Luku 6

Ratkeavuus ja ratkeamattomuus

eli Rekursiiviset, Rekursiivisesti lueteltavat ja kieliluokkien ulkopuolelle jäävät kielet

Palautetaan vielä mieliin seuraavat käsitteet:

- Turingin kone $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}})$ on *totaalinen*, jos se pysähtyy kaikilla syötteillä
- Formaali kieli A on *rekursiivisesti lueteltava*, jos se voidaan tunnistaa jollakin Turingin koneella, ja
- *rekursiivinen*, jos se voidaan tunnistaa jollakin totaalisella Turingin koneella.
- Päätösongelma π on *ratkeava*, jos sitä vastaava formaali kieli A_π on rekursiivinen, ja
- päätösongelma on *osittain ratkeava*, jos A_π on rekursiivisesti lueteltava.
- Ongelma, joka ei ole ratkeava, on *ratkeamaton*. (*Huom.*: ratkeamaton ongelma voi siis olla osittain ratkeava.)

Analogia:

Oletetaan että universumi on ääretön ja siinä on äärettömän monta tähteä. Kuvitellaan, että sinulla olisi ääretön tietokanta, joka sisältää tiedot kaikkien tähtien koordinaateista. Olet kiinnostunut tähtien naapureista, jotka olet määritellyt seuraavasti:

Tähdet A ja B ovat naapureita, jos A:n ja B:n etäisyys on alle R valovuotta eli $\Delta(A, B) \leq R$.

Tarkastellaan seuraavia ongelmia:

i) Onko Koirantähti Kissantähden naapuri? Kissantähden kordinaatit ovat (x, y, z) ja Koirantähden koordinaatit (r, s, t). Nyt lasketaan vain etäisyys $\sqrt{(x-r)^2 + (y-s)^2 + (z-t)^2}$ eli ongelma on selvästi ratkeava. (Vastaava "kieli" olisi $L_{Kissa} = \{x | x$ on tähden koodi ja x on Kissantähden naapuri$\}$. Ongelma palautuu siis "kielentunnistusongelmaksi" Koirantähti $\in L_{Kissa}$?)

ii) Onko Kanintähdellä yhtään naapuria? Olk. Kanintähden koordinaatit (x, y, z). Nyt käytävä pahimmassa tapauksessa koko tietokanta läpi. Kuitenkin jos Kanintähdellä on naapuri, esim. tietokannan i:s tähti, löytyy se siis äärellisessä ajassa (i:nnellä aikaskeleella). Sen sijaan emme saa koskaan varmuutta, mikäli Kanintähdellä ei ole naapuria. Ongelma ratkeaa siis vain "Kyllä"-tapauksessa, mutta ei "Ei"-tapauksessa ja on siis osittain ratkeava. (Haluamme siis ratkaista "kieltä" $L_{Kani} = \{x | x$ on tähden koodi ja x on Kanintähden naapuri$\}$ koskevan ongelman $L_{Kani} \neq \emptyset$?)

iii) Onko Kanintähti yksinäinen tähti? Tämän ongelman "Kyllä"-vastaus on sama kuin edellisen ongelman "Ei"-vastaus eli ratkeamaton. Meidän täytyisi tarkistaa kaikki tietokannan tähdet, joita on äärettömän monta, eli laskenta ei pääty koskaan. (Haluamme siis tietää, onko $L_{Kani} = \emptyset$?)

Täysin ratkeamattomia ongelmia ei ole enää luokiteltu laskennallisen vaikeutensa perusteella, vaikka voisi ajatella, että seuraava ongelma olisi vieläkin vaikeampi: Luettele kaikki universumin yksinäiset tähdet. ("Kielenä" $L_{yksin} = \{x | x$ on tähden koodi ja $L_x = \emptyset\}$, missä L_x sisältää x:n kaikki naapurit.)

6.1 Ratkeavat ongelmat eli rekursiivinen kieli

Jos ratkeavasta ongelmasta muodostetaan komplementtiongelma tai kahdesta ratkeavasta ongelmasta muodostetaan yhdiste tai leikkaus, on tulosongelma edelleen ratkeava. Ts. rekursiivisille kielille pätevät seuraavat sulkeumaominaisuudet:

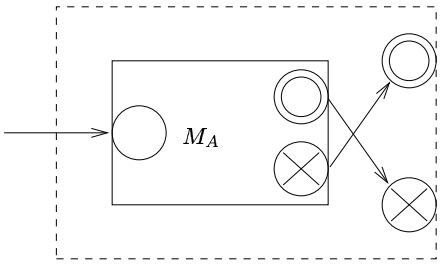
Lause: Olkoot $A, B \subseteq \Sigma^*$ rekursiivisia. Tällöin myös

i) $\bar{A} = \Sigma^* - A$,
ii) $A \cup B$ ja
iii) $A \cap B$

ovat rekursiivisia.

Todistus.

(i)Olkoon M_A totaalinen Turingin kone, joka tunnistaa kielen A (ts. $L(M_A) = A$). Kielen $\bar{A}$ tunnistava totaalinen Turingin kone saadaan vaihtamalla M_A:n hyväksyvä ja hylkäävä lopputila keskenään.



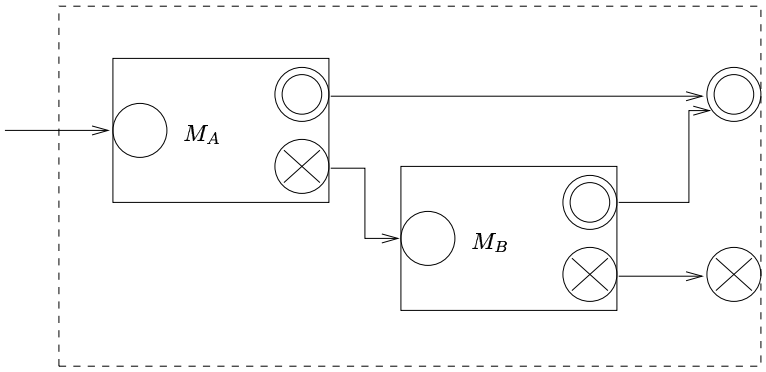
Kuva 6.1: Rekursiivisen kielen komplementin tunnistaminen Turingin koneella.

(ii)Olkoot M_A ja M_B totaaliset Turingin koneet, jotka tunnistavat kielet A ja B (ts. $L(M_A) = A$, $L(M_B) = B$). Kielen $A \cup B$ tunnistava totaalinen Turingin kone M saadaan yhdistämällä M_A ja M_B toimimaan peräkkäin: jos M_A hyväksyy syötteen, myös M hyväksyy; jos M_A päätyy hylkäämiseen, M kokeillaan vielä M_B:tä.

Huom! Tätä varten M_A:n jätettävä pysähtyessää nauhalle alkuperäinen syötejono koskemattomana (ja siivottava omat jälkensä).

(iii) $A \cap B$:lle saadaan totaalinen tunnistajakone de Morganin avulla: $A \cap B = \overline{\bar{A} \cup \bar{B}}$. Ts. muodostetaan ensin komplementtikoneet $M_{\overline{A}}$ ja $M_{\overline{B}}$, yhdistetään ne koneeksi $M_{\overline{A} \cup \overline{B}}$ ja muodostetaan vielä tämän komplementtikone $M_{\overline{\overline{A} \cup \overline{B}}}$. □

(Tehtävä: Piirrä kone $M_{\overline{\overline{A} \cup \overline{B}}}$!)



Kuva 6.2: Kahden rekursiivisen kielen yhdisteen tunnistaminen Turingin koneella.

6.2 Osittain ratkeavat ongelmat eli rekursiivisesti numeroituva kieli

Jos meillä on Turingin koneet (jotka pysähtyvät "Kyllä"-tapauksessa, mutta eivät välttämättä "Ei"-tapauksessa) M_A ja M_B, niin voimme muodostaa myös niiden yhdiste- ja leikkauskoneet $M_{A\cup B}$ ja $M_{A\cap B}$. Ts. rekursiivisesti numeroituville kielille pätevät seuraavat sulkeumaominaisuudet:

Lause: Olkoot $A, B \subseteq \Sigma^*$ rekursiivisesti lueteltavia. Tällöin myös $A \cup B$ ja $A \cap B$ ovat rekursiivisesti lueteltavia.

Todistus: H.T. $\square$

Huom! Rekursiivisesti numeroituville kielille ei päde, että komplementtikieli olisi (välttämättä) rekursiivisesti lueteltava! Ts. jos meillä on Turingin kone M_A, joka tunnistaa kielen A ja pysähtyy "Kyllä"-tapauksessa, mutta ei välttämättä "Ei"-tapauksessa, niin emme voi muodostaa konetta $M_{\overline{A}}$, joka pysähtyisi "Kyllä"-tapauksessa kaikilla syötteillä. Miksi?

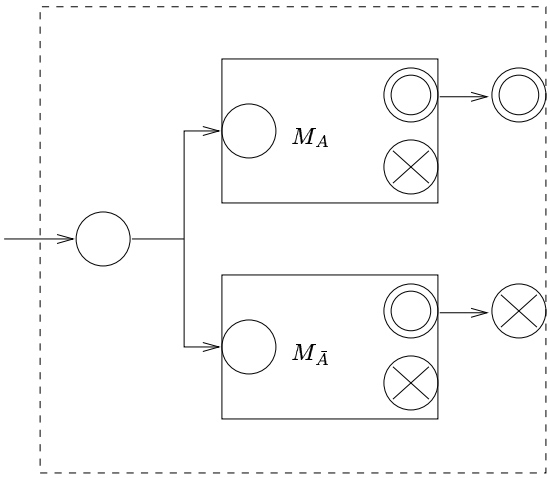
Seuraava tärkeä tulos antaa meille keinon tunnistaa, milloin ongelma on ratkeava:

Lause: Kieli $A \subseteq \Sigma^*$ on rekursiivinen, jos ja vain jos kielet A ja $\bar{A}$ ovat rekursiivisesti lueteltavia.

Todistus. "$\Rightarrow$": seuraa lauseesta 6.1(i).

"$\Leftarrow$": Olkoot M_A ja $M_{\bar{A}}$ Turingin koneet kielten A ja $\bar{A}$ tunnistamiseen. Kaikilla $x \in \Sigma^*$ *joko* M_A *tai* $M_{\bar{A}}$ *pysähtyy* ja hyväksyy x:n. Muodostetaan M_A ja $M_{\bar{A}}$ "rinnakkain" yhdistämällä totaalinen kaksinauhainen tunnistajakone M: M simuloi ykkösnauhallaan konetta M_A ja kakkosnauhallaan konetta $M_{\bar{A}}$. Jos ykkössimulaatio

pysähtyy hyväksyvään lopputilaan, M hyväksyy syötteen; jos taas kakkossimulaatio hyväksyy, M hylkää syötteen. □



Kuva 6.3: Totaalisen Turingin koneen muodostaminen kahdesta rinnakkain toimivasta koneesta.

Huom! Voidaan toteuttaa 2-nauhaisena koneena, joka rinnakkaisesti simuloi toisella nauhallaan M_A:n ja toisella M_B:n laskentaa. Jos M_A pysähtyy hyväksyvään tilaan, niin M hyväksyy syötteen, jos taas M_B hyväksyy syötteen, niin M hylkää sen.

$\Rightarrow$ *Seuraus:* Jos $A \subseteq \Sigma^*$ on rekursiivisesti lueteltava kieli, joka ei ole rekursiivinen, niin kieli $\bar{A}$ ei ole rekursiivisesti lueteltava (ts. ongelma on täysin ratkeamaton)!.

6.3 Ratkeamattomuus

Cantorin diagonaaliargumentin perusteella tiedämme, että on olemassa ratkeamattomia ongelmia. Turingin koneiden avulla päättely olisi seuraava:

1) Turingin koneet ovat äärellisiä
2) Siispä voimme luetella kaikki Turingin koneet (ts. Turingin koneiden joukko on numeroituvasti ääretön)
3) Kaikkien ongelmian joukko ei voi olla numeroituva (Diag. arg.: kaikkien päätösongelmien ts. kielentunnistusongelmien joukko on ylinumeroituva.)

$\Rightarrow$ On olemassa onmgelmia, joille ei ole olemassa ratkaisevaa Turingin konetta (eikä siis mitään muutakaan laskennallista ratkaisua Churchin-Turingin teesin mukaan).

- Haluamme konkreettisen esimerkin täysin ratkeamattomasta ongelmasta
- Rajoittamattomille kielille ei ole olemassa omaa Pumppauslemmaa, mutta voimme silti käyttää diagonalisaatioargumenttia (Valehtelijan paradoksia) sellaisen konstruoimiseksi.
- Idea:
  1. Tehdään vastaväite, että Turingin koneet kykenevät ratkaisemaan mitä tahansa – siis myös itseään koskevia kysymyksiä.
  2. Muodostetaan Turingin kone, joka pysähtyy, jos ei pysähdy, ja ei pysähdy, jos pysähtyy.
  3. Saadaan RR. eli oletus ei päde.
- Ensin täytyy määritellä Turingin kone, joka kykenee tutkimaan Turingin koneiden ominaisuuksia (ts. tulkitsee TM:ien "koodeja" ja "ajaa" niitä). Tällaista Turingin konetta kutsutaan *universaalikoneeksi*.

6.4 Universaalikone ja universaalikieli

- Universaalikone
  - saa syötteenään koneen M koodin ja tämän syötteen w
  - pysähtyy vain jos M pysähtyy syötteellä w
  - tulostaa saman, mitä M tulostaa syötteellä w
- Keksittävä tapa koodata Turingin koneita siten, että
  - voimme esittää minkä tahansa TM:n äärellisen aakkoston koodina
  - universaalikone kykenee purkamaan koodin
- riittää koodata TM:n siirtymäfunktio – se kuvaa koneen toiminnan (ei edes haittaa, vaikka tilat nimettäisiin uudelleen)!

6.4.1 Turingin koneen koodaus kokonaislukuna (bittijonona)

Tarkastellaan standardimallista Turingin konetta $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}})$, jonka syöteaakkosto on $\Sigma = \{0, 1\}$.

Olk. $Q = \{q_0, q_1, \ldots, q_n\}$, missä $q_{\text{yes}} = q_{n-1}$ ja $q_{\text{no}} = q_n$.

$\Gamma \cup \{>, <\} = \{a_0, a_1, \ldots, a_m\}$, missä $a_0 = 0$, $a_1 = 1$, $a_2 = >$ ja $a_3 = <$.

Merk. $\Delta_0 = L$ ja $\Delta_1 = R$.

Määritellään kaikille näille omat koodit, jotta päästään koodaamaan siirtymäfunktiota.

Kooditaulu:

Tilat	q_0	0
	q_1	00
	q_2	000
	.	.
	.	.
	.	.
	$q_{n-1} = q_{yes}$	0^n
	$q_n = q_{no}$	0^{n+1}
Merkit $\Gamma \cup \{>, <\}$	0	0
	1	00
	$>$	000
	$<$	0000
	a_2	$00000 = 0^5$
	a_3	0^6
	.	.
	.	.
	.	.
	a_m	0^{m+1}
Suunnat	L	0
	R	00

Jokainen siirtymäfunktion δ arvoista voidaan nyt koodata seuraavasti:

Säännön $\delta(q_i, a_j) = (q_r, a_s, \Delta_t)$ koodi on

$$c_{ij} = 0^{i+1}10^{j+1}10^{r+1}10^{s+1}10^{t+1}.$$

ts. $c_{ij} = 1(q_i$:n koodi$)1(a_j$:n koodi$)1(q_r$:n koodi$)1(a_s$:n koodi$)1(\Delta_t$:n koodi$)$

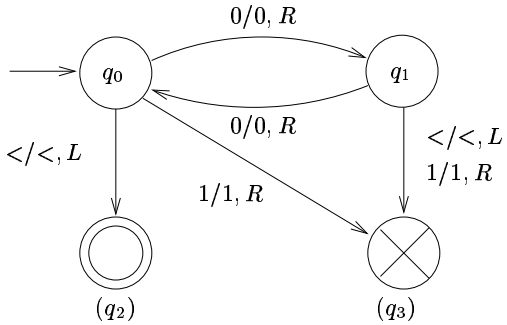
Koko koneen M koodi saadaan yhdistämällä kaikkien siirtymäfunktion arvojen koodit yhdeksi binäärijonoksi, käyttäen erotinmerkkeinä 11:tä ja lisäksi kodin alkuun ja lopuun jonot 111.

Koneen M koodi on siis

$$\begin{aligned} c_M &= 111c_{00}11c_{01}11\ldots11c_{0m}11c_{10}11\ldots11c_{1m}11 \\ &\quad \ldots11c_{n-2,0}11\ldots11c_{n-2,m}111. \end{aligned}$$

Esim kielen $\{0^{2k} \mid k \geq 0\}$ tunistava kone: tunnistavan koneen koodi olisi:

$$c_M = 111\underbrace{01010010100}_{\delta(q_0,0)=(q_1,0,R)}11\underbrace{01001000010010011}_{\delta(q_0,1)=(q_3,1,R)}11\ldots$$



Kuva 6.4: Kielen $\{0^{2k} \mid k \geq 0\}$ tunnistava Turingin kone.

- Jokaista Turingin konetta M vastaa koodi c_M
- Jokaiseen binäärijonoon c voidaan liittää jokin Turingin kone M_c.
- Huom! Binäärijonoihin, jotka eivät ole edellisen koodauksen mukaisia Turingin koneiden koodeja, liitetään jokin triviaali, kaikki syötteet hylkäävä (eli $\emptyset$ - kielen tunnistava) kone M_{triv}.

  Määritellään siis:

$$M_c = \begin{cases} \text{kone } M, \text{ jolla } c_M = c, \text{ jos } c \text{ on kelvollinen koodi;} \\ \text{kone } M_{\text{triv}}, \text{ muuten.} \end{cases}$$

- $\Rightarrow$ Saadaan luettelo kaikista aakkoston $\{0,1\}$ Turingin koneista, ja epäsuorasti myös kaikista aakkoston $\{0,1\}$ rekursiivisesti lueteltavista kielistä.
- Koneet ovat

$$M_\epsilon, M_0, M_1, M_{00}, M_{01}, \ldots,$$

- vastaavat kielet ovat

$$L(M_\epsilon), L(M_0), L(M_1), L(M_{00}), L(M_{01}), \ldots$$

  (indeksit kanonisessa järjestyksessä).

- Huom! Kukin kieli voi esiintyä luettelossa monta kertaa (saman kielen voi tunnistaa useilla koneilla).

6.4.2 Esimerkki ei-rekursiivisesti numeroutuvasta kielestä

Nyt voidaan muodostaa e-rekursiivisesti numeroituva kieli (ts. ratkeamaton ongelma) diagonalisaatiotekniikalla:

Lemma: Kieli

$$D = \{c \in \{0,1\}^* \mid c \notin L(M_c)\}$$

ei ole rekursiivisesti lueteltava.

Todistus. Oletetaan, että olisi $D = L(M)$ jollakin standardimallisella Turingin koneella M. Olkoon d koneen M binäärikoodi, so. $D = L(M_d)$. Tällöin on

$$d \in D \quad \Leftrightarrow \quad d \notin L(M_d) = D.$$

Ristiriidasta seuraa, että kieli D ei voi olla rekursiivisesti lueteltava. □

- Kieltä D vastaava päätösongelma: "Hylkääkö annetun koodin c esittämä Turingin kone syötteen c?"

- Kielen D muodostaminen kuvallisesti: jos kielten $L(M_\epsilon)$, $L(M_0)$, $L(M_1)$, ... karakteristiset funktiot esitetään taulukkona, niin kieli D poikkeaa kustakin kielestä taulukon diagonaalilla:

D $\searrow$	$L(M_\epsilon)$	$L(M_0)$	$L(M_1)$	$L(M_{00})$	$\cdots$
ϵ	$\not{0}^{1}$	0	0	0	$\cdots$
0	0	$\not{1}^{0}$	1	0	$\cdots$
1	0	0	$\not{1}^{0}$	1	$\cdots$
00	0	0	0	$\not{0}^{1}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

6.4.3 Universaalikielen ratkeamattomuus

- Aakkoston $\{0,1\}$ *universaalikieli* U määritellään:

$$U = \{c_M w \mid w \in L(M)\}.$$

- Olkoon A jokin aakkoston $\{0,1\}$ rekursiivisesti lueteltava kieli, ja olkoon M kielen A tunnistava standardimallinen Turingin kone. Tällöin on

$$A = \{w \in \{0,1\}^* \mid c_M w \in U\}.$$

- Ts. universaalikieli sisältää kaikki sellaiset konekoodi-syöte-parien muodostamat merkkijonot, missä kyseinen kone hyväksyy kyseisen syötteen.
- Siihen on siis tallennettu tieto kaikesta, mitä Turingin koneet voivat ratkaista!
- Myös kieli U on rekursiivisesti lueteltava. Kielen U tunnistavia Turingin koneita sanotaan *universaaleiksi Turingin koneiksi.*
- Sen sijaan U ei ole rekursiivinen (eikä sen komplementtikieli $\overline{U}$ siis rekursiivisesti lueteltava)
  $\Rightarrow$ emme voi tunnistaa täysin ratkeamattomia ongelmia laskennallisesti!

Lause: Kieli U on rekursiivisesti lueteltava.

**Todistus.*

- Muodostetaan 3-nauhainen kone universaalikone M_U, joka tunnistaa kielen U
- Laskennan aluksi tarkastettava syöte sijoitetaan koneen M_U ykkösnauhan alkuun.
- Tämän jälkeen kone toimii seuraavasti:

  1. Aluksi M_U tarkastaa, että syöte on muotoa cw, missä c on kelvollinen Turingin koneen koodi. Jos syöte ei ole kelvollista muotoa, M_U hylkää sen; muuten se kopioi merkkijonon $w = a_1 a_2 \ldots a_k \in \{0,1\}^*$ kakkosnauhalle muodossa

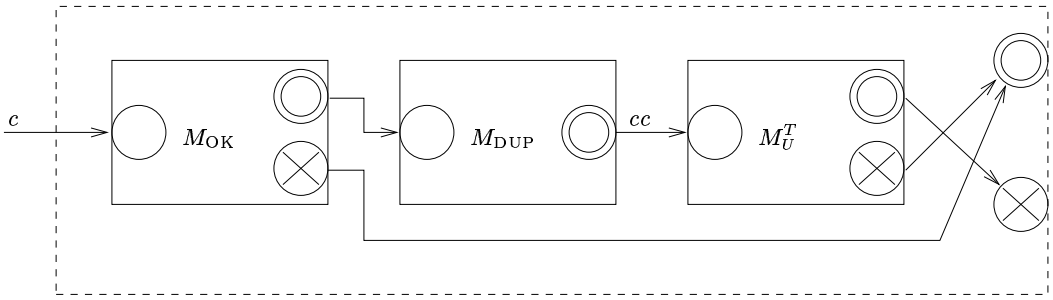
  $$00010^{a_1+1}10^{a_2+1}1 \ldots 10^{a_k+1}10000.$$

  2. Jos syöte on muotoa cw, missä $c = c_M$ jollakin koneella M, M_U:n on selvitettävä, hyväksyisikö kone M syötteen w.
     - tätä varten M_U säilyttää ykkösnauhalla M:n kuvausta c,
     - kakkosnauhalla simuloi M:n nauhaa, ja
     - kolmosnauhalla säilyttää tietoa M:n simuloidusta tilasta muodossa $q_i \sim 0^{i+1}$ (aluksi siis M_U kirjoittaa kolmosnauhalle tilan q_0 koodin 0).
  3. Alkutoimien jälkeen M_U toimii vaiheittain, simuloiden kussakin vaiheessa yhden koneen M siirtymän.
     - aluksi M_U etsii ykkösnauhalta M:n kuvauksesta kohdan, joka vastaa M:n simuloitua tilaa q_i ja merkkiä a_j.
     - Olkoon ykkösnauhalla oleva koodinkohta

     $$0^{i+1}10^{j+1}10^{r+1}10^{s+1}10^{t+1}.$$

     Tällöin M_U korvaa kolmosnauhalla merkkijonon 0^{i+1} merkkijonolla 0^{r+1}, kakkosnauhalla merkkijonon 0^{j+1} merkkijonolla 0^{s+1}, ja siirtää kakkosnauhan nauhapäätä yhden simuloidun merkin vasemmalle, jos $t = 0$, ja oikealle, jos $t = 1$.
     - Jos ykkösnauhalla ei ole yhtään simuloituun tilaan q_i liittyvää koodia, simuloitu kone M on tullut hyväksyvään tai hylkäävään lopputilaan; tällöin $i = k+1$ tai $i = k+2$, missä q_k on viimeinen ykkösnauhalla kuvattu tila. Kone M_U siirtyy vastaavasti lopputilaan q_{yes} tai q_{no}.

  □



Kuva 6.5: Diagonaalikielen D tunnistava kone M_D.

Lause: Kieli U ei ole rekursiivinen.

**Todistus.* Oletetaan, että kielellä U olisi totaalinen tunnistajakone M_U^T. Tällöin voitaisiin Lemman kielelle D muodostaa totaalinen tunnistajakone M_D seuraavasti:

- Olkoon M_{OK} totaalinen Turingin kone, joka testaa, onko syötteenä annettu merkkijono kelvollinen Turingin koneen koodi.
- Olkoon M_{DUP} totaalinen Turingin kone, joka muuntaa syötejonon c muotoon cc.
- Kone M_D muodostetaan koneista M_U^T, M_{OK} ja M_{DUP} yhdistämällä kaavion esittämällä tavalla.

Selvästi kone M_D on totaalinen, jos kone M_U^T on, ja

$$\begin{aligned} c \in L(M_D) &\Leftrightarrow c \notin L(M_{\mathrm{OK}}) \text{ tai } cc \notin L(M_U^T) \\ &\Leftrightarrow c \notin L(M_c) \\ &\Leftrightarrow c \in D. \end{aligned}$$

Mutta lemman mukaan kieli D ei ole rekursiivinen; ristiriita. $\square$.

Seuraus: Kieli

$$\tilde{U} = \{c_M w \mid w \notin L(M)\}$$

ei ole rekursiivisesti lueteltava.

Todistus. Kieli $\tilde{U}$ on oleellisesti sama kuin universaalikielen U komplementti $\bar{U}$; tarkasti ottaen on $\bar{U} = \tilde{U} \cup \mathrm{ERR}$, missä ERR on helposti tunnistettava rekursiivinen kieli

$$\begin{aligned} \mathrm{ERR} = \{&x \in \{0,1\}^* \mid x \text{ ei sisällä alkuosanaan} \\ &\text{kelvollista Turingin koneen koodia}\}. \end{aligned}$$

Jos siis kieli $\tilde{U}$ olisi rekursiivisesti lueteltava, olisi samoin myös kieli $\bar{U}$. Koska kieli U on rekursiivisesti lueteltava, seuraisi tästä, että U on peräti rekursiivinen. Mutta tämä on vastoin edellisen lauseen tulosta, mistä päätellään, että kieli $\tilde{U}$ ei voi olla rekursiivisesti lueteltava. □

6.5 *Ekskursio: Ratkeavuus ja ratkeamattomuus matematiikassa

Monessa laskettavuutta ja laskennan vaativuutta koskevassa tärkeässä tuloksessa meidän on kiittäminen matemaattikoja (mm. Gödeliä ja Churchia), joten pieni ekskursio matematiikkaan lienee paikallaan. Erityisesti Gödelin epätäydellisyyslause on suoraan sovellettavissa (universaali) Turingin koneita (mitä tahansa laskennan mallia) koskevaksi. Valotetaan ensin kuitenkin hämärien termien *rekursiivinen* ja *rekursiivisesti numeroituva kieli* alkuperää.

6.5.1 Rekursiiviset ja rekursiivisesti numeroituvat joukot matematiikassa

Jos ongelman voi ratkaista primitiivirekursiivisella funktiolla, se on laskennallisesti ratkeava ja voimme jopa antaa laskennan vaativuudelle ylärajan. Jos sen voi ratkaista vain rekursiivisella funktiolla, se ratkeaa laskennallisesti eli äärellisessä ajassa, mutta emme voi antaa mitään ylärajaa sen vaativuudelle. Jos ongelma kuuluu rekursiivisesti numeroituviin kieliin, emme voi edes antaa ratkaisevaa laskennallista funktiota, mutta kylläkin deklaratiivisen määritelmän, mitä vastauksen tulisi täyttää. Tämän pohjalta voimme muodostaa ns. *osittaisrekursiivisen funktion.* Määriteltyä vastausta ei kuitenkaan voida laskea äärellisessä ajassa (kaikilla syötteillä). Jos ongelma ei ole edes rekursiivisesti numeroituva, emme pysty antamaan edes osittaisrekursiivista funktiota, joka laskisi vastauksen.

Rekursiiviset funktiot

Sanotaan, että joukko $A \subseteq \mathbb{N}$ on rekursiivinen, jos sen tunnistusongelma π_A (oikeammin joukon A *karakteristinen funktio*) on rekursiivinen funktio. Huom! Mikä tahansa merkkijono voidaan aina koodata luonnollisena lukuna eli kieli A on itse asiassa $\mathbb{N}$:n osajoukko.

Määritellään ensin *primitiivirekursiiviset funktiot*

Määritelmä: Primitiivirekursiivisten funktioden perhe on pienin funktioperhe luonnollisten lukujen joukossa ($f : \mathbb{N} \to \mathbb{N}$), joka sisältää funktiot

$Z(n) = 0$ (nollafunktio)
$S(n) = n + 1$ (seuraajafunktio)

$Pr_i{}^n(x_1, ..., x_n) = x_i$ (projektiofunktio)

sekä näistä yhdistämällä ja seuraavalla ns. rekursiosäännöllä saatavat funktiot.

Rekursiosääntö: Olkoon f n-paikkainen primitiivirekursiivinen funktio ja g $n+2$-paikkainen primitiivirekursiivinen funktio. Tällöin $n+1$-paikkainen funktio h,

$h(0, x_1, ..., x_n) = f(x_1, ..., x_n)$
$h(y+1, x_1, ..., x_n) = g(y, h(x_1, ..., x_n), x_1, ..., x_n)$

on primitiivirekursiivinen.

Kun $n = 0$, niin $h(0) = a$ (vakio) ja $h(y+1) = g(y, h(y))$.

Esimerkiksi yhteenlasku, kertolasku, eksponenttifunktio, vakiofunktio ja luonnollisiin lukuihin rajoitettu vähennyslasku ovat primitiivirekursiivisia. Uusia primitiivirekursiivisia funktioita saadaan myös *rajoitetulla minimalisaatiolla*, merk.
$f(y, x_1, ..., x_n) = (\mu z \leq y)R(z, x_1, ..., x_n)$, jos

$$f(y, x_1, .., x_n) = \begin{cases} \text{pienin } z \leq y, \text{ jolle } R(z, x_1, ..., x_n), & \text{jos tällainen z on olemassa, ja} \\ 0, & \text{muuten} \end{cases}$$

missä R on prim.rek. relaatio.

Rekursiivisten funktioiden perhe määritellään aivan samaan tapaan kuin primitiivirekursiivisten funktioiden perhe, mutta sallimme rajoittamattoman minimalisaation. Merk. $f(x_1, ..., x_n) = \mu y R(y, x_1, ..., x_n)$, jos

$$f(x_1, ..., x_n) = \text{ pienin } y, \text{ jolle } R(y, x_1, ..., x_n), \text{ jos tällainen } y \text{ on olemassa.}$$

Primitiivirekursiivisten ja rekursiivisten funktioiden ero on seuraava: jos $f(x)$ on primitiivirekursiivinen, on $f(n)$:n laskemiseen kuluva aika ennalta arvioitavissa. (Ts. voimme antaa ratkaisevan Turingin koneen aikavaativuudelle ylärajan.) Jos funktio on vain rekursiivinen, tiedämme vain, että $f(n)$ on ratkeava, mutta emme tiedä, montako laskenta-askelta ratkaisu vaatii (ts. vastaavan TM:n laskenta päättyy äärellisessä ajassa, mutta emme voi antaa ennalta mitään arviota, kauanko laskenta tulee kestämään.)

Esimerkiksi Ackermannin funktio A on rekursiivinen, mutta ei primitiivirekursiivinen.

$A(0, x) = x + 1$
$A(n+1, 0) = A(n, 1)$
$A(n+1, x+1) = A(n, A(n+1, x))$

Ackermannin funktio on sikäli kiintoisa, että kaikille primitiivirekursiivisille funktioille f löytyy n, jolle $f(x) < A(n,x)$ kaikilla x. Ja kun n lähenee ääretöntä, antaa Ackermannin funktio ylärajan (majorantin) myös kaikille rekursiivisille funktioille! Toisin sanoen se antaa ylärajan minkä tahansa ratkeavan ongelman aikavaativuudelle.

Rekursiivisesti numeroituvat joukot

Rekursiivisen kielen tunnistusongelma (kuuluuko sana x kieleen A) on rekursiivinen eli algoritmisesti ratkeava, joskaan emme voi antaa laskennan aikavaativuudelle mitään ylärajaa.

Toisaalta jokainen epätyhjä rekursiivinen kieli A (sen sisältämät sanat) voidaan aina numeroida rekursiivisella funktiolla. Määritellään A:n sanoille numerointi: $A = \{f(n)|n \in \mathbb{N}\}$, missä $f(n) = n$, jos $\pi_A(n) = 1$ ja a_0, jos $\pi_A(n) = 0$ $(a_0 \in A)$

$$f(n) = \begin{cases} \text{n}, & \text{jos } \pi_A(n) = 1 \text{ ja} \\ a_0, & \text{jos } \pi_A(n) = 0 \ (a_0 \in A) \end{cases}$$

Määritelmä: Joukko $A \subseteq \mathbb{N}$ on rekursiivisesti numeroituva, jos $A = \emptyset$ tai on olemassa rekursiivinen funktio $f : \mathbb{N} \to \mathbb{N}$ s.e. $A = \{f(n)|n \in \mathbb{N}\}$.

Kaikki rekursiiviset joukot ovat siis myös rekursiivisesti numeroituvia. Rekursiivisesti numeroituvien joukkojen perhe on kuitenkin aidosti suurempi kuin rekursiivisten eli on olemassa joukkoja (kieliä), jotka ovat rekursiivisesti numeroituvia mutta eivät rekursiivisia (Churchin lause).

Rekursiivisuuden ja rekursiivisen numeroituvuuden välinen ero on se, että rekursiivisen funktion arvon voi aina laskea äärellisen monella laskenta-askeleella ja rekursiivisen joukon tunnistusongelma on siis ratkeava (ts. meillä on algoritmi, joka ratkaisee sen). Sen sijaan, jos joukko on vain rekursiivisesti numeroituva, emme voi määritellä mitään funktiota, joka tunnistaisi sen koska funktion määritelmä itsessään edellyttää määriteltävyyttä (voimme laskea funktion arvon kaikilla määrittelyjoukon alkioilla). Voimme kyllä antaa deklaratiivisen määritelmän, mitä jokin rekursiivisesti numeroituva joukko pitää sisällään ja kuvata tällaisia kieliä lukuteorian kaavoilla. Meillä on esim. käsitys miten kaavan $\Phi(x_1, ..., x_n)$ totuus selvitetään, vaikkei universaaliväitteen $\forall x \Phi(x)$ totuus ole laskennallisesti ratkeava (pitäisi tutkia $\Phi(n)$:n totuus äärettömän monella $n \in N$). Tällaista osittain määriteltyä "funktiota", joka on laskettavissa vain joillain määrittelyjoukon arvoilla, kutsutaan osittaisrekursiiviseksi funktioksi.

Mitä sitten ovat täysin ratkeamattomat eli ei-rekursiivisesti numeroituvat joukot

matematiikassa? Niille emme voi antaa edes osittaisrekursiivista funktiota (deklaratiivista määritelmää), joka jotenkin kuvaisi joukon sisällön.

6.5.2 Gödelin epätäydellisyyslause

Gödelin epätäydellisyyslause on matemaattinen vastine Turingin koneiden itsetutkiskelusta seuraavalle epätäydellisyydelle.

Gödel keksi menetelmän, jolla mikä tahansa lukuteorian kaava (tai predikaattilogiikan lause) voidaan koodata yksikäsitteisenä kokonaislukuna, kaavan Gödel-lukuna. Toisin sanoen jokaiseen kaavaan voidaan liittää tasan yksi luku ja jokaiseen lukuun liittyy korkeintaan yksi kaava (osa kokonaisluvuista ei vastaa mitään kaavaa). Tällä tavalla kaavassa päästään viittaamaan toisiin kaavoihin, mikä mahdollistaa kaavojen paradoksaalisen itsetutkiskelun. ($\leftrightarrow$ Turingin koneiden koodaus lukuina)

Gödel-numerointi tapahtuu seuraavasti. Ensin liitetään jokaiseen lukuteorian merkkiin luonnollinen luku:

$\#(0) = 1$
$\#(1) = 2$
$\#(+) = 3$
$\#(\times) = 4$
$\#(=) = 5$
$\#(() = 6$
$\#()) = 7$
$\#(\wedge) = 8$
$\#(\vee) = 9$
$\#(\neg) = 10$
$\#(\exists) = 11$
$\#(\forall) = 12$
$\#(v_0) = 13$
$\#(v_1) = 14$
$\#(v_2) = 15$
....

missä muuttujan v_n järjestysnumero on $\#(v_n) = 13 + n$.

Olkoon w nyt edellisistä merkeistä muodostuva sana $w = w_0 w_1 ... w_n$. Tällöin w:n Gödel-luku on

$$\lceil w \rceil = \Sigma_{i=0}{}^{n} p_i{}^{(\#w_i)} = p_0{}^{(\#(w_0)} p_1{}^{(\#w_1)} ... p_n{}^{(\#w_n)},$$

missä $p_0, p_1, ...$ ovat peräkkäiset alkuluvut 2,3,5,7,...

Esimerkiksi $\lceil (1 = 1) \rceil = 2^6 \times 3^2 \times 5^5 \times 7^2 \times 11^7$.

Tämä itsessään hyvin yksinkertainen työkalu mahdollisti mahtavan tuloksen: sen avulla Gödel todisti kuuluisan Epätäydellisyyslauseensa, jonka mukaan mikään konsistentti aksioomajärjestelmä (esim. aritmetiikka) ei voi olla täydellinen (t.s. se ei voi sisältää kaikki tosia lauseita teoreemoinaan). Gödelin todistus oli myös yksinkertaisuudessaan suunnattoman nerokas ja perustui Valehtelijan paradoksiin (Cantorin diagonalisointiargumenttiin): jos kaava ϕ sanoo "Kaava, jonka Gödel-luku on x ei ole todistuva" ja ϕ:n oma Gödel-luku on x, seuraa ristiriita: toisin sanoen emme voi ratkaista kaikista mahdollisista tosista lauseista, ovatko ne todistuvia vai eivät. (Lause ϕ sanoo siis "En ole todistuva".)

Huom! Voimme kyllä päätellä, että lause ϕ on tosi – sehän ei ollut todistuva kyseisessä järjestelmässä – mutta järjestelmä itse ei kykene ratkaisemaan sitä, mikäli pidämme kiinni järjestelmän konsistenssista. Jos taas sallimme, että järjestelmä on epäkonsistentti (ts. siinä pätee sekä jokin lause että sen negaatio), niin mikä tahansa lause voidaan todistaa oikeaksi. Järjestelmä voi siis täydellinen, vain jos se on epäkonsistentti, mutta jos se on konsistentti, se on epätäydellinen. [4]

Tällä viattoman oloisella vain lausekalkyylia koskevalla lauseella on kuitenkin todella laajat seuraukset. Siitä seuraa, ettei mikään aksiomatisointijärjestelmä, esim. algebra, tai mikään laskennanmalli, kuten Turingin koneet, ole täydellinen!

6.5.3 Kirjallisuutta

Hofstadter, Douglas R.: Gödel, Escher, Bach: An Eternal Golden Braid. Vintage Books, 1979. (chapters XIV-XV)
Nagel, E. & Newman, J.K.: Gödel's Proof. New York University Press, 1986.
Rucker, Rudy: Mieli ja äärettömyys.
Väänänen, Jouko: Matemaattinen logiikka. Gaudeamus, 1987.

[4] Valehtelijan paradoksin matemaattinen versio, Tarskin lause, sanoo, että joukko $T = \{x | x$ on toden lauseen ϕ Gödel-numero$\}$ ei ole rekursiivinen, eikä edes rekursiivisesti numeroituva! Sen sijaan jos kiinnitämme jonkin "efektiivisen" ($\sim$ algoritmisesti ratkeavan) aksioomajärjestelmän $\mathcal{A}$, niin joukko $T' = \{x | x$ on $\mathcal{A}$:ssa todistuvan lauseen ϕ Gödel-numero$\}$ on rekursiivisesti numeroituva, mutta ei rekursiivinen.