Lindenmeyerin systeemit matemaattisessa mallinnuksessa

Mikko Nummelin, 44065E

31.1.2005

Sisältö

Luku 1

Johdatus aiheeseen

1.1 Suppea historiallinen katsaus

Vuonna 1968 unkarilainen biologi Aristid Lindenmeyer (1925-1989) tutki kasvien rakenteita ja päätteli, että monimutkaisuudestaan huolimatta näitä säätelevät usein verraten yksinkertaiset ja toistuvat lainalaisuudet. Jos ajatellaan vaikkapa tyypillistä puun runkoa ja siitä haarautuvia oksia, on periaatteessa jokaisen haarautumisen jälkeen edessä samantapainen rakenne, tosin vain pienempikokoisena. Myös joitakin lisäsääntöjä tarvitaan, esimerkiksi männyn rungosta kyllä haarautuu oksia joka sivulle, mutta oksista itsestään haarautuu enemmän oksia sivusuuntiin kuin ylös tai alas. Se mitä piti kehittää, oli kalkyyli, jolla kuvata näitä toistuvia rakenteita, ottaen käytännössä edellisen rakenteen parametrit ja muokaten niistä useita muunneltuja kopioita edellisen päihin. Tähän tarkoitukseen kaikkein sopivimmaksi osoittautuivat kieliopit, joissa uusia merkkijonoja johdetaan tiettyjen sääntöjen mukaan edellisistä merkkijonoista. Niillä saatiin aikaan rakenteiden toisto. Merkkijonot eivät kuitenkaan vielä ulkoisesti muistuta kasveja, joten graafinenkin tulkinta on tarpeen. Näitä tulkintatapoja on useita, joista eräs (toivottavasti tehokas) esitetään jäljempänä. Sittemmin *Lindenmeyerin systeemien* (tästä eteenpäin, L-systeemien,) tutkimus on huomattavasti laajentunut, käsittäen tärkeinä uranuurtajina mm. useita artikkeleita aiheesta kirjoitelleet Radomir Mêch:in (tšekkiläinen tekniikan tohtori) ja Przemyslaw Prusinkiewicz:in (puolalainen tekniikan tohtori ja professori Calgaryn yliopistossa). Heidän tutkimusalueitaan ovat olleet L-systeemien käytön monipuolistaminen hiomalla uusia ja jo tunnettuja tekniikoita kuten ympäristöön vaikuttaminen, samoin kuin L-systeemien ja niiden graafisen esittämisen tietokonesimulointi käyttäen apuna tunnettuja ohjelmointikieliä laajennettuina monipuolisilla graafisilla kirjastoilla kuten OpenGL:llä. Paitsi, että heidän tutkimustuloksensa ovat visuaalisessa mielessä vaikuttavia, on niillä myös erityistä merkitystä nykyaikaiselle biologialle, koska L-systeemejä on pelkän graafisen ulkomuodon kuvaamisen lisäksi kyetty käyttämään myös eliöissä tapahtuvien prosessien kuvauksiin, kuten vedenottoon, tuholaisiin ja valoon reagointiin. Tätä asiaa tutki jossain määrin itse Lindenmeyerkin alunperin. Lisäksi L-systeemit yhdistettyinä geneettisiin algoritmeihin (ehdot parhaiten täyttävien systeemien valinta useista monien peräkkäisten sukupolvien aikana) on osoittautunut hyvin tärkeäksi tekniikaksi mm. materiaalien ja rakenteiden suunnittelussa. Yksinkertainen, mutta taustalla olevalta matematiikaltaan ei kovin yksityiskohtaisesti selostettu esimerkki löytyy Richard Dawkinsin teoksesta Sokea kelloseppä [3] (engl. The Blind Watchmaker), jossa Dawkins altistaa L-systeemeihin

perustuvat graafinsa, ns. ”biomorfit” keinotekoiselle valinnalle, ohjaten kuviot muistuttamaan yhä enenevässä määrin muun muassa puita ja hyönteisiä, samoin kuin joitakin keinotekoisempia muotoja.

Seuraavissa luvuissa käydään läpi L-systeemien formalismeja useine esimerkkeineen aina kieliopeista lähtien. Tehtävänannon suppeuden vuoksi kaikkein monimutkaisimmat konstruktiot joudutaan sivuuttamaan ja pääpaino on perusasioissa.

Luku 2

Kieliopit

2.1 Kieli

Johdatuksena varsinaisiin L-systeemeihin on tarpeen esitellä *kielen* käsite. Matemaattisessa mielessä kielellä tarkoitetaan mitä tahansa määriteltyä merkkijonojoukkoa. Voitaisiin esimerkiksi määritellä kieli $\mathcal{L} = \{\ulcorner ab \urcorner, \ulcorner ra \urcorner \ulcorner yy \urcorner, \ulcorner 1z7 \urcorner\}$ yksinkertaisesti luettelemalla sen jäsenet eli *sanat* . Tyypillisesti kielen määrittely onnistuu joustavammin konstruoimalla *kielioppi*, jonka kyseisen kielen on tarkoitus toteuttaa. Kielioppeja koskevat määritelmät käydään läpi seuraavissa kappaleissa.

2.2 Katenaatio

Kieliopin tarkoituksena on ilmaista säännöt, joiden perusteella voidaan generoida kieleen kuuluvia sanoja. Yleisesti sanat koostuvat *merkeistä*, joita laitetaan peräkkäin eli katenoidaan. Kieliopin formaalia käsittelyä varten tarvitsemme apuna muutamia kieliä koskevia joukko-opillisia määritelmiä. Olkoon $\mathcal{A} = \{a_0, a_1, a_2 \ldots a_{m-1}, a_m\}$ ja $\mathcal{B} = \{b_0, b_1, b_2 \ldots b_{n-1}, b_n\}$, missä a_k:t ja b_k:t ovat yksittäisiä merkkejä tai sanoja. Tällöin $\mathcal{A}$:n ja $\mathcal{B}$:n *katenaatio* on joukko

$$\mathcal{AB} = \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} \{a_i b_j\} \tag{2.1}$$

johon siis kuuluvat kaikki mahdolliset sellaiset yhdistelmät, joissa $\mathcal{A}$:han kuuluvaan sanaan on lisätty perään joku $\mathcal{B}$:hen kuuluva sana. Jos siis vaikkapa $\mathcal{A} = \{\ulcorner 0 \urcorner, \ulcorner 1 \urcorner\}$ ja $\mathcal{B} = \{\ulcorner 2 \urcorner, \ulcorner 34 \urcorner\}$, niin $\mathcal{AB} = \{\ulcorner 02 \urcorner, \ulcorner 034 \urcorner, \ulcorner 12 \urcorner, \ulcorner 134 \urcorner\}$. Jos sanajoukko (kieli) $\mathcal{L}$ katenoidaan itsensä kanssa peräjälkeen, saadaan muodostettua sanajoukon (kielen) *katenaatiopotenssit* rekursiivisella säännöllä

$$\begin{cases} \mathcal{L}^0 &= \varepsilon \\ \mathcal{L}^{k+1} &= \mathcal{L}^k \mathcal{L} \end{cases} \tag{2.2}$$

(ε tarkoittaa edellisessä kaavassa tyhjää sanaa.) Sanajoukon (kielen) *katenaatiopotenssien joukko* määritellään

$$\mathcal{L}^* = \bigcup_{k=0}^{\infty} \mathcal{L}^k \tag{2.3}$$

Esimerkiksi yllä mainitun $\mathcal{A}$:n katenaatiopotenssien joukko $\mathcal{A}^*$ koostuu tyhjästä sanasta ε ja kaikista merkkien $\ulcorner 0 \urcorner$ ja $\ulcorner 1 \urcorner$ jonoista, mm.
$\ulcorner 01100101 \urcorner, \ulcorner 111 \urcorner, \ulcorner 0000011100011111 \urcorner \in \mathcal{A}^*$.

2.3 Konteksivapaat kieliopit

Konteksivapaa kielioppi on nelikko (V, Σ, R, S), jossa V on *aakkosto*, $\Sigma \subset V$ on *terminaalisymbolien joukko* , funktiojoukko $R \subset ((V \setminus \Sigma) \to V^*)$ on *säännöstö* ja $S \in (V \setminus \Sigma)$ on *aloitussymboli*. Jos kieli $\mathcal{L}$ määritellään kieliopilla $\mathcal{L}_G = (V, \Sigma, R, S)$, tarkoittaa se, että $\mathcal{L}$:ään kuuluvat kaikki ne merkkijonot, jotka saadaan kun aloitetaan symbolista S ja sovelletaan R:n sääntöjä peräjälkeen niin kauan, että jäljellä on vain terminaalisymboleita, s.o. jos $\ulcorner c_1 c_2 c_3 \ldots c_n \urcorner$ on S:stä R:n sääntöjä soveltamalla saatu merkkijono ja pätee $\forall k, c_k \in \Sigma$, niin $\ulcorner c_1 c_2 c_3 \ldots c_n \urcorner \in \mathcal{L}$.

2.3.1 Esimerkki ($\ulcorner a \urcorner$ ja toisiaan vastaavat sulut)

Olkoon $\mathcal{L}_G = (V, \Sigma, R, S)$, missä

$$\begin{aligned} V &= \{S, \ulcorner(\urcorner, \ulcorner)\urcorner, \ulcorner a \urcorner\} \\ \Sigma &= \{\ulcorner(\urcorner, \ulcorner)\urcorner, \ulcorner a \urcorner\} \\ R &= \{1.\{S \to \ulcorner a \urcorner S\}, 2.\{S \to \varepsilon\}, 3.\{S \to \ulcorner(\urcorner S \ulcorner)\urcorner\}\} \end{aligned} \tag{2.4}$$

Tällöin voidaan johtaa esimerkiksi

$$\begin{aligned} S &\Rightarrow_3 \ulcorner(\urcorner S \ulcorner)\urcorner \\ \ulcorner(\urcorner S \ulcorner)\urcorner &\Rightarrow_1 \ulcorner(a\urcorner S \ulcorner)\urcorner \\ \ulcorner(a\urcorner S \ulcorner)\urcorner &\Rightarrow_3 \ulcorner(a(\urcorner S \ulcorner))\urcorner \\ \ulcorner(a(\urcorner S \ulcorner))\urcorner &\Rightarrow_2 \ulcorner(a())\urcorner \end{aligned} \tag{2.5}$$

, joten $\ulcorner(a())\urcorner \in \mathcal{L}$. Toisaalta esimerkiksi $\ulcorner(((\urcorner \notin \mathcal{L}$, sillä säännöstö R selvästi edellyttää tässä (muiden edellytysten ohella) sulkeiden tasapainoa (ks. sääntö 3).

2.3.2 Konteksivapaiden kielten merkityksestä

Konteksivapaat kielet ovat tärkeässä asemassa esim. ohjelmointikielten syntaksien määrittelyssä. On osoitettu, että konteksivapaiden kielioppien kuvaamat kielet voidaan tietyin edellytyksin tunnistaa ja tulkita *pinoautomaattien* avulla. Koska tämä ei kuitenkaan liity varsinaiseen aiheeseen, suosittelen aiheesta kiinnostuneita tutustumaan [5]:een, jonka pohjalta myös yllä olevat konteksivapaan kieliopin määritelmät johdatuksineen on esitetty.

2.4 Konteksiherkät kieliopit

Konteksiherkät kieliopit poikkeavat määritelmältään konteksivapaista siten, että säännöstöön sallitaan myös sellaisia määrittelyjä, joissa vasen puoli (johdettava) ei koostu pelkästään yhdestä ei-terminaalisesta symbolista vaan voi sisältää myös muita ei-terminaalisia ja terminaalisia merkkejä eli *konteksin*, joka kuvaa sitä, missä *asiayhteydessä* kyseinen ei-terminaalinen symboli on. Formaalisti aiemmin esitelty R:ää koskeva määrite korvataan määritteellä $R \subset ((V^*(V \setminus \Sigma)V^*) \to V^*)$.

2.4.1 Esimerkki (Järjestämisalgoritmi)

Olkoon $\mathcal{L}_G = (V, \Sigma, R, S)$, missä

$$\begin{aligned} V &= \{S, H, \ulcorner 0 \urcorner, \ulcorner 1 \urcorner, \ulcorner \$ \urcorner\} \\ \Sigma &= \{\ulcorner 0 \urcorner, \ulcorner 1 \urcorner, \ulcorner \$ \urcorner\} \\ R &= \{1.\{S \to \ulcorner \$ \urcorner H \ulcorner 110 \$ \urcorner\}, 2.\{\ulcorner \$ \urcorner H \ulcorner 0 \urcorner \to \ulcorner \$ 0 \urcorner H\}, \\ &\quad 3.\{\ulcorner \$ \urcorner H \ulcorner 1 \urcorner \to \ulcorner \$ 1 \urcorner H\}, 4.\{\ulcorner 0 \urcorner H \ulcorner 0 \urcorner \to \ulcorner 00 \urcorner H\}, \\ &\quad 5.\{\ulcorner 1 \urcorner H \ulcorner 1 \urcorner \to \ulcorner 11 \urcorner H\}, 6.\{\ulcorner 0 \urcorner H \ulcorner 1 \urcorner \to \ulcorner 01 \urcorner H\}, \\ &\quad 7.\{\ulcorner 1 \urcorner H \ulcorner 0 \urcorner \to H \ulcorner 01 \urcorner\}, 8.\{H \ulcorner \$ \urcorner \to \ulcorner \$ \urcorner\}\} \end{aligned} \tag{2.6}$$

Jos esimerkin käy läpi, huomaa, että tässä tapauksessa kieleen $\mathcal{L}$ kuuluu ainoastaan sana $\ulcorner \$011\$ \urcorner$. Itse asiassa, jos sääntöä 1 muuttaa siten, että oikealla puolella on alussa $\ulcorner \$ \urcorner H$ ja perässä mikä tahansa nollista ja ykkösistä koostuva jono (päätettynä $\ulcorner \$ \urcorner$-merkillä), järjestää kyseinen kielioppi ainoaksi tuntemakseen sanaksi jonon niin, että ensin tulevat nollat ja sitten ykköset. Tämä on havaittavissa muista poikkeavasta säännöstä 7, jossa sääntö ensinnäkin vaihtaa nollan ykkösen edelle ja H siirretään oikean puolen sijasta osajonon vasemmalle puolelle valmiiksi tarvittaessa järjestämään taaempana olevia väärin päin olevia osajonoja.

2.4.2 Laskettavuudesta

Kuten edellinen esimerkki osoitti, konteksiherkkiä kielioppeja voidaan käyttää laskennallisiin tehtäviin ja algoritmien esittelyyn. Itse asiassa on osoitettu, että konteksiherkät kieliopit ovat laskentamahdollisuuksiltaan ekvivalentteja *Turingin koneiden* ja *rekursiivisten funktioiden* kanssa ja niillä voidaan periaatteessa (jos ei oteta huomioon tila- eikä aikarajoituksia) laskea kaikkea, mitä tietokoneilla on mahdollista laskea. Kiinnostuneet voivat tutustua teokseen [5].

Luku 3

L-systeemit

3.1 Relaatiot

Siirtyäksemme kieliopeista L-systeemeihin, esittelemme *relaation* käsitteen. Olkoon $\mathbb{X}$ mikä tahansa joukko, ja $f : \mathbb{X} \to \{0,1\}$. Tällöin relaatiolla f_R tarkoitetaan sitä $\mathbb{X}$:n osajoukkoa, jossa $f(x) = 1$, formaalisti $f_R = \{x \in \mathbb{X} : f(x) = 1\}$. Sanotaan myös, että f on f_R:n karakteristinen funktio. Usein on tarkoituksenmukaista vain luetella tai muuten selittää relaation toteuttavat jäsenet sen sijaan, että käytäisiin läpi karakteristista funktiota. Jos esim: $g : \{0,1\}^2 \to \{0,1\}$ siten, että $g(0,0) = 1, g(0,1) = 1, g(1,0) = 0$ ja $g(1,1) = 1$, voitaisiin helpommin todeta g_R:n olevan sellainen relaatio, johon kuuluvat kaikki muut 0:n ja 1:n järjestetyt lukuparit paitsi $(1,0)$. L-systeemien kanssa tulevat useimmiten kysymykseen vain sellaiset relaatiot, joihin kuuluu koko avaruudesta vain 'muutamia' [1] erikseen lueteltuja jäseniä, joten jatkossa relaatioita saatetaan määritellä lyhyesti vain isoilla tai pienillä kirjaimilla siten, että esim. katenaatio $R(0,0,1,0,1)R(1,1,1,0,1)$ tarkoittaa sitä, että relaatioon R kuuluvat vain $(0,0,1,0,1)$ ja $(1,1,1,0,1)$, muttei muita jäseniä, jos ei erikseen ole mainittu. Relaation R karakteristiseen funktioon viitataan vastaavasti kirjallisuudessa useimmiten symbolilla χ_R. On huomattava, että relaatio voidaan sinänsä ilmaista pelkkänä yhtä tai useampaa muuttujaa koskevana ehtona. Jos esimerkiksi $x, y \in \mathbb{R}$, voitaisiin määritellä esim: $P = \{x, y : x + 2y < 2\}$ tai käsitellä tätä relaatiota vain merkinnällä $(x + 2y < 2)$.

3.2 "Tavalliset" L-systeemit

L-systeemit ovat konteksivapaiden (ks. kappaletta 2.3) ja konteksiherkkien (ks. kappaletta 2.4) kielioppien yleistyksiä, joissa terminaaliset ja ei-terminaaliset symbolit *korvataan vastaavasti terminaalisilla ja ei-terminaalisilla relaatioilla* [2]. Tällöin voidaan muodostaa sääntöjä, joissa mm. luodaan uusia relaatioita, poistetaan vanhoja tai muokataan relaatioita suhteessa näiden edellisiin jäseniin. Vaikka konteksiherkkiä kieliop-

[1] Kuten myöhemmin käy ilmi, 'muutama' saattaa tarkoittaa jopa kymmeniä- tai satojatuhansiakin jäseniä, mutta usein ne siitä huolimatta ovat mitätön osa koko avaruudesta

[2] Joissain lähdeteoksissa pelkkiä kielioppejakin pidetään sinällään L-systeemeinä. Käytännössä tämä ei laskennallisessa mielessä tuota ongelmaa, koska on aina mahdollista muokata kielioppi $\mathcal{L}$ L-systeemiksi $\mathcal{L}^*$ siten, että vaihdetaan jokainen $V \in \mathcal{L}_G$:n symboli v relaatioksi $v^*(0)$, $V^* \in \mathcal{L}_G^*$:ssä tietysti säilyttäen vastaavuuden ei-terminaalinen/terminaalinen symboli $\to$ ei-terminaalinen/terminaalinen relaatio. Tällöin L-systeemi $\mathcal{L}^*$ on ilmeisen "ekvivalentti" alkuperäisen kieliopin kanssa.

peja koskevassa kappaleessa 2.4 todettiinkin niiden valtavat laskennalliset mahdollisuudet, L-systeemeihin siirtyminen kuitenkin yksinkertaistaa monia laskutoimituksia huomattavasti. Muuten jouduttaisiin nimittäin johtamaan monia aputekniikoita lukujen ja funktioiden koodaamiseksi mukaan kielioppeihin. Se, miten tällaiset koodaukset ovat mahdollisia, käy ilmi teoksesta [5] johdettaessa Turingin koneita , kielioppeja ja rekursiivisia funktioita koskevia lauseita. Tässä kuitenkin sivuutamme nämä todistukset, koska ne ovat suhteellisen pitkiä ja työläitä eivätkä liity varsinaiseen aiheeseemme. Eräs tärkeä kysymys, joka nousee esiin, on relaatioiden järjestys. Kuten näimme, tällä saattaa olla merkitystä, etenkin mikäli osa L-systeemistä on kielioppi (tai sitä vastaava rakenne), jonka tarkoitus on generoida nimenomaan tiettyjä sanoja mukaan tai mikäli relaatiot halutaan saada juuri tiettyyn järjestykseen tietokoneella analysointia varten. Toisaalta sääntöjen vasemmalle puolelle saattaa osaksi konteksia tulla joitakin muuttujista riippuvia relaatioita (esim. $(n \geq 3)$), joiden paikalla ei voi katsoa olevan samalla tavoin väliä. Jatkoa varten tehdään siksi muutamia sopimuksia:

- Relaatiojoukossa V (ks. aiempaa kieliopin määritelmää) esitellyt terminaaliset ja ei-terminaaliset relaatiot pidetään siinä järjestyksessä kuin säännöt ne katenoivat, ellei toisin mainita tai elleivät säännöt mahdollista niiden kääntämistä muuhun järjestykseen. Jos tällaisia relaatioita käsitellään säännöstön johdettavina (vasemmalla puolella), niiden järjestys ratkaisee, saako tiettyä sääntöä soveltaa.
- Relaatioissa olevien muuttujien avulla generoidut väliaikaiset (nämä ovat myös aina ei-terminaalisia) relaatiot esitellään johdettavassa relaatiojoukon V jäsenten *oikeanpuolisessa* konteksissa. Jos näitä esiintyy, *tulkitaan katenaatio niiden osalta samaksi asiaksi kuin konjunktio*, s.o. $H(0, n)(n \neq 0) \rightarrow \ldots$ tarkoittaa, että johtamissäännön käyttö on sallittua vain jos *kaikki* kyseisessä johdettavassa olevat relaatiot ovat voimassa (eli tosia).

3.2.1 Esimerkki (2:n potenssit)

Olkoon $\mathcal{L}_G = (V, \Sigma, R, S)$, missä

$$\begin{aligned} V &= \{S(n), P(x, n), p(x)\} \\ \Sigma &= \{p(x)\} \\ R &= \{\{S(n) \rightarrow P(1, n)\} \\ &\quad \{P(x, n)(n > 0) \rightarrow P(2x, n - 1)\} \\ &\quad \{P(x, 0) \rightarrow p(x)\}\} \end{aligned} \tag{3.1}$$

Tämä L-systeemi generoi 2:n potenssit siten, että S:n argumenttia käytetään eksponenttina, ja lopputulos on relaatio p, johon kuuluu ainoastaan kyseinen 2:n potensseista. Eli kun $n \in \mathbb{N}$, niin voidaan johtaa

$$S(n) \Longrightarrow_{\mathcal{L}_G} p(2^n). \tag{3.2}$$

3.2.2 Esimerkki (Jänisjono)

Seuraavan kuuluisan esimerkin nimi on *jänisjono* (engl. *rabbit sequence*), ks. [9]. Se määritellään formaalisti siten, että $\mathcal{L}_G = (V, \Sigma, R, S)$, missä

$$V = \{S(n), M(n), \ulcorner 0 \urcorner, \ulcorner 1 \urcorner\}$$

n	jono
0	$\ulcorner 0 \urcorner$
1	$\ulcorner 1 \urcorner$
2	$\ulcorner 10 \urcorner$
3	$\ulcorner 101 \urcorner$
4	$\ulcorner 10110 \urcorner$
5	$\ulcorner 10110101 \urcorner$

Kuva 3.1: ”Jänisjonoja”

$$\begin{aligned} \Sigma &= \{\ulcorner 0 \urcorner, \ulcorner 1 \urcorner\}\} \\ R &= \{\{S(n)(n>0) \rightarrow M(n-1)\} \\ &\quad \{M(n)(n>0) \rightarrow M(n-1)S(n-1)\} \\ &\quad \{S(0) \rightarrow \ulcorner 0 \urcorner\} \\ &\quad \{M(0) \rightarrow \ulcorner 1 \urcorner\}\} \end{aligned} \tag{3.3}$$

Jonon nimi tulee siitä, että se alkaa yhdestä nuoresta jäniksestä $S(\ldots)$, joka kasvaa aikuiseksi $M(\ldots)$, joka lisääntyy $M(\ldots)S(\ldots)$ tuottaen lopulta (kun iteraatiokierrokset on käyty loppuun), jonon nollia (viimeisen kierroksen nuoret jänikset) ja ykkösiä (aiemmat aikuiset jänikset). Jonolla on todistettu olevan tiettyjä yhteyksiä *Fibonaccin lukuihin* ja *päättymättömiin jakolaskuihin*, mutta jälkimmäinen ominaisuus sivuutetaan tässä. Useat lähteet määrittelevät jänisjonon nimenomaan sellaisena jonona, että em. prosessia jatketaan äärettömiin, s.o. $\lim_{n\rightarrow\infty} S(n) \Longrightarrow_{\mathcal{L}_G} \ldots$, mutta tässä tarkasteltavat L-systeemit on usein konstruoitu käsittämään lopetusehdot, jotta tietokonesimulointi näiden tarkastelujen pohjalta olisi yksinkertaisempaa. Kuvassa 3.1 on taulukoitu muutamia jonoja pienillä n:n arvoilla. Kuten jonosta on nähtävissä, on kullakin askeleella jonon pituus $\mathrm{Fib}(n)$. [3]

3.3 Graafit

Graafi on pari (V, E), jossa V on graafin kärkien joukko ja E sivujen eli kaarien joukko. Kukin sivu yhdistää suunnistetusti kaksi graafin kärkeä, formaalisti $e = (p, q)$, missä $e \in E$ ja $p.q \in V$. L-systeemien visualisoinnissa graafin käsitettä käytetään yleisimmin niin, että sen kärjet kuuluvat johonkin sisätuloavaruuteen, tyypillisesti $\mathbb{R}^2$:een tai $\mathbb{R}^3$:een. Muuta tietoa graafeista yleensäkin, ks. [11].

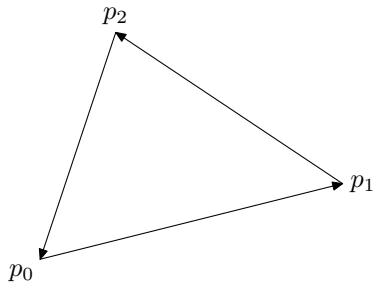
3.3.1 Esimerkki (Kolmio)

Olkoon määritelty graafi $G = (V, E)$ seuraavasti, huom: $p_i \in \mathbb{R}^2$:

$$\begin{aligned} V &= \{p_0 : (0,0), p_1 : (\frac{4}{5}, \frac{1}{5}), p_2 : (\frac{1}{5}, \frac{3}{5})\} \\ E &= \{(p_0, p_1), (p_1, p_2), (p_2, p_0)\} \end{aligned} \tag{3.4}$$

. Tämä on eräs $\mathbb{R}^2$:n suunnistettu graafi, joka ulkomuodoltaan on vastapäivään suunnistetun kolmion näköinen, ks. kuvaa 3.2.

[3] Fibonaccin luvut määritellään rekursiivisesti, $\mathrm{Fib}(0) = \mathrm{Fib}(1) = 1$ ja $\mathrm{Fib}(n) = \mathrm{Fib}(n-1) + \mathrm{Fib}(n-2)$, kun $n \geq 2$.



Kuva 3.2: Kolmionmuotoinen graafi

3.4 Relaatioista graafin osiksi

Kuten edempänä nähtiin, L-systeemit laskevat asioita ottamalla käyttöön kullakin hetkellä systeemissä olevat relaatiojonot ja korvaavat ne systeemissä mainittujen sääntöjen mukaan uusilla. Seuraavassa tarkastelemme eri tapoja, jolla relaatiot voidaan rinnastaa graafin kärkiin ja sivuihin niin, että kun L-systeemi on käynyt laskutehtävänsä loppuun, voidaan lopputulos esittää graafina lopputuloksessa olevien terminaalisten relaatioiden perusteella. Huomattakoon, että esitystapoja on todellakin useita ja se, millainen esitystapa valitaan, riippuu eniten siitä, kuinka vaativasta kuvauksesta on kysymys. Jos esimerkiksi tiedetään, että vaikkapa jotakin fraktaalipuuta generoidessa käytetään vain rajoitetusti $\mathbb{R}^2$:n lineaarikuvauksia, ei tietenkään kannata käyttää vaikkapa sellaista koodaustapaa, joka olisi suunniteltu $\mathbb{C}^5$:lle ja sisältäisi hyvin monimutkaisua avaruuden osien manipulointimenetelmiä.

3.4.1 Eräs yksinkertainen $\mathbb{R}^2$:n menetelmä

Käytetään graafin sivun vastineena tyyppiä $\rho(x_0, y_0, l, \varphi)$ olevia relaatioita, jolloin x_0 ja y_0 kuvaavat sivun alkukärjen koordinaatteja ja loppukärjen koordinaatit ovat vastaavasti

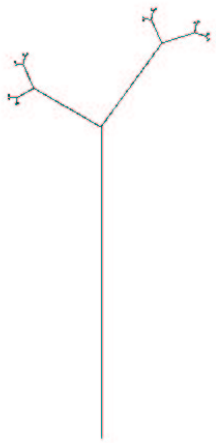
$$\begin{cases} x_1 &= x_0 + l\cos\varphi \\ y_1 &= y_0 + l\sin\varphi \end{cases} \tag{3.5}$$

Kuten helposti nähdään, kuvaa l tuolloin sivun pituutta ja φ suuntakulmaa.

3.4.2 Binaaripuu

Olkoon $\mathcal{L}_G = (V, \Sigma, R, S)$, missä

$$\begin{aligned} (V \setminus \Sigma) &= \{P(x, y, l, \varphi, n), S(n)\} \\ (V \cap \Sigma) &= \{\rho(x, y, l, \varphi)\} \\ R &= \{1.\{S(n) \to P(0, 0, 1, \frac{\pi}{2}, n)\}, \\ &\quad 2.\{P(x, y, l, \varphi, n)(n > 0) \to \\ &\quad P(x, y, l, \varphi, 0) \\ &\quad P(x + l\cos\varphi, y + l\sin\varphi, l/3, \varphi + \frac{\pi}{5}, n - 1) \\ &\quad P(x + l\cos\varphi, y + l\sin\varphi, l/4, \varphi - \frac{\pi}{3}, n - 1)\}, \end{aligned}$$



Kuva 3.3: Binaaripuu

$$3.\{P(x,y,l,\varphi,0) \rightarrow \rho(x,y,l,\varphi)\}\} \tag{3.6}$$

Alkuparametri n on haluttu iteraation syvyys. Seuraava lyhyt C-kielinen [10] ohjelma tulostaa tämän graafin sivujen alku- ja loppukoordinaatit:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double pii;
int iteraatiot;

void P(double x,double y,double l,double phi,int n) {
    if(n>=0) {
        printf("(%1.5lf,%1.5lf)->(%1.5lf,%1.5lf)",
               x,y,x+l*cos(phi),y+l*sin(phi));
        if(n==0)
            printf("*\n");
        else
            printf("-\n");
        P(x+l*cos(phi),y+l*sin(phi),l/3,phi+pii/5,n-1);
        P(x+l*cos(phi),y+l*sin(phi),l/4,phi-pii/3,n-1);
    }
}

int main(int argc, char *argv[]) {
    pii=2*acos(0);
    if(argc>1)
        iteraatiot=atoi(argv[1]);
    else
        iteraatiot=3;
    P(0,0,1,pii/2,iteraatiot);
    return 0;
}
```

Se käännetään komennolla

```
$ gcc -lm bintree.c -o bintree
```

(tietysti olettaen, että ohjelmakoodi tallennettiin nimellä ”bintree.c”) ja ajetaan komennolla

```
$ ./bintree 4
```

Ohjelmalle syötettävä luku on iteroinnin syvyys. Oletusarvo tälle on 3. Ohjelma on laitettu tähän esimerkkinä siitä, miten suoraviivaisesti voidaan usein matemaattisesti suunniteltu L-systeemi siirtää tietokoneohjelmiin. Yksinkertaisena ja lyhyenä pitämisen vuoksi ei tähän esimerkkikoodiin ollut mahdollista laittaa mukaan tietokonegrafiikan piirtoa ikkunaan tai kuvatiedostoon. Tilan säästämiseksi muut ohjelmakoodit (kuten myös edellinenkin) löytyvät liitteistä [12].

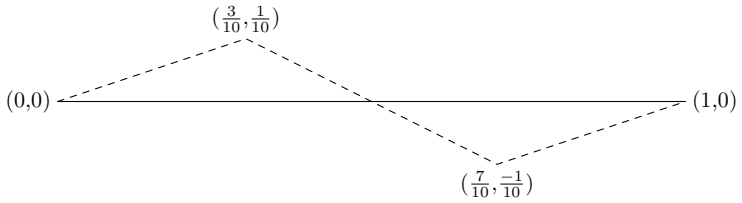
3.4.3 Eräs fraktaalinen $\mathbb{R}^2$:n käyrä

Tässä esimerkissä käytämme toisenlaista $\mathbb{R}^2$:n vektoreiden koodausmenetelmää kuin edellisessä. Pituus- ja suuntakulmaparametrit korvataan tässä vektorin x- ja y-suuntaisilla komponenteilla. Relaatiota $\rho_2(x_0, y_0, \Delta x, \Delta y)$ vastaavat siis sellaiset graafin sivut, joiden alkukärki on (x_0, y_0) ja loppukärki $(x_0 + \Delta x, y_0 + \Delta y)$. Olkoon siis $\mathcal{L}_G = (V, \Sigma, R, S)$, missä

$$
\begin{aligned}
(V \setminus \Sigma) &= \{Q(x, y, \Delta x, \Delta y, n), S(n)\} \\
(V \cap \Sigma) &= \{\rho_2(x, y, \Delta x, \Delta y)\} \\
R &= \{1.\{S(n) \to Q(0, 0, 1, 0, n)\}, \\
&\quad 2.\{Q(x, y, \Delta x, \Delta y, n)(n > 0) \to \\
&\quad Q(x, y, \frac{3}{10}\Delta x - \frac{1}{10}\Delta y, \frac{1}{10}\Delta x + \frac{3}{10}\Delta y, n - 1) \\
&\quad Q(x + \frac{3}{10}\Delta x - \frac{1}{10}\Delta y, y + \frac{1}{10}\Delta x + \frac{3}{10}\Delta y, \\
&\quad \frac{2}{5}\Delta x + \frac{1}{5}\Delta y, -\frac{1}{5}\Delta x + \frac{2}{5}\Delta y, n - 1) \\
&\quad Q(x + \frac{7}{10}\Delta x + \frac{1}{10}\Delta y, y - \frac{1}{10}\Delta x + \frac{7}{10}\Delta y, \\
&\quad \frac{3}{10}\Delta x - \frac{1}{10}\Delta y, \frac{1}{10}\Delta x + \frac{3}{10}\Delta y, n - 1)\}, \\
&\quad 3.\{Q(x, y, \Delta x, \Delta y, 0) \to \rho_2(x, y, \Delta x, \Delta y)\}\}
\end{aligned}
\tag{3.7}
$$

On erikoisesti huomattava, että tässä esimerkissä (toisin kuin binaaripuussa), jokainen relaatio *ei tule* vastaamaan lopullisessa graafissa mitään sivua, vaan tässä aidosti korvataan jokainen ”edellisellä kierroksella” mukana ollut sivu kolmella uudella sivulla (kuva 3.4), jotka menevät ”siksakkia” aiemman sivun molemmin puolin. Herää kysymys, miten em. L-systeemissä säännöt on laadittu, jotta se vastaisi juuri näitä korvaavuussääntöjä. Ideana on ollut se, että kun korvattava viivanpätkä käsitetään vektorina $(\Delta x, \Delta y)$, voidaan todeta $(-\Delta y, \Delta x)$:n olevan tähän nähden vastapäivään kohtisuorassa [4] olevan samanpituisen vektorin. Näistä voidaan muodostaa kanta, jossa lausua uusien kolmen vektorin koordinaattipisteet. Esimerkiksi ensimmäisen katkoviivan päätepistettä vastaa selvästi $\frac{3}{10}(\Delta x, \Delta y) + \frac{1}{10}(-\Delta y, \Delta x) = (\frac{3}{10}\Delta x - \frac{1}{10}\Delta y, \frac{1}{10}\Delta x +$

[4] $(\Delta x, \Delta y) \cdot (-\Delta y, \Delta x) = -\Delta x \Delta y + \Delta x \Delta y = 0$



Kuva 3.4: Kukin viivanpätkä (yhtenäinen viiva) korvataan kolmella uudella viivanpalalla (katkonaiset viivat)

Kuva 3.5: Fraktaalinen käyrä

$\frac{3}{10}\Delta y)$. Tämä ja muut vastaavat tulokset on sitten sijoitettu L-systeemiin uusien johdettujen $Q(\ldots)$-relaatioiden parametreiksi. Kuten toisaalta kuvasta 3.5 näkyy, $\lim_{n\to\infty} S(n) \implies_{\mathcal{L}_G}$ on fraktaalinen käyrä, jolla on sellainenkin mielenkiintoinen ominaisuus, että se on ”moniulotteisempi” kuin pelkkä viiva (joka on 1-ulotteinen), mutta kuitenkin ”vähempiulotteinen” kuin koko taso (joka on 2-ulotteinen). Dimensioksi voidaan karkeasti arvioida

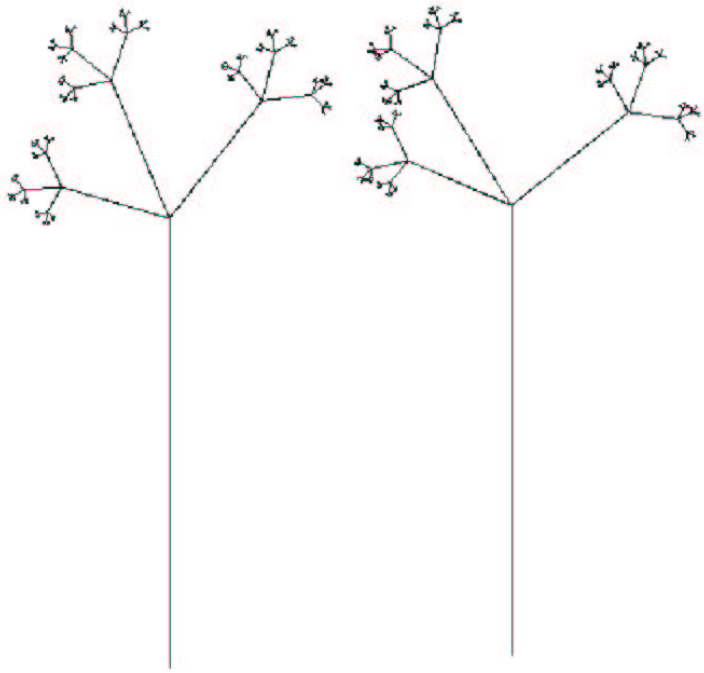
$$\frac{\ln\left(3\left(\frac{2}{\sqrt{10}}+\frac{1}{\sqrt{5}}\right)\right)}{\ln(3)} = 1 + \frac{\ln\left(\frac{2}{\sqrt{10}}+\frac{1}{\sqrt{5}}\right)}{\ln(3)} \approx 1.08 \tag{3.8}$$

Tulos on likimääräinen, koska tätä laskettaessa tehtiin yksinkertaistava oletus, että viiva olisi jaettu kullakin iteraatiokierroksella kolmeen *yhtä pitkään* osaan, mikä ei kuitenkaan ollut tarkkaan ottaen asianlaita. Asiasta enemmän kiinnostuneet voivat tutustua *Hausdorff-dimension* määritelmään verkkosivulla [7] tai seikkaperäisemmin kirjassa [6].

3.5 Satunnaisluvut L-systeemeissä

Joissakin tietokonemallinnuksissa on olennaista tutkia, minkä näköisiä erilaisia malleja tietyille satunnaisilmiöille voi olla, kun satunnaiskoe toistetaan useampia kertoja. Tällaisia mallinnuksia kutsutaan yleisesti *Monte-Carlo-menetelmiksi* [8] . Jotta satunnaislukuja voitaisiin ottaa mukaan L-systeemeihin, tarvitaan avuksi satunnaislukuarvoinen funktio. Määritellään funktio $\mathrm{rand} : \emptyset \to [0,1]$ siten, että palautettujen lukujen jakauma on tasainen, s.o. $\mathrm{rand}()$:illa on tiheysfunktio $\rho : \mathbb{R} \to \mathbb{R}$ seuraavasti:

$$\rho(x) = \begin{cases} 1 & , \quad 0 \le x \le 1 \\ 0 & , \quad (x < 0) \vee (x > 1) \end{cases} \tag{3.9}$$



Kuva 3.6: Ternaaripuita, joiden haaraumakulmat ovat satunnaismuuttujia

Tällöin voidaan lausua esimerkiksi $\varphi = \frac{\pi}{3} + \frac{\pi}{6}\mathrm{rand}()$, jolloin tarkoitetaan kulmaa, joka poimitaan satunnaisesti väliltä $[\frac{\pi}{3}, \frac{\pi}{2}]$. Ternaaripuut (kuva 3.6) on saatu aikaan L-systeemillä, joka muuten on samanlainen kuin binaaripuussa (3.6), mutta jossa sääntö 2 onkin seuraavanlainen:

$$
\begin{aligned}
&2.\{P(x,y,l,\varphi,n)(n>0) \rightarrow \\
&P(x,y,l,\varphi,0) \\
&P(x+l\cos\varphi, y+l\sin\varphi, l/3, \varphi+\frac{\pi}{5}+\frac{\pi}{10}\mathrm{rand}(), n-1) \\
&P(x+l\cos\varphi, y+l\sin\varphi, l/3, \varphi-\frac{\pi}{5}+\frac{\pi}{10}\mathrm{rand}(), n-1) \\
&P(x+l\cos\varphi, y+l\sin\varphi, l/4, \varphi-\frac{\pi}{2}+\frac{\pi}{6}\mathrm{rand}(), n-1)\}
\end{aligned}
\tag{3.10}
$$

3.6 $\mathbb{R}^3$:n L-systeemit

3.6.1 Johdanto

Monet L-systeemit simuloivat kolmiulotteisia malleja. Tällöin on periaatteessa mahdollista toimia samojen perussääntöjen mukaan kuin aiemmissakin esimerkeissä; kirjoitetaan systeemi, jossa ei-terminaalisia relaatioita korvataan toisilla, kunnes lopuksi päädytään sellaisiin objekteihin, jotka on tulkittavissa $\mathbb{R}^3$:n graafien sivuiksi. Pääasiallisina vaikeuksina ovat kysymykset sivujen suunnistussäännöistä sekä systeemiin

kirjoitettavien kaavojen monimutkaistuminen. Käytännössä suunnistusongelma vaatii käyttämään ei-terminaalisissa relaatioissa enemmän tietoa kuin $\mathbb{R}^2$:n L-systeemien tapauksessa, tyypillisesti jonkinlainen menetelmä pää- ja sivunormaalivektoreiden määrittämiseksi on tarpeen. Reaalimaailman vastineena tälle voisi ajatella, mitä tarkoittavat missäkin tilanteessa termit ”ylös/alas”, ”vasemmalle/oikealle” ja ”eteen/taakse” . Kaavojen monimutkaistumisen välttämiseksi tulee harkita, millaisia operaatioita konstruoitavassa L-systeemissä tarvitaan. Esimerkiksi mielivaltaiset $\mathbb{R}^3$:n lineaarikuvaukset tarvitsevat kaksitoista parametria, joista on 9 kpl 3×3-matriisiin kiertoa ja venytystä varten ja 3 kpl siirtovektoria varten; lisäksi näiden parametrien arvojen laskeminen aiemmista parametreista saattaa joissain tapauksissa olla työlästä. Monissa tapauksissa riittää järjestää uudet sivut johdettaviksi vanhojen päätepisteistä ja käyttää operaatioina ainoastaan tasaista skaalausta joka suunnassa ja kiertoja pelkästään edellisen vaiheen tangentti-, pää- ja sivunormaalivektoreiden suhteen.

3.6.2 Kolmikanta

$\mathbb{R}^3$:n L-systeemeissä on siis, käsitellessä ei-terminaalisia relaatioita, tarpeen koodaustapa, joka kertoo sekä vektorin suunnan, pituuden että normaalien suunnat. Tätä varten esitellään pari

$$p = \{\mathbf{t}, \mathbf{n}\} \tag{3.11}$$

jossa $\mathbf{t}$ on *tangenttivektori* ja $\mathbf{n}$ on *päänormaalivektori*. $\mathbf{t}$ ja $\mathbf{n}$ ovat yksikkövektoreita, s.o. $\|\mathbf{t}\| = \|\mathbf{n}\| = 1$ ja kohtisuorassa toisiaan vasten s.o. $\mathbf{t} \cdot \mathbf{n} = 0$. $\mathbf{t}$:stä ja $\mathbf{n}$:stä on sitten vielä ristitulon avulla mahdollista muodostaa *sivunormaalivektori* $\mathbf{b} = \mathbf{t} \times \mathbf{n}$, joka sekin on ristitulon sääntöjen nojalla yksikkövektori ja kohtisuorassa $\mathbf{t}$:tä ja $\mathbf{n}$:ää vastaan.

3.6.3 Kierto-operaattorit kolmikannan suhteen

Kierrettäessä p:n vektoreita kolmikannan vektoreiden suhteen, jätetään se kolmikannan komponenteista muuttamatta, jonka suhteen kierto tapahtuu, mutta muut kaksi lasketaan aiempien kombinaatioina käyttäen kertoimina trigonometrisia funktioita. Saadaan

$$\begin{aligned}
\mathrm{rot}_{\mathbf{b}}(p, \varphi) &= \{\cos\varphi\,\mathbf{t} + \sin\varphi\,\mathbf{n}, \cos\varphi\,\mathbf{n} - \sin\varphi\,\mathbf{t}\} \\
\mathrm{rot}_{\mathbf{t}}(p, \varphi) &= \{\mathbf{t}, \sin\varphi\,\mathbf{b} + \cos\varphi\,\mathbf{n}\} \\
\mathrm{rot}_{\mathbf{n}}(p, \varphi) &= \{\cos\varphi\,\mathbf{t} - \sin\varphi\,\mathbf{b}, \mathbf{n}\}
\end{aligned} \tag{3.12}$$

Lisäksi on huomattava, että $\mathbf{b}$:n arvoa ei laskettu erikseen lainkaan, koska se joka tapauksessa saadaan ristitulosta. Joissakin tapauksissa on tarpeen laskea peräkkäin useampi kierto samassa $\{\mathbf{t}, \mathbf{n}, \mathbf{b}\}$-kannassa. Tällöin voidaan em. kaavoja soveltaa sopiviin kolmikannan vektorien summalausekkeisiin. Esimerkiksi kierto $\mathbf{b}$:n ympäri saadaan matriisilaskusta

$$\begin{aligned}
\mathrm{rot}_{\mathbf{b}}(\alpha_1\mathbf{t} + \alpha_2\mathbf{n} + \alpha_3\mathbf{b}, \varphi) &= \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \\
&= \begin{pmatrix} \alpha_1\cos\varphi - \alpha_2\sin\varphi \\ \alpha_1\sin\varphi + \alpha_2\cos\varphi \\ \alpha_3 \end{pmatrix} \\
&= (\alpha_1\cos\varphi - \alpha_2\sin\varphi)\mathbf{t} + (\alpha_1\sin\varphi + \alpha_2\cos\varphi)\mathbf{n} + \alpha_3\mathbf{b}
\end{aligned} \tag{3.13}$$

Muut kaavat saadaan vastaavasti niin, että kun kierretään $\mathbf{t}$:n suhteen, vaihdetaan vektorien paikkoja edellisessä kaavassa tällä tavoin

$$\{\mathbf{t} \rightarrow \mathbf{n}, \mathbf{n} \rightarrow \mathbf{b}, \mathbf{b} \rightarrow \mathbf{t}\} \tag{3.14}$$

ja kun kierretään $\mathbf{n}$:n suhteen, edelleen

$$\{\mathbf{t} \rightarrow \mathbf{b}, \mathbf{n} \rightarrow \mathbf{t}, \mathbf{b} \rightarrow \mathbf{n}\}. \tag{3.15}$$

3.6.4 Kolmikannan virheiden korjaus

Joskus hyvin suurissa L-systeemeissä edellä mainittujen rotaatiokaavojen hyvin moneen kertaan ajaminen tietokoneella aiheuttaa kolmikantaan kasautuvia pyöristysvirheitä. Nämä tulevat ilmi lähinnä kahdella tavalla, vektorit eivät enää ole yksikkövektoreita eivätkä ne ole kohtisuorassa toisiaan vastaan. Kun tuloksia katsotaan lopullisessa graafissa, näkyvät virheet skaalan ja kulmien poikkeamina ja kuvioiden ”venymisinä”. Käytännössä on kuitenkin mahdollisuus toimittaa L-systeemiä simuloivassa ohjelmassa korjauslaskuja seuraavasti: Olkoon pari $p_e = \{\mathbf{t}_s, \mathbf{n}_e\}$ vääristynyt siten, että $\mathbf{t}_s$ on muuntunut väärän mittaiseksi, mutta sitä voidaan kuitenkin pitää oikean suuntaisena, kun taas $\mathbf{n}_e$ on sekä suunnaltaan että pituudeltaan virheellinen. Käytännössä kannattaa aloittaa siitä, että $\mathbf{t}$ tulee normeeratuksi yksikkövektoriksi kaavalla

$$\mathbf{t} = \frac{\mathbf{t}_s}{\|\mathbf{t}_s\|} \tag{3.16}$$

Sen jälkeen saadaan $\mathbf{n}$ korjatuksi seuraavilla kahdella kaavalla:

$$\begin{aligned} \mathbf{n}_s &= \mathbf{n}_e - \frac{\mathbf{t} \cdot \mathbf{n}_e}{\|\mathbf{t}\|^2}\mathbf{t} \\ &= \mathbf{n}_e - (\mathbf{t} \cdot \mathbf{n}_e)\mathbf{t} \\ \mathbf{n} &= \frac{\mathbf{n}_s}{\|\mathbf{n}_s\|} \end{aligned} \tag{3.17}$$

Ensinmainittu korjaa suunnan poistamalla $\mathbf{n}_e$:stä virheellisen, $\mathbf{t}$:n suuntaisen komponentin [5], toinen normeeraa tuloksen yksikkövektoriksi. $\mathbf{b}$ ei vaadi korjauslaskua, koska se lasketaan ristitulolla $\mathbf{t}$:n ja $\mathbf{n}$:n korjausten jälkeen.

3.6.5 Kolmiulotteinen puu

Kehitetään edellä esitettyihin kaavoihin ja tekniikoihin perustuen L-systeemi kolmiulotteiselle puulle. Aloitetaan puun ”rakentaminen” rungonpätkästä

$$S(m) \rightarrow \mathrm{Runko}(\mathbf{s}_0, \mathbf{t}_0, \mathbf{n}_0, l_0, m) \tag{3.18}$$

missä alkuparametrit olkoon $\mathbf{s}_0 = (0,0,0)$, $\mathbf{t}_0 = (0,0,1)$, $\mathbf{n}_0 = (-1,0,0)$ ja $l_0 = 1$. Alkupätkä on siis origoon pystyyn asetettu ”tukki”, jonka pituus on 1 ja jolla on päänormaali vasemmalle päin, sivunormaalin ollessa ristitulon oikeakätisyyden perusteella negatiivisen y-akselin suuntaan eli kuvassa suunnilleen ”katsojaan päin”. Kunkin rungonpätkän

[5] $\|\mathbf{t}\|^2$ supistuu pois nimittäjästä, koska $\mathbf{t}$ on jo normeerattu yksikkövektoriksi.

Kuva 3.7: Kolmiulotteinen fraktaalinen puu

perusteella kasvatetaan sen päästä uusi hieman lyhyempi rungonpätkä, neljä viistosti ylös sivusuuntiin suuntautuvaa oksaa

$$\begin{aligned}
\mathrm{Runko}(\mathbf{s},\mathbf{t},\mathbf{n},l,m)(m>0) \quad \rightarrow \quad & \mathrm{Runko}(\mathbf{s},\mathbf{t},\mathbf{n},l,m-1) \\
& \mathrm{Runko}(\mathbf{s}+l\mathbf{t},\mathbf{t},\mathbf{n},0.8l,m-1) \\
& \mathrm{Oksa}(\mathbf{s}+l\mathbf{t},\cos\frac{5\pi}{12}\mathbf{t}-\sin\frac{5\pi}{12}\mathbf{b},\mathbf{n},0.6l,m-1) \\
& \mathrm{Oksa}(\mathbf{s}+l\mathbf{t},\cos\frac{5\pi}{12}\mathbf{t}+\sin\frac{5\pi}{12}\mathbf{n},\mathbf{b},0.6l,m-1) \\
& \mathrm{Oksa}(\mathbf{s}+l\mathbf{t},\cos\frac{5\pi}{12}\mathbf{t}+\sin\frac{5\pi}{12}\mathbf{b},-\mathbf{n},0.6l,m-1) \\
& \mathrm{Oksa}(\mathbf{s}+l\mathbf{t},\cos\frac{5\pi}{12}\mathbf{t}-\sin\frac{5\pi}{12}\mathbf{n},-\mathbf{b}, \\
& 0.6l,m-1) \qquad (3.19)
\end{aligned}$$

ja kustakin oksanpätkästä yksi eteenpäintyöntyvä ja kaksi sivullepäin menevää oksaa

$$\begin{aligned}
\mathrm{Oksa}(\mathbf{s},\mathbf{t},\mathbf{n},l,m)(m>0) \quad \rightarrow \quad & \mathrm{Oksa}(\mathbf{s},\mathbf{t},\mathbf{n},l,m-1) \\
& \mathrm{Oksa}(\mathbf{s}+l\mathbf{t},\cos\frac{\pi}{12}\mathbf{t}+\sin\frac{\pi}{12}\mathbf{b},\mathbf{n},0.5l,m-1) \\
& \mathrm{Oksa}(\mathbf{s}+l\mathbf{t},\cos\frac{\pi}{12}\mathbf{n}+\sin\frac{\pi}{12}\mathbf{b},-\mathbf{t},0.5l,m-1) \\
& \mathrm{Oksa}(\mathbf{s}+l\mathbf{t},-\cos\frac{\pi}{12}\mathbf{n}+\sin\frac{\pi}{12}\mathbf{b},\mathbf{t}, \\
& 0.5l,m-1). \qquad (3.20)
\end{aligned}$$

Jos $m=0$, niin lopetetaan kasvu ja laaditaan graafin sivut johtamalla näitä vastaavat terminaaliset relaatiot. Tietokoneohjelmassa näiden johtosääntöjen tilalla saattaa olla suoraan grafiikanpiirtokäskyjä.

$$\begin{aligned}
\mathrm{Runko}(\mathbf{s},\mathbf{t},\mathbf{n},l,0) \quad &\rightarrow \quad \rho(\mathbf{s},l\mathbf{t}) \\
\mathrm{Oksa}(\mathbf{s},\mathbf{t},\mathbf{n},l,0) \quad &\rightarrow \quad \rho(\mathbf{s},l\mathbf{t}) \qquad (3.21)
\end{aligned}$$

Tässä ρ:n ensimmäinen argumentti on alkukärki, toinen taas loppukärjen ja alkukärjen erotus. Tuloksena on kuvan 3.7 mukainen fraktaalinen puu.

3.7 Interaktiiviset L-systeemit

3.7.1 Johdanto

Edellä esitetyissä esimerkeissä laadimme L-systeemejä toistamalla mekaanisesti yhä uudelleen sääntöjä, joilla relaatioita korvattiin uusilla, aina lopetusehtoon saakka. Lisäksi totesimme, että tällaiset ”staattiset” L-systeemit ovat kohtalaisen vähällä vaivalla toteutettavissa itseään uudelleen kutsuvilla ohjelmanpätkillä. Näin ei kuitenkaan enää ole asianlaita kun ajattelemme reaalimaailman ilmiöitä tarkemmin - esimerkiksi kasvit joutuvat tositilanteissa kamppailemaan niin valosta, ravinteista kuin elintilastakin (ks. [1]). Voidaan siis sanoa, että kun tiettyä L-systeemin relaatiota ollaan korvaavuussääntöjen perusteella vaihtamassa uusiin, on tavanomaisen konteksin lisäksi aihetta kiinnittää huomiota myös siihen ajankohtaan, milloin laskenta tapahtuu ja siihen, millainen on ympäristön tila sillä hetkellä. Kun korvaavuussääntöä sitten sovelletaan, on tärkeää päivittää ympäristön tila uutta tilannetta vastaavaksi. Asian ymmärtää kun ajattelee vaikkapa kahta rinnakkain kasvavaa lehtipuuta ja mitä etua toiselle niistä on saada ensin kasvatettua latvustoaan toisen yläpuolelle, samalla varjostaen kilpailijaansa ja kaventamalla sen elintilaa. Todennäköisesti tässä kilpailussa häviölle jäänyt puu jäisi lopullisestikin lyhyemmäksi ja kitukasvuisemmaksi kuin ”voittajansa”.

3.7.2 Ympäristö

Ympäristöä voidaan kuvata usein eri tavoin, riippuen siitä, millaista dataa halutaan tallettaa ja käyttää hyväksi. Joskus on tarpeellista pitää laajaa matriisia koko siitä vektoriavaruudesta, johon L-systeemin graafinen esitys on tarkoitus kuvata, joskus taas riittää harva matriisi, joka kuvaa vain joidenkin tämän avaruuden pisteiden tiloja. Tyypillisesti ympäristöä kuvaavat muuttujat joka tapauksessa käsittävät varsin paljon dataa, joten ei ole tarkoituksenmukaista kuvata niitä jokaisessa L-systeemin johtamissäännössä. Vaihtoehtoinen tapa on ennen L-systeemin esittelyä sopia jostakin tavasta, jolla kuvataan vain näiden muuttujien alkutila ja niihin tulleet muutokset.

3.7.3 Listojen käyttö

Koska interaktiivisissa L-systeemeissä on tärkeää, että uusien relaatioiden johtaminen vanhoista tapahtuu simulointikellon kannalta oikeaan aikaan, on aiempi toteutusmalli korvattava listatoteutuksella. Tällöin aina kun saavutaan uudelle kierrokselle, on käytössä listat v ja u, joista ensinmainitussa on ”vanhat” relaatiot ja jälkimmäisenä mainittuun (joka on kierroksen alussa tyhjä), sijoitetaan evaluointisääntöjen perusteella muodostetut ”uudet” relaatiot. Joissakin tapauksissa myös ympäristön tilaa voidaan tallettaa listoihin.

3.7.4 Kamppailu elintilasta

Seuraavassa esimerkissä, joka on hyvin samantapainen kuin [1]:n kappaleessa 5 esitetty esimerkki (”A model of branch tiers”), tosin eri formalismilla, kuvataan kolmea elintilasta kamppailevaa ”rihmastoa”, jotka kullakin kellojaksolla kasvattavat kustakin

Kuva 3.8: Elintilasta kamppailevat ”rihmastot”

silmusta uudet pätkät, edellyttäen, ettei lähellä ole ennestään liikaa aikaisempia silmuja (omia tai vieraita). Jotta aiempien silmujen asema voitaisiin kätevästi tarkastaa, on ne iteraation kuluessa talletettava listaan. Edellisistä esimerkeistä poiketen, käytetään esimerkissä seuraavia ylimääräisiä merkintöjä: E_1 on ”ympäristö”, tässä tapauksessa silmuista muodostettava lista, $\mathrm{tr}(x, y)$ taas tarkoittaa $(x + l\cos\varphi, y + l\sin\varphi)$, koska se yksinkertaistaa kaavoja. Kun listaan E_1 viitataan, voidaan käyttää normaaleja joukko-opillisia relaatioita, esim $(x, y) \in E_1$, kun siihen lisätään alkioita, käytetään sijoitusoperaattoria tyyliin $E_1 := \mathrm{concat}(E_1, \{p_k\})$. concat tarkoittaa katenaatiota.

$$\begin{aligned} S(n) \quad \rightarrow \quad & P(0, 0, 2, \frac{\pi}{6}, n) \\ & P(0, 0, 2, -\frac{\pi}{6}, n) \\ & P(15, 0, 2, -\frac{11\pi}{12}, n) \\ & E_1 := \{(0, 0)(15, 0)\} \end{aligned} \tag{3.22}$$

Alkutilassa siis kaksi 2:n pituista versoa ”kasvaa” origosta ja yksi $(15, 0)$:sta, suuntakulmien ollessa annettu kaavoissa. Silmulista kattaa aluksi vain alkupisteet.

$$\begin{aligned} & P(x, y, l, \varphi, n)(n > 0) \\ (\#\{(x_E, y_E) \in E_1 : \|\mathrm{tr}(x, y) - (x_E, y_E)\| < 0.9\} < 2) \quad \rightarrow \quad & P(\mathrm{tr}(x, y), \frac{4}{5}l, \varphi + \frac{3\pi}{10}, n - 1) \\ & P(\mathrm{tr}(x, y), \frac{9}{10}l, \varphi + \frac{3\pi}{40}, n - 1) \\ & P(\mathrm{tr}(x, y), \frac{17}{20}l, \varphi - \frac{9\pi}{40}, n - 1) \\ & \rho(x, y, l, \varphi) \\ P(x, y, l, \varphi, n - 1) \quad \rightarrow \quad & E_1 := \mathrm{concat}(E_1, (x, y)) \end{aligned} \tag{3.23}$$

Tässä vaatii muutama asia selvennystä. Ensinnäkin, sääntö $(\#\ldots)$ tarkoittaa sitä, että listassa ei saa olla kahta tai enempää sellaista pistettä, jotka olisivat lähempänä evaluoitavaa silmua kuin 0.9:n verran, jotta uusia graafin sivuja luotaisiin. Niinikään huomiota saattaa herättää viimeisenä oleva sääntö, jossa iteraatiomuuttujan arvo on jo $n-1$. Se (ne) suoritetaan kuitenkin samassa kellojaksossa kuin edellisetkin, koska kyseinen sääntö päivittää silmulistan ajan tasalle seuraavaa iteraatiokierrosta varten.

3.8 Muut formalismit

Tässä dokumentissa on L-systeemien formalisointi johdettu suoraan kieliopeista ja relaatioista käsin. Tämä menetelmä, samoin kuin muutkaan menetelmät eivät kuitenkaan ole universaaleja standardeja, vaan eri kirjoittajat ovat artikkeleissaan käyttäneet muitakin esitystapoja. Muista formalismeista esitettäköön tässä yksi esimerkki sen vuoksi, että sillä on käytännön merkitystä $\mathbb{R}^3$:n L-systeemien kuvauksissa.

3.8.1 Kilpikonnamenetelmä

Mêch ja Prusinkiewicz ovat artikkelissaan [1], käyttäneet ns. ”kilpikonnamenetelmää”. Menetelmä perustuu siihen, että graafi piirretään sen mukaan, miten osoitin, ns. ”kilpikonna” kulkee. ”Kilpikonnan” kulloistakin tilaa kuvataan paikkavektorilla, suunnalla ja päänormaalivektorilla, kuten aiemmin (3.11):ssa kolmikannan tapauksessa. Käytännössä tarvitaan vain muutamia perusoperaattoreita:

$$F$$

siirtää ”kilpikonnaa” eteenpäin,

$$/\ \mathrm{ja}\ \backslash \tag{3.24}$$

kiertävät ”kilpikonnaa” tangenttivektori kiertoakselina,

$$\&\ \mathrm{ja}\ \wedge \tag{3.25}$$

kiertävät ”kilpikonnaa” päänormaalivektori kiertoakselina,

$$-\ \mathrm{ja}\ + \tag{3.26}$$

kiertävät ”kilpikonnaa” sivunormaalivektori kiertoakselina,

$$[\ \mathrm{ja}\ ] \tag{3.27}$$

laittavat ”kilpikonnan” tilan pinoon ja palauttavat pinoon tallennetun tilan takaisin. On huomattava, että tämä idea perustuu pitkälti siihen aiemmin mainittuun tosiasiaan, että konteksivapaiden kielioppien kuvaamat kielet ovat tulkittavissa pinoautomaatilla. Vielä on huomattava Mêch:in ja Prusinkiewicz:in käyttäneen hieman erilaista syntaksia L-systeemiensä johtosäännöissä kuin tämä artikkeli. Heidän artikkelissaan [1], samoin kuin monissa muissa vastaavan aihepiirin artikkeleissa käytetään artikkelissa [2] esiteltyä muotoa, jossa säännöt ovat seuraavaa tyyppiä:

$$id : lc < pred > rc : cond \rightarrow succ : prob \tag{3.28}$$

tai aiemmin kuvatulla kieliopeista johdetulla formalismilla suunnilleen

$$\Sigma : (V^{*\ulcorner} <^{\urcorner} (V \setminus \Sigma)^{\ulcorner} >^{\urcorner} V^* : c_R) \rightarrow V^* : \mathrm{rand}() < prob \tag{3.29}$$

missä id on tunnus (ensimmäiseksi evaluoitava sääntö määräytyy siitä, että tunnus on ω eikä siitä, että vasen puoli olisi S.), $pred$ on johdettavana oleva ei-terminaalinen symboli/relaatio, lc ja rc muodostavat konteksin, $cond$ on ehtorelaatio ja $prob$ on todennäköisyys. Tässä artikkelissa käytin toisenlaista formalismia, koska ehdot ja todennäköisyydet ovat luonnollisella tavalla ajateltavissa osaksi konteksia, jos siellä olevat lausekkeet tulkitaan relaatioiksi. Joitakin ongelmia Mêch:in ja Prusinkiewicz:in formalismissa saattaa aiheuttaa $<$:n ja $>$:n monimerkityksinen käyttö sekä vertailumerkkeinä että konteksierottimina, mikä voi pitkissä johtosäännöissä aiheuttaa kaavojen tulkintavaikeuksia.

3.9 L-systeemit viitteellisten arvojen laskemisessa

Artikkelissaan [4] ovat Parish ja Müller käyttäneet L-systeemejä apuna kaupunkien asemakaavojen simuloinnissa. Vaikka artikkeli sinänsä kertoo heidän lähestymistapansa perustuvan Mêch:in ja Prusinkiewicz:in formalismiin [1], on ”kaupunkiartikkelia” lukevan kuitenkin huomioitava olennainen seikka: Parish ja Müller eivät käytä formuloinneissaan suoraa vastaavuutta L-systeemin lausekkeista graafien sivuihin vaan heidän sovelluksessaan L-systeemin arvot ovat *viitteellisiä*, ja alistetaan useille korjausoperaatioille, kuten sivun pätkimiseen ja uuden graafin kärjen luomiseen tilanteessa, että luotava sivu risteäisi jonkun aiemman kanssa. Niinikään on otettu mahdollisuus poistaa joitakin sellaisia elementtejä, jotka eivät sovi ulkoa annettavaan kaavaan. Tilanpuute ei salli käsitellä tässä artikkelissa mitään esimerkkejä tästä aiheesta, mutta Parishin ja Müllerin artikkeli [4] on joka tapauksessa suositeltava kiinnostuneille. Jonkinlaisena ongelmana voidaan pitää sitä, että he eivät ole ilmaisseet kaikkia tarvittavia algoritmeja siihen, että muut voisivat verifioida heidän tuloksensa (tätä on kritisoitu myös verkkokirjoittelussa). On huomattava, ettei tuonkaltaista tarkoitusperää varten olevien L-systeemien tarvitse välttämättä olla vain viitteellisiä. Osien sopivuus kokonaisuuteen on aivan yhtä hyvin mahdollista koodata L-systeemien sääntöihin ympäristöä kuvaavien muuttujien suhteen.

Luku 4

Lopuksi

4.1 Yhteenveto

Nykyaikainen elävän luonnon ja yhdyskuntien kehityksen tutkimus tarvitsee tuekseen matemaattisia mallinnusvälineitä. Tärkeäksi apuvälineeksi ovat osoittautuneet 1900-luvun alun ja puolenvälin tietämystekniikan tulokset yhdistettyinä relaatioita ja graafeja koskeviin teorioihin, jolloin on kyetty simuloimaan niin elävien kudosten rakenteita kuin ihmisen suunnittelemia yhdyskuntiakin. Näiden simulointi ei rajoitu pelkästään ulkoasuun vaan kattaa myös toiminnallisuuden tutkimusta. Johdettaessa Lindenmeyerin systeemien teoriaa, lähtökohtana ovat kieliopit ja ei-terminaalisten symbolien korvaaminen uusilla. Kun symbolien tilalle ottaa relaatiot, saadaan laskentaa merkittävästi pelkistettyä ja rekursiivisten määritelmien esittäminen kielioppien sisällä helpottuu. Graafit esitellään joukko-opillisesti ja sitten erikoistetaan niin, että niiden kärjet sidotaan johonkin vektoriavaruuteen, yleensä $\mathbb{R}^2$:een tai $\mathbb{R}^3$:een. Tämän jälkeen voidaan esittää kuvauksia terminaalisten relaatioiden joukolta graafin kärkiin ja sivuihin, mikä mahdollistaa monien fraktaalisten objektien havainnollistamisen kuvin. Apuna voidaan käyttää myös satunnaismuuttujia, jos halutaan useammilla simulointikerroilla tutkia vaihtoehtoisia käyttäytymismalleja. Monimutkaisempien ilmiöiden simuloimiseen on johtateltu laatimalla protokolla, jolla voidaan kommunikoida matriisein tai listoin simuloidun ympäristön kanssa sen sisältöön vaikuttaen ja ohjaamalla L-systeemien kehitystä sen sisällön mukaan. On huomioitu, että L-systeemejä voidaan käyttää myös viitearvojen hakuun ja saatuja tuloksia voidaan prosessoida muilla menetelmillä.

4.2 Tästä eteenpäin

Tätä tutkimustyötä voitaisiin jatkaa ja syventää käsittelemällä monimutkaisempia esimerkkejä interaktiosta ympäristön kanssa tai esittämällä käytännön sovelluksia viitearvohausta. Samoin voitaisiin tutkia, millaisia funktioperheitä voitaisiin määrittää, jos $\mathbb{R}^2$:n ja $\mathbb{R}^3$:n tilalle otettaisiin monimutkaisempia sisätuloavaruuksia kuten esim: $L^2([0,1])$. Voidaan samoin esittää kysymyksiä näiden fysikaalisesta merkityksestä.

Kirjallisuutta

[1] Radomir Mêch, Przemyslaw Prusinkiewicz: *Visual models of plants interacting with their environment*, http://algorithmicbotany.org/papers/enviro.sig96.html

[2] Radomir Mêch, Przemyslaw Prusinkiewicz, Jim Hanan: *An L-system-based plant modeling language*, http://algorithmicbotany.org/papers/cpfg.agtive99.html

[3] Richard Dawkins: *Sokea kelloseppä*, WSOY 1989.

[4] Yoav I.H. Parish, Pascal Müller: *Procedural Modeling of Cities*, http://www.centralpictures.com/ce/tpapers.html

[5] Lewis, Papadimitriou: *Elements of the theory of computation*, Prentice Hall 1981.

[6] Gariepy, Ziemer:*Modern real analysis*, PWS Publishing Company.

[7] Eric W. Weisstein:*"Hausdorff Dimension." From MathWorld–A Wolfram Web Resource.* http://mathworld.wolfram.com/HausdorffDimension.html

[8] Eric W. Weisstein:*"Monte Carlo Method." From MathWorld–A Wolfram Web Resource.* http://mathworld.wolfram.com/MonteCarloMethod.html

[9] Eric W. Weisstein:*"Rabbit Sequence." From MathWorld–A Wolfram Web Resource.* http://mathworld.wolfram.com/RabbitSequence.html

[10] Brian W. Kernighan, Dennis M. Ritchie: *The C Programming Language*, Prentice Hall, 1988.

[11] Robert Sedgewick:*Algorithms*, Addison-Wesley, 1989.

[12] Mikko Nummelin:*Lindenmeyerin systeemit matemaattisessa mallinnuksessa, lähdekoodipaketti+liitteet*, 2004-2005
http://www.math.hut.fi/∼mnummeli/lsysteemit/index.html

Hakemisto