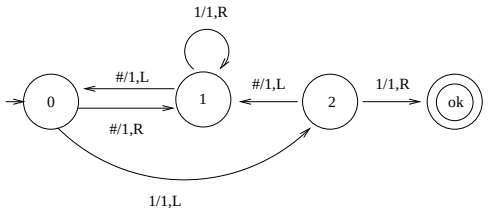
7.7.5 *Joitain hauskoja ratkeamattomia ongelmia

Busy Beaver

Busy Beaver-ongelmassa tehtävänä on keksiä n-tilainen Turingin kone (jonka on siis joka askeleella siirryttävä joko oikealle tai vasemmalle – paikallaan ei saa pysyä), joka aloittaa toimintansa tyhjällä nauhalla ja on pysähtyessään kirjoittanut nauhalle mahdollisimman monta merkkiä. Aakkostona voidaan käyttää mitä tahansa unaari-aakkostoa, esim. $\Sigma = \{a\}$, jolloin tehtävänä on kirjoittaa nauhalle mahdollisimman monta a-kirjainta. Vaihtoehtoisesti tehtävänä voi olla laatia mahdollisimman hidas Turingin kone, joka siis suorittaa mahdollisimman monta askelta ennen pysähtymistään. Kummassakin ongelmassa koneen täytyy kuitenkin pysähtyä lopulta. Tilojen lukumäärään n ei lasketa mukaan lopputiloja.

Tällaisen koneen laatiminen on helpompaa, kun oletamme kahteen suuntaan äärettömän työnauhan (tämä ei vaikuta ongelman ratkeavuuteen). Erityisen alku- ja loppumerkkien sijasta käytössä on tyhjä merkki '#'.

Kuvassa Busy Beaver -kone, joka pysähtyy 13 siirtymän jälkeen ja kirjoittaa 6 merkkiä:



Busy Beaver -funktio $f(n) = \{x | n$-tilainen kone kirjoittaa maksimissaan x merkkiä aloittaessaan tyhjältä nauhalta$\}$ kasvaa tavattoman nopeasti. Esim. 6-tilaisen koneen ennätys on 95,524,079 merkkiä ja kone pyörii parhaimmillaan 8,690,333,381,690,952 askeleen ajan. Funktiota ei kuitenkaan pysty laskemaan, joten tulokset on saatu kokeilemalla ja uusia ennätyksiä on odotettavissa. [5]

Post correspondence problem (PCP)

PCP-ongelman voi mallintaa dominopelinä, jossa on käytettävissä äärellinen kokoelma erilaisia dominonappuloita. Kunkin nappulan ylä- ja alareunassa on merkkijono,

[5] Lisätietoa Busy Beaverista `http://grail.cba.csuohio.edu/~somos/bb.html`

esim. $[\frac{b}{ca}], [\frac{a}{ab}], [\frac{ca}{a}], [\frac{abc}{c}]$. Kustakin nappulasta voi "taikoa" niin monta kopiota kuin ikinä haluaa. Tehtävänä on asettaa nappulat sillä tavalla peräkkäin, että ala- ja yläreunaan muodostuu sama merkkijono. Esim. edellisen kokoelman nappuloilla eräs ratkaisu olisi $[\frac{a}{ab}][\frac{b}{ca}][\frac{ca}{a}][\frac{a}{ab}][\frac{abc}{c}]$.

Yleisesti ongelma on siis seuraava: Annettuna kokoelma $P = [\frac{t_1}{s_1}], [\frac{t_2}{s_2}], ..., [\frac{t_k}{s_k}]$, onko olemassa sellaista indeksijonoa $i_1, i_2, ..., i_n$, että $s_{i_1}s_{i_2}...s_{i_n} = t_{i_1}t_{i_2}...t_{i_n}$?

Kaakelointiongelma (Tiling problem)

Kaakelointiongelman perusidea on seuraava: sinulla on käytössäsi ääretön varasto erityyppisiä kaakeleita. Kaakelityyppejä on kuitenkin vain äärellinen määrä. Ne ovat kaikki samankokoisia neliöitä, mutta niissä on niissä voi olla eri värejä tai kuvioita. Kaakelointimallissasi on määritelty, minkä tyyppisiä kaakeleita saat asettaa vierekkäin, jotta syntyy oikea kuvio. Tehtävänäsi on tutkia, voitko kaakeloida *minkätahansa kokoisen* neliönmuotoisen lattian annetun kuvioinnin mukaan. Huomaa, että kaakeleita ei saa kääntää ympäri tai kiertää mitenkään. Ohessa esimerkki kaakelointimallista, jolla kaakelointi voidaan suorittaa (Kuvat 7.15 ja kuva:kaakelit2), ja toisesta, jolla kaakelointi ei onnistu (7.17). [6]

Kuva 7.15: Näillä kolmella kaakelityypillä voit kaakeloida minkätahansa kokoisen lattian, kun vaaditaan, että kaakelien samanväriset sivut ovat aina vierekkäin.

Formaalisti määriteltynä ongelma on seuraava: Annettuna äärellinen joukko kaakelityyppejä D, vasempaan yläkulmaan sijoitettava kaakeli $d_0 \in D$, sekä kuvion määrittävät säännöt H ja V. $H, V \in D \times D$ ovat relaatioita, jotka määrittelevät, mitkä kaakelityypit saavat sijaita vierekkäin vaaka- ja pystytasossa. Esimerkiksi $(d_1, d_2) \in H$ kertoo, että tyypin d_1 kaakeli sijaitsee tyypin d_2 kaakelin vasemmalla puolella, ja $(d_1, d_3) \in V$ kertoo, että tyypin d_1 kaakeli voi sijaita tyypin d_3 kaakelin yläpuolella. Tehtävänä on löytää kuvaus $f : \mathbb{N} \times \mathbb{N} \rightarrow D$ siten että $f(0,0) = d_0$, $(f(m,n), f(m+1,n)) \in H\ \forall m,n \in \mathbb{N}$ ja $(f(m,n), f(m,n+1)) \in V\ \forall m,n \in \mathbb{N}$.

Osoittautuu, että tällaista kuvausta ei voi löytää laskennallisesti – ongelma on siis laskennallisesti ratkeamaton. Voisimme yrittää ratkaista ongelman seuraavalla ta-

[6] Kuvien esimerkit löytyvät mm. luentomateriaalista Savi Maharaj: Computability and the Halting Problem, `http://www.cs.stir.ac.uk/~sma/IT21/computability.pdf`.