Mitä merkitsee käsite “laskettavuus” ?

- Laskettavuus on tässä yhteydessä käännös sanasta *computability*.

- *Algoritmi* merkitsee intuitiivisesti äärellistä sääntökokoelmaa, joiden perusteella syötteestä (input) saadaan *laskettua (compute)* tuloste (output) äärellisen monella askeleella.

- Vaaditaan myös, että syöte on spesifioitava äärellisesti.

Esimerkkejä

- Kokonaislukujen tulo on algoritmisesti laskettavissa: Syötteenä on kaksi kokonaislukua, tuloste on näiden lukujen tulo.

$$\begin{array}{r} 17 \\ \times\, 15 \\ \hline {\scriptstyle 1}\,85 \\ +17 \\ \hline 255 \end{array} \quad \not{3}$$

- Kahden kokonaisluvun osamäärä on algoritmisesti laskettavissa: Jakokulmassa esimerkiksi 255/18 tuottaa 14 sekä jakojäännöksen 3.

- Kokonaisluvun alkutekijähajotelma on algoritmisesti selvitettävissä.

- *Propositiomuuttujista* x_1, x_2, x_3, ... voidaan muodostaa *propositioita* käyttämällä *konnektiiveja*

$$\wedge \quad \text{(ja)} \qquad \vee \quad \text{(tai)} \qquad \neg \quad \text{(ei)}$$

sekä sulkumerkkejä. Muuttujat voivat saada arvon 0 tai 1. Koko proposition arvo määräytyy propositiomuuttujien arvoista seuraavien sääntöjen nojalla:

$$\begin{array}{lll} 0 \wedge 0 = 0 & 0 \vee 0 = 0 & \\ 0 \wedge 1 = 0 & 0 \vee 1 = 1 & \neg 0 = 1 \\ 1 \wedge 0 = 0 & 1 \vee 0 = 1 & \neg 1 = 0 \\ 1 \wedge 1 = 1 & 1 \vee 1 = 1 & \end{array} .$$

Sanotaan, että propositio on *toteutuva*, jos se saa arvon 1 jollakin muuttujien arvojen valinnoilla. Esimerkiksi

$$(\neg x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3)$$

on toteutuva propositio: $x_1 = 1$, $x_2 = 1$ ja $x_3 = 0$ on *toteuttava arvotus*. Propositio $x_1 \wedge \neg x_1$ puolestaan ei ole toteutuva.

- Kysymys, onko jokin annettu propositio toteutuva, on algoritmisesti ratkaistavissa.

- Kysymys, pysähtyykö annettu *algoritmi* tietyllä syötteellä *ei ole algoritmisesti ratkaistavissa* (undecidability).

Formaaliset kielet

- Tässä Σ on äärellinen joukko, jota kutsutaan *aakkostoksi*.

- *Formaalinen Σ-kieli* on mikä tahansa kokoelma Σ:n alkioista muodostettuja äärellisiä jonoja. Tällaisia jonoja kutsutaan *sanoiksi*.

Esimerkki:

Olkoon $\Sigma = \{0, 1\}$ niin kutsuttu *binääriaakkosto*. Tällöin

$$\begin{aligned} \mathcal{L}_1 &= \{1, 11, 1110, 0000, 10000, 1111111001110111111\} \quad \text{ja} \\ \mathcal{L}_2 &= \{10110^n1 \mid n \in \mathbb{N}\} \end{aligned}$$

ovat *binäärisiä formaalisia kieliä*.

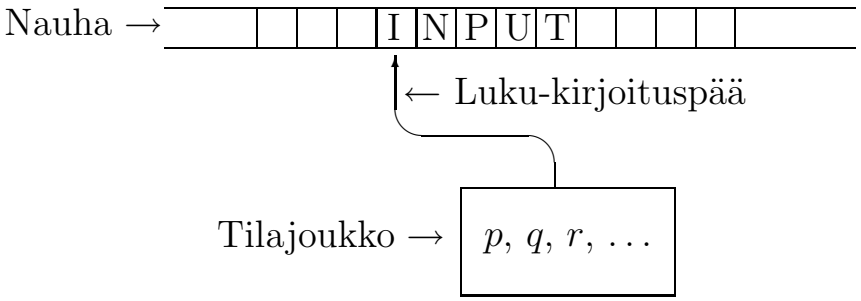
- Algoritmin toiminta voidaan siis käsittää laskentana (computation), jossa syötesanasta (input word) muodostetaan tulostesana (output word).

- Algoritmisen "kyllä - ei" -ongelman ratkaisu vastaa formaalisen kielen tunnistamista.

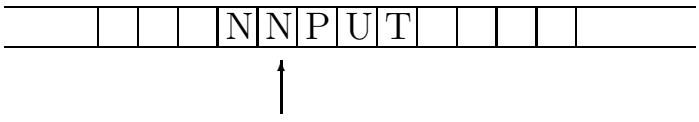
Turingin kone

- Äärellinen *aakkosto* $\Sigma = \{a, b, \ldots\}$
- Äärellinen *tilajoukko* $Q = \{p, q, r, \ldots\}$
- *Transitiosäännöt* δ
- *Nauha*

Transitiosäännöt ovat muotoa $\delta(q, a) = (p, b, -/0/+)$.



Jos esimerkissä kone on tilassa q sekä $\delta(q, I) = (p, N, +)$, kirjoittaa kone I:n tilalle N:n, luku-kirjoituspää siirtyy askeleen oikealle ja kone siirtyy tilaan p:



Ilmeisesti Turingin koneen toiminta on "fysikaalisesti toteutettavissa".

Church-Turingin teesi: Turingin kone on intuitiivisen algoritmikäsitteen matemaattinen vastine

Laskenta mielekkäässä ajassa

- Algoritmin syötteen suuruudella tarkoitetaan syötesanan pituutta.
- Algoritmin käyttämällä ajalla tarkoitetaan tarvittavien laskuaskelten määrää.
- Algoritmin *kompleksisuus (complexity)* kuvaa algoritmin käyttämän ajan riippuvuutta syötteen suuruudesta.
- Yleisen sopimuksen mukaan katsotaan, että polynomaalisessa ajassa ratkeavat ongelmat ovat käytännössä ratkeavia (tractable), muut käytännössä ratkeamattomia (intractable).
- Merkintää **P** käytetään deterministisellä Turingin koneella polynomiajassa tunnistettavien kielten luokasta.

Esimerkkejä

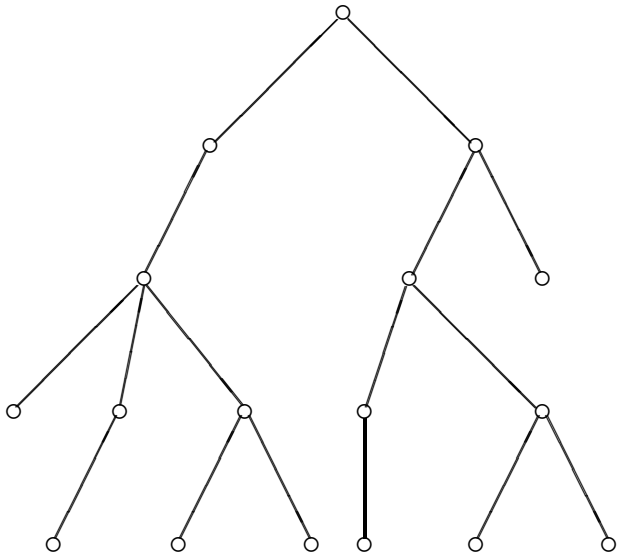
- Kahden kokonaisluvun tulo voidaan laskea polynomiajassa.
- Ei tiedetä, voidaanko proposition ratkeavuus selvittää polynomiajassa.

Epädeterministinen Turingin kone

- Kukin transitiosääntö voi määrätä useita toimintavaihtoehtoja.

- Intuitiivisesti voidaan ajatella, että kyseessä on Turingin kone, joka "kloonaa" itsensä aina, kun toimintavaihtoehtoja on useita (kutakin vaihtoehtoa kohti yksi klooni). Kukin klooni jatkaa toimintaa muista riippumatta.

- Epädeterministisen Turingin koneen toimintaa voidaan kuvata puurakenteen avulla (laskentapolut)



- Kaikki epädeterministisellä Turingin koneella suoritettavat tehtävät voidaan suorittaa myös deterministisellä Turingin koneella ja päinvastoin.

- Merkintää **NP** käytetään epädeterministisesti polynomiajassa tunnistettavien kielten luokasta.

- Joka tapauksessa $\mathbf{P} \subseteq \mathbf{NP}$, mutta ei tiedetä, onko $\mathbf{P} \neq \mathbf{NP}$. Näin otaksutaan vahvasti.

- Epädeterministisen koneen realisointi on kyseenalaista.

- *Probabilistinen algoritmi* ei aina anna oikeaa tulosta.

- Ei tiedetä, voidaanko alkulukutestaus suorittaa polynomaalisessa ajassa *varmuudella*. Jos kuitenkin hyväksytään pieni virhepäätelmän mahdollisuus, voidaan testaus suorittaa polynomiajassa.

- Epädeterministinen Turingin kone on *virherajoitettu Turingin kone*, mikäli laskennan aika kullakin syötteellä on polynomaalinen ja mikäli *ainakin* 3/4 *laskentapoluista johtaa oikeaan tulokseen*

- Virherajoitetuilla koneilla tunnistettavien kielten luokkasta käytetään merkintää **BPP** (Bounded Probability of error in Polynomial time).

Laajennettu Church-Turingin teesi: Käytännössä ratkeavien ongelmien luokka on **BPP**.

- Tiedetään, että $\mathbf{P} \subseteq \mathbf{BPP}$, mutta ei tiedetä, onko $\mathbf{P} \neq \mathbf{BPP}$.

- Ei tiedetä, miten luokat **BPP** ja **NP** suhtautuvat toisiinsa.

DNA-Laskenta

Kvanttilaskenta

Kvanttilaskenta

- R. Feynman 1982, D. Deutsch 1985, P. Shor 1994.

- Kvanttimekaaniset objektit voivat olla *klassisten tilojen* (merkitään esim. $|0\rangle$, $|1\rangle$, ...) lisäksi myös *superpositiossa*

$$c_0 |0\rangle + c_1 |1\rangle + \ldots + c_{n-1} |n-1\rangle.$$

  Kertoimet c_0, c_1, ... ovat kompleksilukuja, jotka toteuttavat ehdon

$$|c_0|^2 + |c_1|^2 + \ldots + |c_{n-1}|^2 = 1.$$

- Kaksitilaista kvanttimekaanista systeemiä kutsutaan *kvanttibitiksi* (qubit). Kvanttibitti voi yleisesti olla tilassa

$$c_0 |0\rangle + c_1 |1\rangle, \quad |c_0|^2 + |c_1|^2 = 1.$$

- Pituutta n olevalla *kvanttirekisterillä* tarkoitetaan n:ää kvanttibittiä, jotka ovat keskenään vuorovaikutuksessa. Tällainen rekisteri voi yleisesti olla tilassa

$$\begin{aligned} & c_0 |0\ldots00\rangle + c_1 |0\ldots01\rangle + c_2 |0\ldots10\rangle + \ldots + c_{2^n-1} |1\ldots11\rangle \\ = \; & c_0 |0\rangle + c_1 |1\rangle + c_2 |2\rangle + \ldots + c_{2^n-1} |2^n-1\rangle. \end{aligned}$$

- Kvanttibittien ja rekisterien aikakehitys on *unitaarista*:

$$|x\rangle \to U |x\rangle, \; U \text{ on unitaarinen lineaarikuvaus.}$$

- Voidaan todistaa: Jos Turingin kone pystyy laskemaan syötesanasta x tulostesanan $f(x)$ polynomiajassa, niin on olemassa jono unitaarikuvauksia U_1, U_2, ..., joilla saadaan aikaan *kvanttilaskenta*

$$U_m \dots U_2 U_1 \,|x, 0\rangle = |x, f(x)\rangle.$$

  Lisäksi kuvausten U_1, U_2 ... määrä on polynomaalinen x:n pituuteen nähden.

Esimerkki

Seuraavat ehdot määrittelevät unitaarisen kuvauksen:

$$\begin{aligned} U\,|0\rangle &= \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ U\,|1\rangle &= \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$

Jos kvanttirekisteri on aluksi tilassa

$$|0 \dots 00\rangle = \underbrace{|0\rangle \dots |0\rangle\,|0\rangle}_{n \text{ kappaletta}},$$

saadaan kuvausta U kuhunkin kvanttibittiin soveltamalla tila

$$\begin{aligned} & \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \dots \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ = \;& \tfrac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle + \dots + |2^n - 1\rangle). \end{aligned}$$

Tällöin rekisteri sisältää kaikki luvut 0, 1, ..., $2^n - 1$ yhtäläisesti painotettuina.

- Kvanttilaskenta sisältää epädeterministisen laskennan piirteitä: Rekisteri voidaan aluksi preparoida tilaan

$$\frac{1}{\sqrt{2^n}}(|0\rangle\,|0\rangle + |1\rangle\,|0\rangle + |2\rangle\,|0\rangle + \ldots + |2^n-1\rangle\,|0\rangle).$$

  Soveltamalla nyt kuvauksia U_1, U_2, ... saadaan

$$\frac{1}{\sqrt{2^n}}(|0\rangle\,|f(0)\rangle + |1\rangle\,|f(1)\rangle + \ldots + |2^n-1\rangle\,|f(2^n-1)\rangle).$$

  Tällöin on siis laskettu 2^n arvoa *samaan aikaan*.

- Superposition

$$\frac{1}{\sqrt{2^n}}(|f(0)\rangle + |f(1)\rangle + \ldots + |f(2^n-1\rangle)$$

  havainnointi tuottaa kuitenkin vain yhden arvon $f(x)$, kunkin todennäköisyydellä $1/2^n$.

- Yleisemmin, superposition

$$c_0\,|0\rangle + c_1\,|1\rangle + \ldots + c_{n-1}\,|n-1\rangle.$$

  havainnointi tuottaa arvon k todennäköisyydellä $|c_k|^2$.

- Interferenssi

- Peter W. Shor (1994): Kvanttialgoritmilla voidaan selvittää luvun alkutekijät luvun pituuteen nähden polynomaalisessa ajassa.

- Peter W. Shor (1994): Kvanttialgoritmilla voidaan laskea *diskreetti logaritmi* polynomaalisessa ajassa.

- 1985 D. Deutsch määritteli kvanttimekaanisen version Turingin koneesta (Quantum Turing machine).

- **BQP** on niiden kielten luokka, jotka voidaan tunnistaa kvanttimekaanisella Turingin koneella polynomiajassa.

- On voimassa $\mathbf{P} \subseteq \mathbf{BPP} \subseteq \mathbf{BQP}$, mutta ei tiedetä, onko $\mathbf{BPP} \neq \mathbf{BQP}$.

- Ei tiedetä, onko $\mathbf{NP} \subseteq \mathbf{BQP}$.