen tiloista poistaa käymällä aina kirjoittamassa syötteen loppuun kulloinenkin tila, mutta pahimmassa tapauksessa tämänkin ajatuksen toteutukseen tarvitaan vähintään n tilaa, jolloin kone M' ei ole konetta M pienempi.

6. **Tehtävä**:

   (a) Osoita, että mikä tahansa päätösongelma, jolla on vain äärellisen monta mahdollista syötettä, on ratkeava.

   (b) Osoita, että päätösongelma "esiintyykö luvun π desimaalikehitelmässä jossain kohden sata peräkkäistä nollaa" on ratkeava. Mitä tulos kertoo (i) π:n desimaalikehitelmästä, (ii) ratkeavuuden ja ratkeamattomuuden käsitteistä?

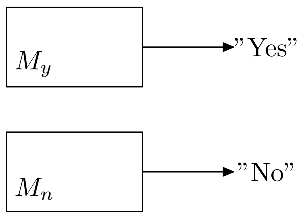
   Vastaus:

   (a) Jos päätösongelmalla on vain äärellisen monta mahdollista syötettä, on mahdollista luoda Turingin kone, jonka tiloihin on suoraan koodattu kaikki mahdolliset syötevaihtoehdot, ja jokaisen syötteen kuuluminen/kuulumattomuus kieleen. Siispä tällaiselle kielelle on aina mahdollista luoda totaalinen Turing-kone ja kieli on ratkeava.

   (b) Päätösongelmalla: "Esiintyykö luvun π desimaalikehitelmässä jossain kohden sata peräkkäistä nollaa?" on vain yksi syöte, π, joten a-kohdan perusteella ongelma on ratkeava.

   Ongelmaa hankaloittaa se, että luonnollinen tapa ratkaista se olisi laskea desimaalikehitelmää luku kerrallaan, ja tarkistaa löytyykö sataa peräkkäistä nollaa. Tämä laskenta ei kuitenkaan välttämättä pysähdy! Mikäli vaadittua määrää nollia ei löydy, jatkaa se ikuisesti π:n desimaalien tarkastamista. Niinpä tällä menetelmällä voidaan päätösongelma ratkaista vain osittain.

Ongelmalla on kuitenkin vain yksi vastaus: π:ssä joko on sata peräkkäistä nollaa tai sitten siinä ei ole niitä. Näin ollen toinen seuraavista Turingin koneista ratkaisee ongelman:



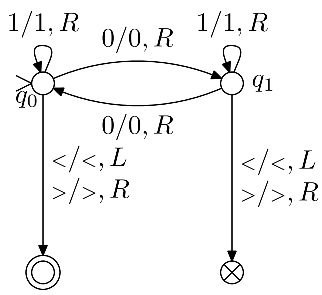
Kone M_y hyväksyy syötteen ja kone M_n hylkää sen. Valitettavasti ei voida sanoa, kumpi koneista on oikea (tosin M_y vaikuttaa todennäköisemmältä).

Kuten yllä nähdään, ratkeavuuden käsite on hyvin heikko. Ongelma voi olla teoreettisesti ratkeava, vaikka sitä ei käytännössä pystyttäisikään ratkaisemaan.

Liite: Turingin koneiden binäärikoodaus

Turingin kone voidaan esittää monella eri tapaa. Ihmislukijalle havainnollisin esitystapa on tilakaavio, kun taas tietokone käsittelee formaalin määritelmän joukko-opillista notaatiota sujuvasti. Binäärikoodauksen tarkoituksena on esittää kone muodossa, jota voidaan käsitellä toisilla Turingin koneilla.

Muodostetaan koodaus allaolevalle koneelle M:



Koneen eri osien koodaukset ovat seuraavat:

Tila	Koodi	Merkki	Koodi	Suunta	Koodi
q_0	0	0	0	L	0
q_1	00	1	00	R	00
q_{acc}	000	$<$	000		
q_{rej}	0000	$>$	0000		

Tilaan q_0 liittyvien siirtymien koodaukset ovat:

Siirtymä	Koodaus
$(q_0, 0, q_1, 0, R)$	01010010100
$(q_0, 1, q_0, 1, R)$	010010100100
$(q_0, <, q_{\mathrm{acc}}, <, L)$	010001000100010
$(q_0, >, q_{\mathrm{acc}}, >, R)$	010000100010000100

Tilaan q_1 liittyvät siirtymät ovat:

Siirtymä	Koodaus
$(q_1, 0, q_0, 0, R)$	00101010100
$(q_1, 1, q_0, 1, R)$	00100100100100
$(q_1, <, q_{\mathrm{rej}}, <, L)$	00100010000100010
$(q_1, >, q_{\mathrm{rej}}, >, R)$	0010000010000010000100

Koneen M koodaus saadaan asettamalla kaikkien siirtymien koodaukset peräkkäin siten, että siirtymien välillä on aina kaksi 1-merkkiä.

$$111\underline{01010010100}11\underline{010010100100}11\underline{010001000100010}11\underline{010000100010000100}11$$
$$\underline{00101010100}11\underline{00100100100100}11\underline{00100010000100010}11\underline{0010000010000010000100}111$$