Turingin kone

Laskennan teorian opintopiiri
Anna Haataja

Turingin koneet kehitti brittiläinen matemaatikko Alan Turing 1930-luvulla algoritmien havainnollistamiseksi. Turingin vuonna 1936 julkaisemaa tutkimusta voidaan pitää yhdestä merkittävimmistä tietojenkäsittelytieteen edistysaskelista; siitä syntyi kokonaan uusi matemaattisen logiikan haara, joka myöhemmin enemmän tai vähemmän sulautui tietojenkäsittelytieteeseen.

Turingin koneelle on olemassa lukuisia erilaisia määritelmiä, joista tässä lyhyesti esitellään kaksi: Turingin kone on äärellinen automaatti, joka koostuu äärettömän pitkästä nauhasta, joka toimii koneen muistina, lukupäästä ja äärellisestä joukosta tiloja. Kone kykenee liikkumaan nauhalla oikealle ja vasemmalle, pysähtymään, lukemaan nauhan solun ja tulostamaan sen paikalle jonkin sallitun merkin. Matemaattisesti Turingin kone siis määritellään seitsikkona $(Q, \Gamma, b, \Sigma, \delta, q, F)$, missä Q on äärellinen joukko tiloja, Γ äärellinen nauha-aakkosto, johon tyhjä merkki b kuuluu, Σ syöteaakkosto, δ siirtymäfunktio, q alkutila ja F hyväksyvien lopputilojen joukko. Erityisesti on huomattava, että tyhjä merkki ei voi kuulua syöteaakkostoon. Siirtymäfunktio puolestaan vastaa lukupään siirtymistä yhden askeleen verran oikealle tai vasemmalle.

Kumpikin näistä määritelmistä tuottaa klassisen, yksinkertaisimman mahdollisen tyypin Turingin koneen. Tätä perustyyppiä on varioitu erilaisin menetelmin esimerkiksi lisäämällä nauhojen ja lukupäiden määrää. Muuttamalla seuraajafunktion määritelmää niin, että funktio antaa mahdollisesti useita eri vaihtoehtoja seuraavaksi siirroksi, voidaan määritellä epädeterministinen Turingin kone. Nämä ovat ideatasolla tavalisia yksinauhaisia deterministisiä Turingin koneita parempia suorittamaan haluttuja tehtäviä, mutta voimme myös todistaa, että itse asiassa jokaista moninauhaista konetta voidaan simuloida yksinauhaisella koneella ja jokaista epädeterminististä konetta deterministisellä.

Huomattavaa on, että molemmat aiemmista Turingin koneen perusmääritelmistä tuottavat rakenteeltaan todella yksinkertaisen koneen, joka kuitenkin pystyy simuloimaan lähes kaikkea modernin tietokoneen oleellista toimintaa. Koska aito Turingin kone vaatii rajoittamattoman mittaisen nauhan, ei oikeaa Turingin konetta pystytä rakentamaan. Kuitenkaan kukaan ei koostanut edes toimivaa ”pienoismallia” ennen kuin vuosikymmeniä Turingin paperin jälkeen. Onkin siis hämmästyttävää, miten Turing sai määriteltyä koneensa vastaamaan näinkin hyvin nykyaikaisten tieokoneiden toimintaa, varsinkin kun Turingin koneilla ei ollut teknisesti juurikaan osaa tietokoneiden kehityksessä.

Seuraavaksi käsitellään yksinkertainen esimerkki kahden luonnollisen luvun yhteenlaskun suorittavasta Turingin koneesta: Koneemme nauha-aakkosto on nyt joukko (0,1). Määritellään, että lukua n vastaa koneen nauhalla $n+1$ ykköstä ja kaksi lukua erotetaan toisistaan nollalla. Haluamme myös, että kone antaa vastauksen $n+m$ yhtenä $n+m+1$ ykköstä sisältävänä merkkijonona. Alussa koneelle kirjoitetaan siis luvut n ja m sen nauhalle muodossa $n+1$ ykköstä, nolla, $m+1$ ykköstä. Koneen alkutila on vasemmalta katsottuna ensimmäisen ykkösen kohdalla. Seuraavaksi käsketään konetta siirtymään nauhalla oikealle, kunnes kone lukee ensimmäisen nollan, ja kirjoittamaan sen paikalle ykkösen. Nyt nauhalla on $n+1+1+m+1$ ykköstä yhtenä jonona eli kaksi ykköstä liikaa. Käsketään siis konetta siirtymään oikealle, kunnes se lukee ensimmäisen nollan ja pääsee siis merkkijonon viimeisen ykkösen oikean puoleiseen soluun. Seuraavaksi koneen tulee siirtyä kaksi kertaa yksi askel vasemmalle ja tulostaa molemmilla kerroilla solun ykkösen tilalle nollan. Viimeiseksi kone palaa taas merkkijonon vasemmanpuoleiseen päähän. Näin nauhalle on saatu tulostumaan yksi merkkijono $n+m+1$, joka siis on haluttu summa. Vastaavaan tapaan voidaan ohjelmoida useampipaikkainen yhteenlasku.

Kuten edellisestä voi huomata, Turingin koneen ohjelmoiminen ja toiminta on jokseenkin hidasta, joten se ei ole koskaan voinut kilpailla suorituskyvyillään tai tehokkuudella nykyaikaisten tietokoneiden ja ohjelmointikielien kanssa. Turingin koneen suosio perustuukin sen käyttökelpoisuuteen teoreettisten tuloksien todistamisessa ja havainnollistamisessa. Erityisesti universaali Turingin kone eli UTM, jonka Turing jo 1936 määritteli, on käytössä lukuisissa standarditodistuksissa. UTM:n erityispiirre muihin Turingin koneisiin verrattuna on sen kyky simuloida kaikkia muita Turingin koneita eli ottaa mikä tahansa koneista syötteekseen sen syötteen kera.

Kuuluisin kaikista Turingin koneella todistettavista tuloksista on pysähtymisongelma, joka kysyy, voidaanko sanoa, pysähtyykö mielivaltainen Turingin kone eli ohjelma annetulla syötteellä. Tähänkin ongelmaan on annettu useita eri tekniikoihin perustuvia todistuksia, muun muassa Alonzo Churchin lambda-kalkyyliin perustuva versio julkaistiin vain kuukautta myöhemmin kuin Turingin alkuperäinen UTM:ään perustuva todistus. Matemaatikoiden onneksi vastaus kysymykseen on, että emme voi rakentaa konetta, joka osaisi kertoa kaikissa tapauksissa vastauksen. Siis ei voida myöskään rakentaa tietokonetta, joka voisi selvittää kaikki todet lauseet annetusta aksiomaattisesta mallista.

Turingin todistus pohjautuu kahden sisäkkäisen funktion tai ohjelman määrittelyyn:

Olkoon *f(i,x)=1, jos ohjelma i pysähtyy syötteellä x*
0, muutoin
ja *g(i)=1, jos h(i,i)=0*
muutoin määrittelemätön,
kun *h* on mielivaitainen kaksipaikkainen laskettava funktio.

Siis funktio *f* antaa arvon *1*, jos sen syötteenään saama ohjelma pysähtyy omalla syötteellään, ja jos ohjelma ei pysähdy, niin arvon *0*. Funktio *f* siis toimii UTM:n tavoin todistuksessa. Funktio *g* palauttaa puolestaan arvon *1*, jos sen syötteenä oleva ohjelma *i* johtaa jonkin toisen funktion *h* syötteenä arvoon *0*. Muussa tapauksessa *g* on määrittelemätön, eli se Turingin kone, joka vastaa funktiota *g* ei pysähdy lainkaan.

Turingin suurin oivallus oli tutkia, mitä tapahtuu, jos funktio *h* onkin *f*. Tällöin *g* pysähtyy jos ja vain jos *f* ***ei*** pysähdy, kun sen syötteenä on ohjelma, joka on saanut syötteekseen itsensä. Eli tilanteessa, jossa *f(i,i)=1*, *g(i)* ei pysähdy, ja jotta funktio *f* voisi saada arvon *0,* se itse ei voi pysähtyä ollenkaan. Kummassakin tapauksessa funktiota *g* vastaava Turingin kone jää ikuiseen luuppiin, mikä tietenkin tarkoittaa sitä, että olemme löytäneet ohjelman ja syötteen, josta kone ei osaa sanoa pysähtyykö se vaiko ei.

Turingin koneita käytetään perustana lukuisissa eri matemaattisen logiikan ja tietojenkäsittelytieteen peruskäsitteiden määritelmissä ja se esiintyy myös kuuluisassa Churchin-Turingin teesissä. Teesin mukaan kaikki laskettavissa oleva voidaan laskea universaalin Turingin koneen avulla eli toisin sanoin algoritmein laskettavien ongelmien joukko vastaa Turingin koneilla ratkaistavissa olevien ongelmien joukkoa. Muita esimerkkejä Turingin koneiden käytöstä määritelmissä on esimerkiksi Turing-täydellinen järjestelmä, joka tarkoittaa, että kyseinen järjestelmä voi simuloida UTM:ää.

Algoritmien ja Turing koneiden merkitystä tietojenkäsittelytieteen ja matemaattisen logiikan kehitykselle on vaikea korostaa kylliksi: niin moni alojen keskeinen tulos joko perustuu niihin tai on palautettavissa Turingin koneiden teoriaan. Vaikka käytännössä Turingin koneet ovat mahdottomia toteuttaa, niiden teoreettinen voima takaa sen, että niiden avulla voidaan vielä tehdä paljon uraauurtavaa tutkimusta laskennan ja logiikan saralla.

Lähteet:
Cutland, Nigel. Computability: an Introduction to recursive function theory. Cambridge University Press, 1980.
http://plato.stanford.edu/entries/turing-machine/
http://en.wikipedia.org/wiki/Turing_machine
http://www.cs.helsinki.fi/u/tpkarkka/opetus/09s/lama/luento10.pdf