L-systeemeistä

Mikko Koli

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Pro gradu -tutkielma
Tammikuu 2001

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Mikko Koli: L-systeemeistä
Pro gradu -tutkielma, 76 sivua, 1 liitesivu

Tammikuu 2001

Tässä työssä käsitellään alunperin biologista mallintamista varten kehitettyjä L-systeemejä. Työssä annetaan tärkeimpien L-systeemityyppien määritelmät ja suhteutetaan ne Chomskyn kielihierarkiaan. Lisäksi työssä käsitellään L-systeemejä sovellettuna erityisesti puumaisten haarautuvien rakenteiden mallintamiseen. Myös joitakin muita tyypillisiä L-systeemien sovellusalueita käsitellään lyhyesti. Lisäksi työn lopuksi annetaan itsekehitetty esimerkki L-systeemien soveltamisesta koirien rodunjalostukseen.

Avainsanat ja -sanonnat: L-systeemit, L-hierarkia, päätösongelmat, merkkijonojen tulkinta, biologinen mallintaminen.

1. JOHDANTO

L-systeemien edeltäjänä voidaan pitää Chomskyn kielioppeja. Chomskyn kielioppeja käytettiin alunperin kielitieteessä luonnollisen kielen esittämiseen ja kuvaamiseen. L-systeemien kehittäjä Aristid Lindenmayer halusi biologina mallintaa kasveja ja niiden kasvua, ja tarvitsi näin ollen uudenlaisen formalismin. Tästä uudesta formalismista alettiin käyttää kehittäjänsä mukaan nimeä L-systeemit.

"L-systeemit tarjoavat matemaattisen formalismin kehittyvien organismien aksiomaattiselle teorialle", kuten Przemyslaw Prusinkiewicz toteaa lähteessä [36]. Aluksi 1970-luvulla tietojenkäsittelyn alalla L-systeemejä tutkittiinkin lähinnä formaalien kielten teoreettisesta näkökulmasta. 1980-luvun puolivälin tienoilla L-systeemien tutkimus alkoi hiljalleen painottua enemmän L-systeemien sovellusten alalle. Esimerkiksi Alvy Ray Smith esitti vuonna 1984, että L-systeemien avulla voidaan luoda todentuntuisia kuvia erilaisista kasveista. Lisäksi hän osoitti, että L-systeemeillä ja Benoit Mandelbrotin fraktaaleilla oli selvä yhteys keskenään. Smithin tulokset innoittivat Prusinkiewiczin ja Hananin työtä, kun he tutkivat L-systeemien soveltuvuutta tietokonegrafiikan alalla. Prusinkiewicz ja Hanan etsivät alunperin vastausta siihen, voidaanko L-systeemien avulla mallintaa luonnosta löytyviä erilaisia kasvilajeja, sekä voidaanko L-systeemien avulla tuottaa kuvia erilaisista fraktaaleista. Tässä työssä käsitellään kaikkia yllämainittuja L-systeemejä koskevia aihealueita, tosin osaa niistä melko pintapuolisesti.

Tämä työ on kaksijakoinen. Aluksi luvuissa 3-5 käsitellään L-systeemien perusteoriaa, loppuosa työstä käsittelee L-systeemien sovelluksia. Työn teoriaosuudessa on käytetty tärkeimpinä lähteinä Hermanin ja Rozenbergin kirjaa *Developmental Systems and Languages* (1975) sekä Rozenbergin ja Salomaan kirjaa *The Mathematical Theory of L Systems* (1980), mutta teoriaosuudessakin on toki käytetty jonkin verran myöskin uudempaa lähdemateriaalia. L-systeemien sovellusten käsittely pohjautuu pääosin Prusinkiewiczin ja Lindenmayerin erinomaiseen kirjaan *The Algorithmic Beauty of Plants* (1990). Lisäksi sovellusosiossa (ja oikeastaan koko työssä) on käytetty hyödyksi varsin paljon myös muita, uudempia Prusinkiewiczin tutkimuksia.

Luvussa 2 annetaan ensin määritelmät tässä työssä käytettäville käsitteille ja merkintätavoille. Lisäksi luvussa 2 esitetään Chomskyn kielihierarkia. Luvussa 3 esitellään tärkeimmät L-systeemityypit ja annetaan niiden formaalit määritelmät. Luku 4 käsittelee ns.

L-hierarkiaa, joka suhteutetaan aiemmin esitettyyn Chomskyn kielihierarkiaan. Teoria-osuuden lopuksi luvussa 5 käsitellään vielä L-systeemien päätösongelmia.

L-systeemien sovelluksien käsittely aloitetaan määrittelemällä luvussa 6 tässä työssä käytettäväksi valittu tulkintatapa L-systeemien avulla tuotetuille merkkijonoille. Tätä tulkintatapaa sovelletaan luvussa 7 erityisesti haarautuvien rakenteiden mallintamiseen. Työn lopuksi luvussa 8 esitellään lyhyesti muita L-systeemien sovellusalueita, sekä hieman laajemmin itsekehitetty esimerkki L-systeemeistä sovellettuna rodunjalostukseen.

2. MÄÄRITELMIÄ JA KÄSITTEITÄ

Tässä luvussa annetaan määritelmät työssä käytettäville käsitteille. Samalla kohdassa 2.1 esitellään joitakin tuloksia, joita käytetään myöhemmin tässä työssä. Tärkeimmille lauseille annetaan myös todistukset ja tässä työssä todistamatta jäävien lauseiden todistuksiin annetaan lähdeviittaukset.

Kohdassa 2.1 käsitellään formaalien kielten peruskäsitteitä sekä määritelmiä, kohdassa 2.2 päätösongelmia ja kohdassa 2.3 annetaan lyhyt katsaus fraktaaleihin.

2.1 FORMAALEISTA KIELISTÄ

Tässä kohdassa annetaan määritelmät ja merkintätavat tärkeimmille formaalien kielten peruskäsitteille. Näitä merkintätapoja ja määritelmiä käytetään soveltuvin osin tässä työssä jatkossa esitettäviin L-systeemeihin ja niiden määritelmiin.

Vaikka tässä kohdassa esiteltävät määritelmät koskevatkin Chomskyn kielioppeja, eivätkä siis työn aiheena olevia kielioppityyppejä (L-systeemejä), on Chomskyn kielioppeihin liittyvien määritelmien ja merkintätapojen esittäminen tässä yhteydessä silti perusteltua. L-systeemien yhteydessä käytetyt käsitteet voidaan nimittäin ymmärtää Chomskyn kieliopeissa käytettyjen käsitteiden erikoistapauksiksi. Näin ollen jos Chomskyn kielioppien käsitteet ovat lukijalle ennestään tuttuja, niin myös L-systeemejä koskevat käsitteet tuntuvat jo valmiiksi tutuilta. Toinen huomionarvoinen seikka on se, että Chomskyn kielioppeja ja kieliperheitä on tutkittu jo vuosikymmeniä, ja lukuisia näihin kieliperheisiin liittyviä väittämiä, lauseita ja ominaisuuksia on pystytty todistamaan. Tästä syystä L-systeemejä kannattaa yrittää suhteuttaa Chomskyn kielihierarkiaan, sillä jos onnistutaan todistamaan jonkin tietyn L-systeemiperheen sisältyvän johonkin Chomskyn kieliperheeseen, niin samalla kaikki ko. Chomskyn kieliperhettä koskevat tulokset pätevät myös tähän L-systeemiperheeseen.

Algoritmin ja proseduurin peruskäsitteet oletetaan tässä työssä tunnetuiksi. Algoritmin tyyppi merkitään muodossa ($X \Rightarrow Y$). Merkintä tarkoittaa, että algoritmin syöte kuuluu joukkoon X ja tuloste joukkoon Y. Jatkossa tarkastellaan erityisesti tyyppiä ($X \Rightarrow B$) olevia algoritmeja, missä B on totuusarvojen true (T) ja false (F) muodostama joukko. [25]

Kuvaus f: X $\rightarrow$ Y on *algoritminen* (*algoritmisesti laskettavissa*), jos on olemassa sellainen täydellinen, deterministinen tyyppiä (X$\Rightarrow$Y) oleva algoritmi A, että kaikille joukon X alkioille x pätee f(x) = A(x). [25]

Siirrytään sitten varsinaisiin formaalien kielten käsitteisiin. *Aakkosto* on äärellinen joukko, jonka alkiot ovat *merkkejä*. *Lause* muodostuu äärellisestä aakkostoon kuuluvien merkkien jonosta. *Tyhjä lause* Λ ei sisällä yhtään merkkiä. *Kieli* on mikä tahansa tietyn aakkoston merkeistä koostuvien lauseiden joukko. Jos aakkostosta käytetään merkintää V, niin V^* tarkoittaa kaikkien V:n lauseiden joukkoa (V^* siis sisältää myös tyhjän lauseen). Vastaavasti V^+ tarkoittaa kaikkien ei-tyhjien V:n lauseiden joukkoa $V^* \setminus \{\Lambda\}$. Jos x ja y ovat kaksi joukon V^* lausetta, niin niiden *liitos* xy saadaan kirjoittamalla lauseet peräkkäin. Merkintä x^i tarkoittaa merkkijonoa, joka saadaan liittämällä i kappaletta merkkijonoja x toisiinsa. Näin ollen $x^0 = \Lambda$, $x^1 = x$, $x^2 = xx$, jne. [16, 25]

Merkintä V^* tarkoittaa siis liitosoperaation transitiivista ja refleksiivistä sulkeumaa joukon V merkkien suhteen ja V^+ saman operaation transitiivista sulkeumaa. Kielen L ($\subseteq V^*$) komplementilla L^c tarkoitetaan niiden joukon V^* lauseiden muodostamaa joukkoa, jotka eivät kuulu kieleen L.

Kieliperhe on yksinkertaisesti tietyt ominaisuudet sisältävien kielten joukko. Kieliperhe voidaan määritellä ainakin seuraavin vaihtoehtoisin tavoin [25]:

1. annetaan kieliä generoivien algoritmien joukko
2. annetaan kieliä hyväksyvien algoritmien joukko
3. määritellään perhe matemaattisten ominaisuuksiensa perusteella
4. tarkastellaan kielten sijasta niissä käytettäviä ilmauksia ja määritellään näiden ilmaisujen syntaksi ja semantiikka.

Esimerkit kaikista em. tavoista määritellä kieliperhe löytyvät lähteestä [25]. Seuraavaksi esitettävässä Chomskyn kieliopin määritelmässä esitetään lähteisiin [16, 25] perustuva esimerkki kohdan 1 tavasta määritellä kieliperhe. Ennen sitä määritellään kuitenkin kieliperheen merkintätapa.

Jos jotakin kieltä, jolla on jokin tietty ominaisuus p, merkitään kirjaimella X, niin kaikkien ominaisuuden p sisältävien kielten perhettä merkitään L_X.

Jos kieliperhe halutaan määritellä antamalla kieliä *generoivien* algoritmien joukko, niin tavallisin tapa generoida kieliperheeseen kuuluvat kielet on käyttää Chomskyn kielioppeja. Chomskyn kieliopin määritelmä on seuraava [16, 25]:

Chomskyn kielioppi on järjestelmä $G = \langle N,T,P,S \rangle$, missä

N on *apumerkkien* aakkosto (*nonterminaalit*),

T on *perusmerkkien* aakkosto (*terminaalit*),

P on äärellinen *sääntöjen* joukko aakkostossa $N \cup T$ ja

S on joukon N erillinen alkio, jota kutsutaan *alkumerkiksi*.

Sääntö on muotoa $\alpha \rightarrow \beta$ oleva ilmaus, missä $\alpha \in (N \cup T)^+$ ja $\beta \in (N \cup T)^*$.

L-systeemeille annetaan vastaavat määritelmät myöhemmin luvussa 3, mutta jo tässä vaiheessa voidaan kertoa oleellisimmat erot Chomskyn kielioppien ja L-systeemien välillä. Chomskyn kieliopeissa niiden sääntöjä sovelletaan peräkkäin ja niiden aakkosto on jaettu erikseen perusmerkkeihin ja apumerkkeihin. L-systeemeissä puolestaan sääntöjä sovelletaan rinnakkain ja niiden aakkostossa ei tehdä eroa apu- ja perusmerkkien välillä. Chomskyn kielioppien ja L-systeemien välimuotona voidaan pitää ns. puhtaita kielioppeja (*pure grammars*), joissa sääntöjä sovelletaan Chomskyn kielioppien tapaan peräkkäin, mutta aakkostoa ei ole jaettu. Puhtaiden kielioppien formaaleja määritelmiä ei tässä työssä anneta, mutta tarvittaessa ne löytyvät mm. lähteestä [23].

Tässä työssä merkeistä ja merkkijonoista (lauseista) käytetään seuraavia merkintöjä:

- perusmerkeistä käytetään pieniä aakkoston alkupään kirjaimia a, b, c,...
- apumerkeistä käytetään isoja kirjaimia A, B, ..., S,...
- perusmerkeistä muodostuvaa merkkijonoa merkitään pienillä (englantilaisen) aakkoston loppupään kirjaimilla z, y, x, w,...
- aakkoston $N \cup T$ merkeistä muodostuvia lauseita merkitään kreikkalaisilla kirjaimilla α, β,...

Jos oletetaan, että kielioppi $G = \langle N,T,P,S \rangle$ sekä lauseet ρ ja $\sigma \in (N \cup T)^*$ on annettu, niin σ on johdettu G:ssä suoraan (*directly derived*) ρ:stä (merk. $\rho \Rightarrow \sigma$) jos ja vain jos seuraavat ehdot täyttyvät:

1. ρ voidaan kirjoittaa muodossa $\gamma\alpha\delta$,
2. σ voidaan kirjoittaa muodossa $\gamma\beta\delta$, ja
3. sääntö $\alpha \rightarrow \beta \in P$.

Relaation $\Rightarrow$ transitiivisesta sulkeumasta käytetään merkintää $\Rightarrow^+$ ja transitiivisesta, reflek-siivisesta sulkeumasta merkintää $\Rightarrow^*$. Merkintä $\rho \Rightarrow^+ \sigma$ tarkoittaa siis, että σ saadaan ρ:stä

soveltamalla peräkkäin äärellinen määrä annettuja sääntöjä. Tapauksessa $\rho \Rightarrow^* \sigma$ on lisäksi mahdollista, että sääntöjä on sovellettu nolla kertaa, jolloin $\rho = \sigma$. [16, 25]

Kieliopin $G = \langle N,T,P,S \rangle$ tuottama kieli määritellään joukkona $L(G)=\{w \mid w \in T^* \text{ ja } S \Rightarrow^+ w\}$. Lause w kuuluu siis kieleen L(G) jos se koostuu pelkistä perusmerkeistä ja se on johdettavissa S:stä. [25]

Chomskyn kielihierarkia esitetään lauseessa 2.1 sekä kuvassa 1. Sitä ennen esitetään kuitenkin määritelmät 2.1 – 2.3, joita tarvitaan lauseessa 2.1, sekä merkintätapa 2.1, jota tarvitaan määritelmässä 2.1.

Merkintätapa 2.1.
Jos α on lause, niin merkinnällä $|\alpha|$ tarkoitetaan lauseen α pituutta, eli lauseen merkkien lukumäärää. Lisäksi merkinnällä $|\alpha|_a$ tarkoitetaan merkkien a esiintymislukumäärää lauseessa α.

Määritelmä 2.1.
Muotoa $A \rightarrow aB$, $A \rightarrow B$, $A \rightarrow a$ ja $A \rightarrow \Lambda$ olevat säännöt ovat *säännöllisiä*. Muotoa $A \rightarrow \alpha$, $A \in N$, $\alpha \in (N \cup T)^*$, oleva sääntö on *kontekstiton*. Sääntö $\alpha \rightarrow \beta$, $\alpha, \beta \in (N \cup T)^*$, on *lyhentämätön*, jos $|\alpha| \leq |\beta|$. [25]

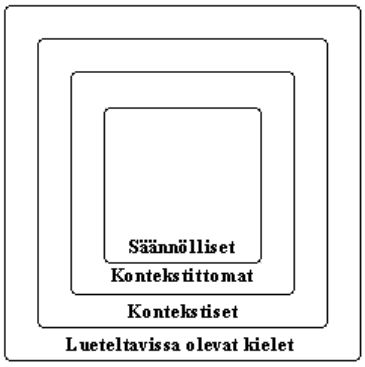
Määritelmä 2.2.
Olkoon $G = \langle N,T,P,S \rangle$ kielioppi. Jos P:n jokainen sääntö on lyhentämätön, niin G on *kontekstinen*. Jos P:n kaikki säännöt ovat kontekstittomia, niin myös G on *kontekstiton*. Jos P:n säännöt ovat säännöllisiä, niin G on *säännöllinen*. [16, 25]

Määritelmä 2.3.
Kieltä L sanotaan *lueteltavissa olevaksi kieleksi*, jos on olemassa sellainen kielioppi G, että L = L(G). Kieli L on lisäksi kontekstinen, kontekstiton tai säännöllinen mikäli G on vastaavasti kontekstinen, kontekstiton tai säännöllinen kielioppi. [16]

Tässä työssä termi "lueteltavissa oleva" lyhennetään RE (*recursively enumerable*), "kontekstinen" CS (*context sensitive*), "kontekstiton" CF (*context free*) sekä "säännöllinen" RG (*regular*). Nyt voidaan kirjoittaa:

Lause 2.1.
$L_{RG} \subsetneqq L_{CF} \subsetneqq L_{CS} \subsetneqq L_{RE}$.

Todistus.
Se, että kukin kieliperhe sisältyy sitä seuraavaan kieliperheeseen seuraa määritelmistä 2.1 – 2.3. Sisältymisen aitoutta ei tässä työssä todisteta, mutta tarvittavat todistukset löytyvät mm. lähteistä [25, 18].



Kuva 1. Lauseessa 2.1 esitetty Chomskyn kielihierarkia.

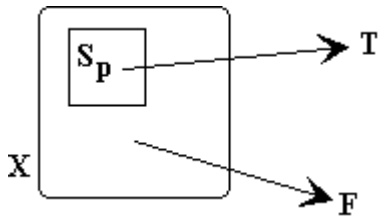
2.2 PÄÄTÖSONGELMISTA

Päätösongelmia käsiteltäessä tarkastellaan totuusarvoisia funktioita (*predikaatteja*) $p:X\rightarrow B$, joiden määrittelyalueena on jokin algoritmeissa mahdollinen ääretön perusjoukko. Päätösongelman ratkaisu määrittää, onko annetun joukon X alkiolla x voimassa $p(x) = T$, eli onko alkiolla x ominaisuus p. [25]

Predikaattiin $p:X\rightarrow B$ liittyvä päätösongelma on *algoritmisesti ratkeava* (tai lyhyemmin *ratkeava*), jos p on algoritminen funktio. Muussa tapauksessa ongelma on (*algoritmisesti*) *ratkeamaton*. [25]

Joukkoa $S_p = \{x \in X \mid p(x) = T\}$ sanotaan päätösongelman $p:X\rightarrow B$ karakteristiseksi joukoksi [25]. S_p on siis kaikkien niiden alkioiden joukko, joilla on ominaisuus p.

Kuvassa 2 havainnollistetaan ratkeavan päätösongelman p tilannetta. Päätösongelman p ratkaisee tyyppiä $(X\Rightarrow B)$ oleva algoritmi A_p, joka palauttaa arvon T täsmälleen silloin, kun sen syöte kuuluu päätösongelman p karakteristiseen joukkoon S_p. [25]



Kuva 2. Ratkeava päätösongelma.

Kannattaa pitää mielessä, että ratkeavuuden määritelmässä on kyse vain algoritmin olemassaolosta, eikä siitä, voitaisiinko vaadittava algoritmi jossain käytännön tilanteessa antaa [25].

2.3 LYHYESTI FRAKTAALEISTA JA FRAKTAALIGEOMETRIASTA

Osa tässä työssä jatkossa esitettävistä kuvioista voidaan ymmärtää fraktaaleiksi. Tästä syystä tässä kohdassa annetaan määritelmät fraktaalille, fraktaalidimensiolle sekä Kochin käyrälle, joka lienee yksi tunnetuimmista fraktaalikäyristä. Tämän kohdan esitys fraktaaleista on varsin pintapuolinen, mutta tarvittaessa perusteellisempaa tietoa löytyy mm. lähteistä [22, 27, 28].

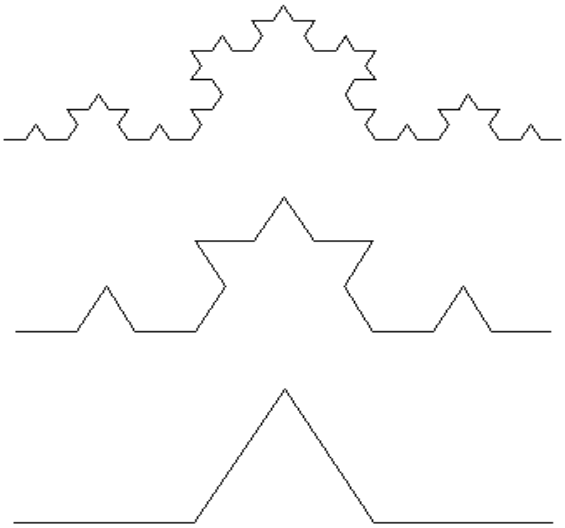
Mandelbrot määrittelee fraktaalin käyräksi, jonka fraktaalidimensio D_H (toiselta nimeltään Haussdorff-Besicovitch-dimensio) on aidosti suurempi kuin sen topologinen dimensio D_T [38]. Fraktaalidimensio kertoo kuinka "rosoinen" käyrä on. Esimerkiksi tavallisen janan tai muun käyrän topologinen dimensio on 1, tason tai pinnan dimensio on 2 ja kolmiulotteisen kappaleen tietenkin 3. Kun puhutaan fraktaalikäyrästä, niin sen voidaan ajatella olevan niin rosoinen käyrä, että se muodostaa oikeastaan 2-ulotteisen pinnan. Voidaan siis ajatella että rosoisuuden kasvaessa myös käyrän dimensio kasvaa. Kun janan topologinen dimensio on 1 ja tason siis 2, niin fraktaalikäyrän dimensio voi olla mikä tahansa reaaliluku väliltä [1…2]. [12]

Vaikkei fraktaalidimension D_H täsmällinen määritelmä (laskukaava) olekaan tämän työn kannalta mitenkään erityisen tärkeä, annetaan se kuitenkin Kochin käyrän määritelmän jälkeen.

Tärkeä fraktaaleja kuvaileva käsite on itsesimilaarisuus (*self-similarity*). Tunnetuin esimerkki tästä on ns. Kochin käyrä, josta eri mitta-asteikoilla tarkasteltaessa voidaan aina nähdä sama perusmuoto (ks. kuva 3). Kochin käyrä on saanut nimensä ruotsalaisen matemaatikon Helge von Kochin mukaan, joka kuvasi sen ensimmäisen kerran vuonna 1904. Kochin käyrä voidaan konstruoida seuraavasti [11, 28]:

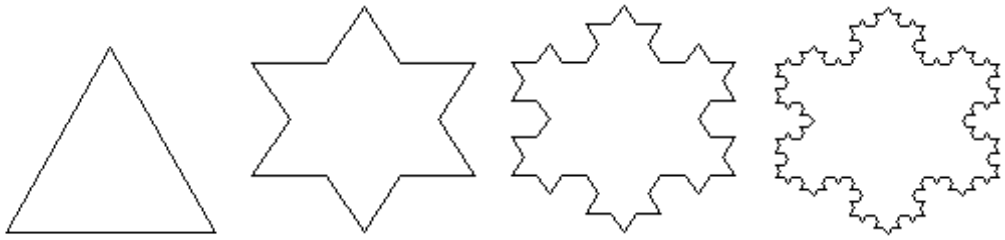
1. piirretään pituudeltaan yhden yksikön mittainen vaakasuora jana
2. jokaisen janan keskelle lisätään tasasivuinen kolmio, jonka sivun pituus on 1/3 (alkuperäisestä janasta poistetaan keskimmäinen kolmannes, kuten myös lisättävän kolmion kanta)
3. kohtaa 2 toistetaan äärettömästi.

Kuvassa 3 on esitetty yo. Kochin käyrän konstruointitavan iteraatiot 1-3.



Kuva 3. Kochin käyrä.

Kun edellä esitetyn Kochin käyrän konstruointitavan 1. kohdaksi vaihdetaan tasasivuinen kolmio, jonka sivujen pituudet ovat 1 yksikköä, niin saadaan aikaan proseduuri ns. Kochin lumihiutaleen muodostamiselle. Kuvassa 4 on esitetty iteraatiot 0-3 Kochin lumihiutaleen konstruoinnista. [11, 28]



Kuva 4. Kochin lumihiutale.

Esimerkin vuoksi voidaan todeta, että yllä esitetyn Kochin lumihiutaleen fraktaalidimensio on 1,2618. Se lasketaan kaavalla

$$D_H = \frac{\log(L2/L1)}{\log(S1/S2)}, \qquad (1)$$

missä L1 ja L2 ovat mitattujen käyrien (tai janojen) pituudet mittayksikköinä ja S1 ja S2 ovat käytettyjen mittayksiköiden pituudet [12, 22, 27, 28]. Jos otetaan esimerkiksi kuvan 4 kaksi ensimmäistä kuviota, niin ensimmäisen kuvion tasasivuisen kolmion kannan pituus L1 on 1 mittayksikköä ja käytetyn mittayksikön pituus on 1. Kuvan 4 toisessa kuviossa on ensimmäisen kuvion kolmion kanta korvattu käyrällä (tai tässä paremminkin jana-sekvenssillä), jonka pituus L2 on 4 mittayksikköä. Toisessa kuviossa käytetyn mittayksikön pituus S2 on 1/3. Kun em. luvut sijoitetaan kaavaan (1), niin fraktaalidimensioksi saadaan 1,2618. (Fraktaalidimensio lasketaan siis peräkkäisillä iteraatiokierroksilla syntyvien kuvioiden perusteella).

Jatkossa tässä työssä käytetään termiä Kochin käyrä hieman yleisemmin, eli aloitus-objektina voi olla paitsi jana tai tasasivuinen kolmio, myöskin jokin muu geometrinen objekti, kuten esimerkiksi neliö (vrt. kuva 14). Aiemmin esitetyn kuvan 4 käyrä tullaan esittämään esimerkinomaisesti myös muutamien L-systeemien avulla luvussa 6.

3. L-SYSTEEMIT

Aristid Lindenmayer määritteli vuonna 1968 uuden kielioppityypin (*parallel derivation grammar*). Uuden kielioppityypin kielioppeja alettiin kutsua kehittäjänsä mukaan L-systeemeiksi. Alkuperäisenä tarkoituksena oli mallintaa yksinkertaisten kuitumaisten organismien kehitystä. Tällaisissa organismeissa kehittyminen tapahtuu rinnakkaisesti, samanaikaisesti kaikkialla organismissa. Tämän takia L-systeemiin kuuluvia sääntöjä sovelletaan rinnakkaisesti jokaisella iteraatiokierroksella, kun taas Chomskyn kieliopeissa yhdellä iteraatiokierroksella sovelletaan ainoastaan yhtä sääntöä. Chomskyn kielioppeja käytettäessä rinnakkaisuuden voi näennäisesti saavuttaa soveltamalla peräkkäin sääntöjä jokaiseen lauseen merkkiin, mutta *synkronoiva kontrollimekanismi* puuttuu kuitenkin tällaisista kieliopeista. Näin ollen rinnakkaisuutta ei voida kunnolla simuloida peräkkäisyyteen perustuvilla kieliopeilla. [20]

Oletetaan esimerkiksi, että ainoa käytettävissä oleva sääntö on $a \rightarrow a^2$, ja aksiooma on a^3. Jos tätä ainoata sääntöä sovelletaan aksioomaan peräkkäisesti, niin lauseen jokainen a korvataan a^2:lla, eli tähän kieleen kuuluvat lauseet a^i, $i \geq 3$. Jos sääntöä sovelletaan rinnakkaisesti aksioomaan, niin ensimmäisen iteraation jälkeen aksioomasta muodostuu lause a^6. Lauseet a^4 ja a^5 eivät siis kuulu ko. kieleen, jos sääntöä sovelletaan aksioomaan rinnakkaisesti, vaan kieli muodostuu lauseista a^{3*2^i}, $i \geq 0$. On helppo todeta, että kieli $\{a^{3*2^i} \mid i \geq 0\}$ ei ole muodostettavissa millään kontekstittomalla kieliopilla, kun kieliopin sääntöjä sovelletaan peräkkäin. [20]

Myöskin identiteettisäännön $a \rightarrow a$ rooli riippuu siitä, sovelletaanko kieliopin sääntöjä peräkkäin vai rinnakkain. Jos sääntöjä sovelletaan peräkkäin, niin identiteettisäännöllä ei ole mitään vaikutusta kieliopin avulla määriteltävään kieleen, eli identiteettisääntö voidaan tässä tapauksessa poistaa kieliopista. Jos taas sääntöjä sovelletaan rinnakkain, niin identiteettisääntö mahdollistaa sääntöjen peräkkäisen soveltamisen simuloimisen rinnakkain sovellettavilla säännöillä: varsinaista sääntöä ($a \rightarrow a^2$) sovelletaan yhteen a:n ilmentymään ja kaikkiin muihin a:n ilmentymiin (annetussa lauseessa) sovelletaan identiteettisääntöä. Näin kaikki lauseet a^i, $i \geq 3$ voidaan muodostaa soveltamalla sääntöjä myöskin rinnakkain, jos molemmat säännöt $a \rightarrow a^2$ ja $a \rightarrow a$ ovat käytettävissä. [20]

L-systeemit määriteltiin alunperin mallintamaan monisoluisten organismien kehitystä, ts. mallintamaan todellisten elävien organismien kehitystä. L-systeemit ovat kuitenkin osoittaneet käyttökelpoisuutensa myöskin moninaisissa tietokonegrafiikkaa hyödyntävissä

sovelluksissa, joissa L-systeemejä käytetään kuvaamaan ei-todellisia elämän muotoja. L-systeemien käyttö tällaisten mielikuvituksellisten organismien, ts. tekoelämän (*artificial life*), kuvaamiseen on ymmärrettävää, koska L-systeemejä on helppo muokata kuvaamaan erilaisia uusia tekoelämän ilmiöitä. Toisaalta todellisten olemassa olevien organismien kuvaamiseen tarvittavien L-systeemien muodostaminen on kaikkea muuta kuin triviaali ongelma. [20]

Vaikka L-systeemit kehitettiin alunperin kasvien kehittymisen formaaliksi mallinnusmenetelmäksi, on niitä nyttemmin sovellettu varsin paljon myös mm. fraktaalikuvioiden, maisemien ja kasvinkaltaisten rakenteiden generoimiseen, ihmiskehon organismien kuvaamiseen ja jopa musiikin säveltämiseen. L-systeemien sovellusalueita on nykyään niin runsaasti, että kaikkia niitä ei tämän työn puitteissa ole mahdollista käsitellä, mutta joitakin sovellusalueita esitellään kuitenkin tämän työn luvuissa 7 ja 8.

L-systeemejä voidaan luokitella monellakin eri tavalla. Näistä tavoista ehkä formaalein on esitetty luvussa 4, jossa esitellään ns. L-hierarkia. Jos L-systemejä halutaan kuitenkin soveltaa biologisessa kontekstissa, saattaa lähteiden [38, 43] tapa olla luontevin. Tässä luokittelussa L-systeemit jaetaan säännöissä esiintyvien erojen mukaan D0L-systeemeihin, puu 0L -systeemeihin (*tree 0L-systems*), pino 0L -systeemeihin (*bracketed 0L -systems*), stokastisiin L-systeemeihin, kontekstisiin L-systeemeihin (*context sensitive L-systems*) ja parametroituihin L-systeemeihin (*parametric L-systems*).

Muista L-systeemien luokittelutavoista mainittakoon lähteiden [1, 2] tapa, jossa L-systeemit luokitellaan seuraaviin kuuteen pääluokkaan:

- kontekstiset L-systeemit, IL-systeemit (*context-sensitive systems*),
- kontekstittomat L-systeemit, 0L-systeemit (*context-free systems*),
- deterministiset L-systeemit, DL-systeemit (*deterministic systems*),
- propagoivat L-systeemit, PL-systeemit (*propagative / propagating systems*),
- laajennetut L-systeemit, EL-systeemit (*systems with extensions*) ja
- taulukko L-systeemit, TL-systeemit (*systems with tables*).

Näitä erilaisia L-systeemejä voidaan yhdistellä, esimerkiksi D0L-systeemi on deterministinen kontekstiton L-systeemi, PD0L-systeemi on propagoiva, deterministinen ja kontekstiton, jne. Myös tämä luokittelu perustuu eroihin sovellettavissa säännöissä. Varsinkin L-systeemien teoriaa käsittelevissä lähteissä käytetään yleisesti nimenomaan tätä luokittelutapaa. Tässä luvussa määritellään tämän työn kannalta tärkeimmät L-systeemityypit lähinnä

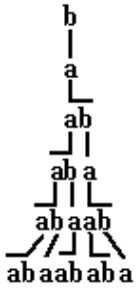
jälkimmäisen L-systeemijaottelun perusteella. Ensimmäisen jaottelun mukaisista L-systeemityypeistä tässä luvussa määrittelemättä jäävät L-systeemityypit määritellään myöhemmin L-systeemien sovelluksia käsittelevässä luvussa 7.

Tämän luvun kohdissa 3.1 – 3.6 esitellään lähteissä [38, 43] esitetyn jaon mukaiset L-systeemityypit seuraavasti: kohdassa 3.1 määritellään 0L-, D0L- ja P0L-systeemit, kohdassa 3.2 E0L- ja T0L-systeemit, kohdassa 3.3 kontekstiset L-systeemit, kohdassa 3.4 stokastiset L-systeemit ja lopuksi kohdissa 3.5 ja 3.6 parametroidut L-systeemit. Lisäksi luvussa 6 kerrotaan L-systeemeillä tuotettujen merkkijonojen graafisesta tulkinnasta sekä luvussa 7 erityisesti haarautuvien rakenteiden kuvaamisessa käytetyistä menetelmistä. Kohdat 3.1 ja 3.3 – 3.6 ovat mukana tässä työssä L-systeemien sovelluksia esittelevän osion (luvut 6-8) takia, kohta 3.2 oikeastaan pelkästään luvun 4 tarpeiden takia.

3.1 0L-, D0L-, JA P0L-SYSTEEMIT

Deterministiset kontekstittomat L-systeemit muodostavat kaikkein yksinkertaisimman L-systeemien luokan. Tämän luokan L-systeemejä nimitetään yhteisesti D0L-systeemeiksi. D0L-systeemien periaate käy ilmi seuraavasta esimerkistä.

Merkkijono (lause) koostuu kahdesta kirjaimesta a ja b, jotka voivat esiintyä monta kertaa merkkijonossa (lauseessa). Kumpaankin kirjaimeen liittyy oma sääntö (*rewriting rule*), jota soveltamalla alkuperäiseen merkkijonoon saadaan aikaan uusi merkkijono. Kirjaimeen a liittyvä sääntö on a→ab, joka käytännössä tarkoittaa sitä, että kaikki a:n esiintymät merkkijonossa korvataan merkeillä ab. Vastaava kirjaimeen b liittyvä sääntö on b→a, joka tarkoittaa puolestaan sitä, että kaikki b:n esiintymät merkkijonossa korvataan kirjaimella a. Alkuperäistä merkkijonoa, johon näitä sääntöjä ryhdytään soveltamaan, kutsutaan aksioomaksi. Olkoon tämän esimerkin aksiooma kirjain b. Kun tähän aksioomaan sovelletaan em. sääntöjä vaikkapa 5 kertaa, niin lopputulokseksi saadaan merkkijono abaababa (ks. kuva 5).



Kuva 5. Esimerkki D0L-systeemillä tuotetusta merkkijonosta.

Kuten esimerkistä nähdään, kaikki D0L-systeemien säännöt ovat muotoa:

säännön vasen puoli $\rightarrow$ säännön oikea puoli,

ja säännön tulkinta on se, että säännön vasen puoli korvataan säännön oikealla puolella annetussa lauseessa.

D0L-systeemien ja niiden operaatioiden formaalit määritelmät ovat seuraavat [38, 16]:
(sis. 0L, D0L, ja P0L-systeemien määritelmät)

Määritelmä 3.1.
Merkkijonoja tuottava 0L-systeemi on järjestetty kolmikko $G = \langle V, \omega, P \rangle$, missä V on systeemin aakkosto, $\omega \in V^+$ on aksioomaksi kutsuttu ei-tyhjä lause, ja $P \subset V \times V^*$ on äärellinen sääntöjen joukko. Sääntö $(a, \chi) \in P$ kirjoitetaan muodossa $a \rightarrow \chi$, missä aakkoston merkkiä a kutsutaan säännön vasemmaksi puoleksi (*predecessor*) ja lausetta χ säännön oikeaksi puoleksi (*successor*). Jokaista merkkiä $a \in V$ kohti on olemassa ainakin yksi sellainen lause $\chi \in V^*$, että $a \rightarrow \chi$. Jos säännön vasemmalle puolelle $a \in V$ ei ole eksplisiittisesti ilmaistua sääntöä, niin identiteettisääntö (*identity production*) $a \rightarrow a$ oletetaan kuuluvaksi sääntöjen joukkoon P. 0L-systeemi on deterministinen (eli D0L-systeemi), jos ja vain jos jokaista $a \in V$ kohti on olemassa täsmälleen yksi sellainen $\chi \in V^*$, että $a \rightarrow \chi$ on systeemin sääntö. 0L-systeemi on propagoiva (eli P0L-systeemi), jos ja vain jos jokaiselle $a \in V$ pätee $a \rightarrow \Lambda \notin P$ ja $\omega \neq \Lambda$.

Kun verrataan L-systeemejä koskevaa määritelmää 3.1 Chomskyn kieliopin määritelmään sivulla 5 huomataan, että L-systeemien aakkostoa V vastaa Chomskyn kieliopin määritelmän joukko $N \cup T$. Ts. L-systeemeissä aakkostoa ei jaeta erikseen apumerkkeihin ja perusmerkkeihin. Chomskyn kieliopin määritelmän alkumerkki vastaa L-systeemien aksioomaa. Näiden suurin periaatteellinen ero on se, että alkumerkki on tasan yhden merkin mittainen merkkijono, kun taas aksiooman pituutta ei ole rajoitettu.

Määritelmä 3.2.
Olkoon $\mu = a_1 \ldots a_m$ mielivaltainen V:n lause. Lause $\nu = \chi_1 \ldots \chi_m \in V^*$ on johdettu suoraan (*directly derived*) μ:stä (merk. $\mu \Rightarrow \nu$) jos ja vain jos $a_i \rightarrow \chi_i$ jokaiselle $i = 1, \ldots, m$. Lause ν on johdettavissa n:n pituisella johdolla L-systeemin G avulla, jos on olemassa sellainen lausesekvenssi (*developmental sequence*) $\mu_0, \mu_1, \ldots, \mu_n$, että $\mu_0 = \omega$, $\mu_n = \nu$ ja $\mu_0 \Rightarrow \mu_1 \Rightarrow \ldots \Rightarrow \mu_n$. Määritelmän perusteella $\Lambda \Rightarrow y$ jos ja vain jos $y = \Lambda$. [38]

Chomskyn kielioppeja koskevat määritelmät sivulla 5 eroavat vastaavasta L-systeemejä koskevasta määritelmästä 3.2 kahdella oleellisella tavalla:

1. L-systeemien tapauksessa kaikki aksioomasta johdettavissa olevat lauseet kuuluvat ko. kieleen, kun Chomskyn kieliopin tapauksessa kieleen kuuluvat vain perusmerkeistä koostuvat lauseet.
2. L-systeemeissä sääntöjä sovelletaan rinnakkain, siis lauseen jokaiseen merkkiin samanaikaisesti, kun Chomskyn kieliopeissa yhden iteraatiokierroksen aikana sovelletaan ainoastaan yhtä sääntöä yhteen lauseen merkkiin kerrallaan.

Seuraavassa on esitetty toinen esimerkki D0L-systeemin toiminnasta. Esimerkissä simuloidaan hyvin karkealla tasolla solun jakautumista solukoksi. Symbolit a ja b esittävät solujen "tilaa" (solun kokoa ja sen valmiutta jakautua) ja symboleihin liittyvät alaindeksit v ja o indikoivat mihin tilaan solun jakautumisen yhteydessä syntyvät uudet solut asettuvat. Kehitystä kuvataan seuraavalla L-systeemillä [38]:

$$
\begin{aligned}
&\omega : a_o \\
&p_1 : a_o \rightarrow a_v b_o \\
&p_2 : a_v \rightarrow b_v a_o \\
&p_3 : b_o \rightarrow a_o \\
&p_4 : b_v \rightarrow a_v .
\end{aligned}
\qquad (2)
$$

Kun aksioomaan (a_o) aletaan soveltaa ylläkuvattua L-systeemiä, saadaan lausesekvenssi, joka on havainnollistettu kuvassa 6. Kannattaa kuitenkin huomioida, että L-systeemien diskreetistä luonteesta johtuen solujen jatkuvaa kasvua jakautumisten välillä ei voida tämän mallin avulla esittää.

a_o

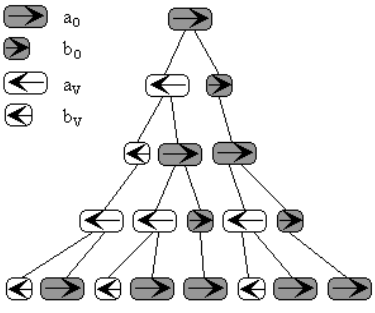
$a_v b_o$

$b_v a_o a_o$

$a_v a_v b_o a_v b_o$

$b_v a_o b_v a_o a_o b_v a_o a_o$

...



Kuva 6. Solun jakautumisen simulointi D0L-systeemillä.

3.2 EOL- JA TOL-SYSTEEMIT

Tässä kohdassa tavallisten 0L-systeemien määritelmään lisätään aluksi apumerkit. Eli samaan tapaan kuin Chomskyn kieliopeissakin, E0L-systeemeissä kielen tuottaviin sääntöihin voidaan sisällyttää sellaisia merkkejä, jotka eivät esiinny ko. kieliopin määrittelemän kielen lauseissa.

Lähteessä [16] on esitetty useita syitä E0L-systeemien tutkimiselle (ks. [16, s.93]). Tämän työn kannalta niistä ehkä mielenkiintoisin on se, että E0L-systeemejä tutkimalla saadaan tavallisia 0L-systeemejä paremmin esiin sääntöjen rinnakkaisen soveltamisen vaikutukset syntyvän kielen ominaisuuksiin. Perusteluna tähän seikkaan on se, että kun 0L-systeemit eroavat Chomskyn kieliopeista sekä tavassa, jolla sääntöjä sovelletaan (rinnakkaisuus / peräkkäisyys), että aakkoston jakamisessa (Chomskyn kieliopeissa aakkosto jaetaan apu- ja perusmerkkeihin), niin E0L-systeemit eroavat Chomskyn kieliopeista pelkästään sääntöjen soveltamistavan suhteen.

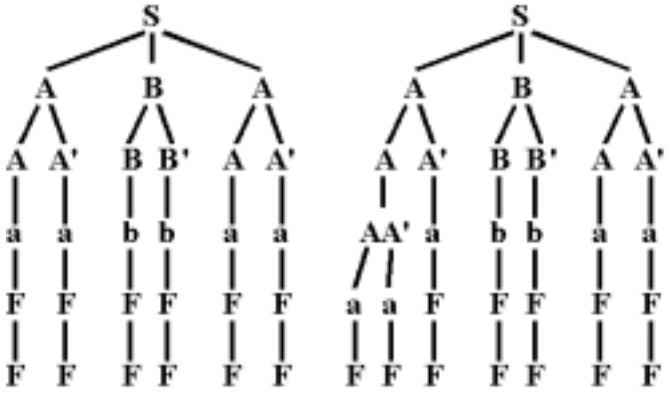
Koska tässä työssä E0L-systeemejä käsitellään jatkossa pelkästään luvussa 4 esitettävässä L-hierarkiassa, niin alla on esitetty lyhyesti vain E0L-systeemien formaali määritelmä, sekä yksi esimerkki E0L-kielestä. E0L-systeemeistä ja niiden suhteesta Chomskyn kielioppeihin kiinnostuneelle lähde [16] on erinomainen kirja lisätietojen hankkimiselle. Myöskin lähde [39] on tutustumisen arvoinen.

E0L-systeemi on järjestelmä $G = \langle V, \omega, P, V_1 \rangle$, missä $G' = \langle V, \omega, P \rangle$ on 0L-systeemi ja $V_1 \subset V$. E0L-systeemin G tuottama kieli L(G) määritellään $L(G) = L(G') \cap V_1^*$, ja sitä kutsutaan usein myös E0L-kieleksi. [16]

E0L-systeemiä $G = \langle V, \omega, P, V_1 \rangle$ ja sen tuottamaa kieltä L(G) sanotaan propagoivaksi (vastaavasti deterministiseksi) jos ja vain jos vastaava 0L-systeemi $\langle V, \omega, P \rangle$ on propagoiva (vastaavasti deterministinen). [16]

Kieli $L = \{a^n b^n a^n \mid n > 0\}$ on E0L-kieli (vieläpä PE0L-kieli), kun L = L(G), missä $G = \langle V, \omega, P, V_1 \rangle$, $V = \{S, A, B, A', B', F, a, b\}$, $\omega = S$, $V_1 = \{a, b\}$ ja $P = \{S \rightarrow ABA, A \rightarrow AA', A \rightarrow a, B \rightarrow BB', B \rightarrow b, A' \rightarrow A', A' \rightarrow a, B' \rightarrow B', B' \rightarrow b, F \rightarrow F, a \rightarrow F, b \rightarrow F\}$. [16]

Kuvassa 7 on esitettynä kaksi em. E0L-systeemin tavanomaista derivaatiopuuta.



Kuva 7. E0L-systeemin derivaatiopuita.

Kuvasta 7 nähdään, että vasemmasta derivaatiopuusta löytyy kolmannen iteraation jälkeen lause aabbaa, joka kuuluu kieleen L(G). Sen sijaan kuvan oikea derivaatiopuu on esimerkki "väärin" menneestä derivaatiosta, eli tässä tapauksessa ei saatu aikaan kieleen L(G) kuuluvaa lausetta. Ainoa merkkiin F liittyvä sääntö ko. E0L-systeemissä on F→F, eli kuvan 7 oikean derivaatiopuun tapauksessa ei ole mahdollista saada aikaan kieleen L(G) kuuluvaa lausetta vaikka iteraatioita jatkettaisiin loputtomiin. [16]

Monien biologisten organismien kehittyminen riippuu monista organismia ympäröivistä tekijöistä, kuten esimerkiksi kosteudesta, valaistuksesta, lämpötilasta, jne. T0L-systeemissä tällaisen organismin kehityksen kuvaamiseen sisällytetään monia erilaisia sääntöjoukkoja kuvaamaan erilaisia tiloja organismia ympäröivissä tekijöissä.

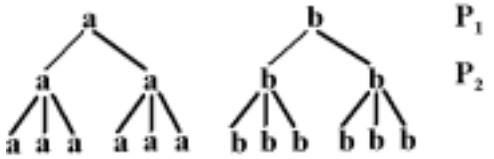
T0L-systeemit käsitellään tässä työssä varsin pintapuolisesti samasta syystä kuin edellä esitetyt E0L-systeemitkin. T0L-systeemeistä lisätietoa haluaville suositellaan jälleen lähdettä [16]. Seuraavassa on esitetty T0L-systeemien formaali määritelmä [29].

T0L-systeemi on järjestelmä $G_T = \langle V, \omega, P, T \rangle$, missä V, ω ja P on määritelty kuten 0L-systeemeissä ja T on taulukoksi kutsuttu kokoelma P:n osajoukkoja. Jokaista taulukkoa $t \in T$ ja merkkiä $a \in V$ kohti oletetaan olevan vähintään yksi sellainen sääntö $p \in t$, jonka vasen puoli on a.

Lause ν on johdettu suoraan μ:stä T0L-systeemissä $G_T = \langle V, \omega, P, T \rangle$, jos ν on johdettu suoraan μ:stä 0L-systeemissä $G = \langle V, \omega, P \rangle$ ja kaikki ko. johdossa käytetyt säännöt kuuluvat samaan taulukkoon $t \in T$. Yhden iteraatiokierroksen aikana käytettyjen sääntöjen täytyy kuulua samaan taulukkoon, mutta eri iteraatiokierroksilla voidaan soveltaa eri taulukoiden sääntöjoukkoja. [29]

T0L-systeemeillä tehtävät derivaatiot täytyy määritellä formaalisti, jotta merkkijono muodostuu aina samanlaiseksi, kun tietyn T0L-systeemin tiettyyn aksioomaan sovelletaan ko. systeemin sääntöjä. Näin tulee määritellyksi tapa, jolla valitaan ensin kullakin iteraatiokierroksella sovellettava sääntöjoukko, ja sitten sovellettava sääntöjoukon sääntö kuhunkin käsiteltävän lauseen merkkiin. Tämä formalisointi ei ole tämän työn kannalta kiinnostava, joten se sivuutetaan. Tarvittaessa tämä formalisointi löytyy kuitenkin lähteestä [16, s.115].

Kuvassa 8 esitetään derivaatiopuu T0L-systeemillä tuotetusta merkkijonosta. Esimerkki on lähteestä [16], jossa on täsmällisesti määritelty myöskin esimerkin derivaatio ([16, s.116]). Tässä esimerkissä kuitenkin tyydytään toteamaan, että valinta käytettäväksi sääntöjoukoksi on tehty "täsmällisesti". Kullakin iteraatiokierroksella käytetty sääntöjoukko on merkitty derivaatiopuun oikeaan reunaan. Esimerkin T0L-systeemi $G_T = \langle\{a, b\}, ab, P, \{P_1, P_2\}\rangle$, missä $P_1 = \{a \rightarrow a^2, b \rightarrow b^2\}$, $P_2 = \{a \rightarrow a^3, b \rightarrow b^3\}$ ja P on tietenkin $P_1 \cup P_2$.



Kuva 8. T0L-systeemillä tuotetun merkkijonon derivaatiopuu.

3.3 KONTEKSTISET L-SYSTEEMIT (IL-SYSTEEMIT)

Kontekstittomissa L-systeemeissä sääntöjä voidaan soveltaa riippumatta säännön vasemman puolen kontekstista. Monilla sovellusalueilla on kuitenkin välttämätöntä kiinnittää huomiota myöskin sovellettavan säännön vasemman puolen kontekstiin (esimerkiksi simuloitaessa vuorovaikutusta kasvin eri osien kesken).

Jokaisella kontekstisen L-systeemin säännöllä voi olla vasen ja/tai oikea konteksti. Sellaista systeemiä, jolla on molemmat kontekstit, kutsutaan 2L-systeemiksi, ja sellaista systeemiä, jolla on vain jompikumpi konteksti, kutsutaan 1L-systeemiksi. 2L-systeemien säännöt ovat muotoa $a_l < a > a_r \rightarrow \chi$, missä merkki a (*strict predecessor*) voi tuottaa χ:n jos ja vain jos a:ta edeltää a_l ja seuraa a_r. 1L-systeemien säännöt ovat vastaavasti joko muotoa $a_l < a \rightarrow \chi$ tai $a > a_r \rightarrow \chi$. [33, 38]

0L-systeemit, 1L-systeemit ja 2L-systeemit kuuluvat laajempaan IL-systeemien luokkaan (I = with interactions), jota kutsutaan usein myös (k, l)-systeemiksi. (k, l)-systeemissä vasen konteksti on k:n mittainen lause ja oikea konteksti l:n mittainen lause. [38]

Seuraavassa esimerkissä simuloidaan signaalin propagointia 1L-systeemin avulla [38]. Tässä esimerkissä oletetaan, että kontekstisilla säännöillä on korkeampi presedenssiarvo kuin kontekstittomilla säännöillä, eli jos tiettyyn merkkiin voitaisiin soveltaa sekä kontekstista että kontekstitonta sääntöä, niin sovellettavaksi säännöksi valitaan kontekstinen sääntö. Tarkastellaan L-systeemiä

$$\begin{aligned} &\omega : baaaaaa \\ &p_1 : b < a \rightarrow b \qquad (3) \\ &p_2 : \quad b \rightarrow a. \end{aligned}$$

Kun tämän L-systeemin (3) sääntöjä aletaan soveltaa aksioomaan, saadaan

baaaaaa
a**b**aaaaa
aa**b**aaaa
aaa**b**aaa
aaaa**b**aa
... .

Eli merkki b liikkuu lauseessa vasemmalta oikealle, ikään kuin välittäen viestikapulaa lauseen vasemmasta reunasta kohti oikeaa reunaa.

IL-systeemien formaali määritelmä ei ole tämän työn kannalta mitenkään erityisen tärkeä, mutta koska sitä tarvitaan apulauseen 4.15 todistuksessa, niin määritelmä annetaan tässä yhteydessä.

Olkoot k ja l ei-negatiivisia kokonaislukuja. Nyt (k, l)-systeemi on järjestetty nelikko

$$G = \langle V, \omega, P, g \rangle,$$

missä

V on äärellinen ei-tyhjä joukko (G:n aakkosto),
ω on V^*:n alkio (G:n aksiooma),

g on alkio, joka ei kuulu joukkoon V,
P on äärellinen ei-tyhjä relaatio (G:n sääntöjoukko)

$$P \subset (V \cup \{g\})^k \times V \times (V \cup \{g\})^l \times V^*,$$

jolle pätee, että

(1) jos $\langle w_1, a, w_2, w_3 \rangle \in P$,

niin

(1.1) jos $w_1 = w_{11}gw_{12}$ joillekin $w_{11}, w_{12} \in (V \cup \{g\})^*$, niin $w_1 \in \{g\}^*$
(1.2) jos $w_2 = w_{21}gw_{22}$ joillekin $w_{21}, w_{22} \in (V \cup \{g\})^*$, niin $w_2 \in \{g\}^*$

ja

(2) jokaista $\langle w_1, a, w_2 \rangle \in (V \cup \{g\})^k \times V \times (V \cup \{g\})^l$ (w_1 ja w_2 täyttävät kohtien (1.1) ja (1.2) ehdot) kohti on olemassa w_3 ($\in V^*$), jolle pätee

$\langle w_1, a, w_2, w_3 \rangle \in P$.

Jos $\langle w_1, a, w_2, w_3 \rangle$ on P:n alkio, niin tämä voidaan merkitä muodossa $\langle w_1, a, w_2 \rangle \rightarrow w_3$.

Kohdassa 7.1 esitetään lisäksi esimerkki kontekstisesta puu L -systeemistä sekä kohdassa 7.2 kontekstisesta pino L -systeemistä, kun ensin vastaavissa kohdissa on määritelty tavalliset puu 0L -systeemit ja pino 0L -systeemit.

3.4 STOKASTISET L-SYSTEEMIT

Stokastisissa L-systeemeissä sovellettava sääntö valitaan sääntöjoukosta P sääntöihin merkittyjen todennäköisyyksien perusteella. Todennäköisyydet merkitään sääntöihin "$\rightarrow$" –merkin yläpuolelle. Stokastisten L-systeemien avulla tuotettuihin malleihin saadaan eroavaisuuksia sääntöjen todennäköisyyksien perusteella, vaikka tuotettuihin malleihin onkin sovellettu samaa sääntöjoukkoa P. Esimerkki malleihin saatavista eroavaisuuksista on esitetty myöhemmin kohdan 7.2 kuvassa 22.

Stokastinen L-systeemi on järjestetty nelikko $G_\pi = \langle V, \omega, P, \pi \rangle$. Aakkosto V, aksiooma ω ja sääntöjoukko P on määritelty kuten D0L-systeemeissä. Funktion $\pi : P \rightarrow (0, 1]$ (todennäköisyysjakauma) avulla sääntöjoukosta P muodostetaan sääntötodennäköisyydet (*production probabilities*). Jokaista merkkiä $a \in V$ koskevien sääntöjen yhteenlaskettu todennäköisyys on 1. [29, 38]

G_π:n johtoa $\mu \Rightarrow \nu$ kutsutaan stokastiseksi johdoksi, jos sellaisen säännön p, jonka vasen puoli on a, soveltamisen todennäköisyys jokaiseen lauseen μ merkkiin a = π(p) [29, 38]. Tietyssä lauseessa tietyn merkin eri ilmentymiin voidaan siis soveltaa eri sääntöjä samalla iteraatiokierroksella.

Lähteessä [5] laajennetaan tavallisen stokastisen L-systeemin määritelmää ns. kaksois-stokastisiin L-systeemeihin (*doubly stochastic L-system*, *dsL-system*). DsL-systeemit koostuvat kahdesta osasysteemistä: ylemmän tason stokastisesta L-systeemistä ja alemman tason perus-L-systeemistä. Tärkein periaatteellinen ero tavallisten stokastisten L-systeemien ja dsL-systeemien välillä on se, että dsL-systeemeissä myös todennäköisyysjakaumaa π voidaan muuttaa sääntöjen avulla stokastisesti eri iteraatiokierrosten välillä. Tavallisissa stokastisissa L-systeemeissähän π on vakio. Tässä työssä ei käsitellä dsL-systeemeitä kuitenkaan tämän enempää, mutta lisätietoa asiasta löytyy tarvittaessa esimerkiksi lähteestä [5].

Stokastisten L-systeemien avulla tuottettujen merkkijonojen tulkinnasta on esitetty esimerkki kohdassa 7.2

3.5 PARAMETROIDUT 0L-SYSTEEMIT

Parametroiduissa L-systeemeissä hyödynnetään parametroituja lauseita (*parametric words*), jotka ovat kirjaimista ja niihin liittyvistä parametreista koostuvia merkkijonomoduuleita. Kirjaimet ovat aakkoston V merkkejä ja parametrit ovat reaalilukuja (reaalilukujen joukkoa merkitään tässä työssä symbolilla $\Re$). Moduulia, joka koostuu kirjaimesta $A \in V$ ja parametreista $a_1, a_2, \ldots, a_n \in \Re$, merkitään $A(a_1, a_2, \ldots, a_n)$. Jokainen moduuli kuuluu joukkoon $M = V \times \Re^*$, missä $\Re^*$ on kaikkien äärellisten parametrisekvenssien joukko. Merkkijonomoduulien joukkoa merkitään vastaavasti $M^* = (V \times \Re^*)^*$ ja ei-tyhjien merkkijonojen joukkoa $M^+ = (V \times \Re^*)^+$. [32, 34, 38]

Lauseiden arvotetut todelliset parametrit ovat sidoksissa L-systeemin sääntöjen formaaleihin parametreihin. Jos formaalien parametrien joukosta käytetään merkintää Σ, niin $C(\Sigma)$ tarkoittaa sellaista loogista ilmausta, jonka parametrit kuuluvat joukkoon Σ, ja $E(\Sigma)$ tarkoittaa vastaavasti sellaista aritmeettista ilmausta, jonka parametrit kuuluvat joukkoon Σ. Molemmantyyppiset ilmaukset koostuvat formaaleista parametreista ja numeerisista vakioista, jotka on sidottu yhteen aritmeettisten operaattoreiden (+, -, *, /), eksponentti-operaattorin (^), relationaalisten operaattoreiden (<, >, =), loogisten operaattoreiden (!, &) ja sulkeiden () avulla. Ilmauksissa voidaan käyttää myöskin matemaattisia standardifunktioita, kuten esimerkiksi luonnollista logaritmia tai trigonometrisia funktioita [32, 33].

Ilmauksissa käytettävissä olevat operaattorisymbolit sekä säännöt ilmausten syntaksin oikeellisuuden tarkistamiseen ovat samat kuin C-ohjelmointikielessä [32, 33]. Syntaktisesti oikeinmuodostettujen loogisten ilmausten joukkoa merkitään $\delta(\Sigma)$ ja vastaavien aritmeettisten ilmausten joukkoa $\varepsilon(\Sigma)$. [38]

Parametroitu 0L-systeemi on järjestetty nelikko $G = \langle V, \Sigma, \omega, P \rangle$, missä V on systeemin aakkosto, Σ on formaalien parametrien joukko, $\omega \in (V \times \Re^*)^+$ on aksioomaksi kutsuttu eityhjä parametroitu lause ja $P \subset (V \times \Sigma^*) \times \delta(\Sigma) \times (V \times \varepsilon(\Sigma))^*$ on äärellinen sääntöjen joukko. [32, 33, 34, 38]

Symboleja ":" ja "$\rightarrow$" käytetään erottamaan säännön kolme osaa toisistaan: säännön vasen puoli, ehto ja säännön oikea puoli. Säännöt ovat muotoa

säännön vasen puoli (parametrit): ehto $\rightarrow$ säännön oikea puoli (looginen tai aritmeettinen ilmaus).

Esimerkiksi sääntö, jonka vasen puoli on A(t), ehto t>5 ja oikea puoli B(t+1)CD(t^0.5, t-2) kirjoitetaan muodossa

A(t): $t > 5 \rightarrow$ B(t + 1)CD(t ^ 0.5, t-2). [32, 33, 34, 38] (4)

Sääntöä voidaan soveltaa parametroidun lauseen moduuliin, jos seuraavat ehdot täyttyvät:

- moduulin kirjain on sama kuin säännön vasemman puolen kirjain,
- todellisten parametrien lukumäärä on sama kuin formaalien parametrien lukumäärä ja
- ehto arvottuu todeksi, jos säännön formaalit parametrit korvataan todellisilla parametreilla. [32, 33, 34, 38]

Kun parametroidun lauseen moduuliin sovelletaan systeemin jotakin sääntöä, niin tällöin moduulin todelliset parametrit sijoitetaan moduuliin sovellettavaan sääntöön. Esimerkiksi sääntöä (4) voidaan soveltaa moduuliin A(9), koska moduulissa oleva kirjain A on sama kuin säännön vasemman puolen kirjain, todellisten parametrien määrä moduulissa A(9) on sama kuin säännön vasemman puolen A(t) formaalien parametrien määrä ja koska ehto $t > 5$ arvottuu todeksi, kun formaali parametri t korvataan todellisella parametrilla 9 ($9 > 5$). Eli nyt moduulin A(9) todellinen parametri 9 voidaan sijoittaa säännön oikean puolen formaalien parametrien paikalle, jolloin muodostuu parametroitu lause B(9 + 1)CD(9 ^ 0.5, 9-2), eli B(10)CD(3,7). [32, 33, 34, 38]

Jos *moduuli* a *tuottaa* parametroidun lauseen χ, kun siihen sovelletaan jotakin parametroidun L-systeemin G sääntöä, niin se merkitään a $\Rightarrow$ χ. Oletetaan mielivaltainen parametroitu lause $\mu = a_1 \ldots a_m$ annetuksi. Tällöin lause $\nu = \chi_1 \ldots \chi_m$ on johdettu suoraan μ:stä (merk. $\mu \Rightarrow \nu$) jos ja vain jos $a_i \Rightarrow \chi_i$ jokaiselle i = 1, ..., m. Lause ν on johdettavissa n:n pituisella johdolla L-systeemin G avulla, jos on olemassa sellainen lausesekvenssi $\mu_0, \mu_1, \ldots, \mu_n$, että $\mu_0 = \omega$, $\mu_n = \nu$ ja $\mu_0 \Rightarrow \mu_1 \Rightarrow \ldots \Rightarrow \mu_n$.

Annetaan seuraavaksi yo. määritelmään liittyen esimerkki parametroidun 0L-systeemin soveltamisesta. Kun parametroidun 0L-systeemin

ω : B(2)A(4,4)
p_1 : A(x,y) : $y \leq 3 \rightarrow$ A(x * 2, x + y)
p_2 : A(x,y) : $y > 3 \rightarrow$ B(x)A(x/y, 0) (5)
p_3 : B(x) : $x < 1 \rightarrow$ C
p_4 : B(x) : $x \geq 1 \rightarrow$ B(x - 1)

sääntöjä sovelletaan aksioomaan neljä kertaa saadaan seuraava lausesekvenssi

μ_0 : B(2)A(4,4)
μ_1 : B(1)B(4)A(1,0) (sovellettiin sääntöjä 4 ja 2)
μ_2 : B(0)B(3)A(2,1) (sovellettiin sääntöjä 4, 4 ja 1)
μ_3 : CB(2)A(4,3) (sovellettiin sääntöjä 3, 4 ja 1)
μ_4 : CB(1)A(8,7) (sovellettiin sääntöjä 4 ja 1).

Aivan kuten aiemminkin esitetyissä L-systeemien määritelmissä, myöskin tässä yhteydessä jokainen merkki korvaa iteraatioissa itsensä, jos L-systeemin säännöissä ei ole eksplisiittisesti esitettyä sääntöä jollekin merkille. Tässä esimerkissä siis C korvaa itsensä neljännessä iteraatiossa.

Edellinen esimerkki on esitetty kaikissa neljässä lähteessä [32, 33, 34, 38], mutta kaikkiin niistä on lipsahtanut pieni huolimattomuusvirhe. Nimittäin lähteissä [32, 33] on sovellettu väärää sääntöä neljännessä iteraatiossa moduuliin A(4,3). Lähteissä [34, 38] on kyllä sovellettu em. kohtaan oikeaa sääntöä, mutta parametrien arvotus on tehty väärin x:n osalta.

Luvussa 6 on lisäksi esitetty vielä yksi esimerkki parametroidun D0L-systeemin käytöstä. Determinismi tarkoittaa parametroitujenkin L-systeemien kohdalla samaa asiaa kuin

muissakin L-systeemeissä, eli parametroiduissa D0L-systeemeissä kuhunkin moduuliin voidaan soveltaa täsmälleen yhtä sääntöä.

3.6 PARAMETROIDUT 2L-SYSTEEMIT

Edellisen kohdan L-systeemien säännöt ovat kontekstittomia, eli sääntöä voidaan soveltaa parametroidun lauseen moduuliin riippumatta säännön vasemman puolen kontekstista. Jos parametroidun lauseen peräkkäisten moduulien välille halutaan mallintaa riippuvuus-suhteita, täytyy parametroidun L-systeemin säännöt muuttaa kontekstisiksi samaan tapaan kuin tavallisissa kontekstisissa L-systeemeissä. Parametroitujen 2L-systeemien säännöt vastaavat syntaksiltaan tavallisten 2L-systeemien sääntöjä. Esimerkiksi kontekstista sääntöä

$$A(x) < B(y) > C(z) : x + y + z > 10 \rightarrow E((x + y) / 2)F((y + z) / 2) \qquad (6)$$

voidaan soveltaa parametroidun lauseen A(4)B(5)C(6) moduuliin B(5), koska parametroidun lauseen kirjainsekvenssi A, B, C on sama kuin säännössä (6), formaalien ja todellisten parametrien lukumäärä on sama ja ehto $x + y + z > 10$ arvottuu todeksi $(4 + 5 + 6 > 10)$. Kun sääntöä (6) sovelletaan moduuliin B(5), muodostuu moduulit E(4.5)F(5.5). Samalla iteraatiokierroksella sovelletaan tietenkin moduuleihin A(4) ja C(6) niihin sopivia sääntöjä. [32, 34, 38]

Jos sääntöjoukko P sisältää enemmän kuin yhden säännön, jota voidaan soveltaa tiettyyn moduuliin, täytyy L-systeemiin sisällyttää mekanismi, jolla valitaan moduuliin sovellettava sääntö. Parametroiduissa stokastisissa L-systeemeissä tämä valinta tehdään samalla periaatteella kuin tavallisissa stokastisissa L-systeemeissä, eli siis sääntöihin sisällytettyjen todennäköisyyksien perusteella. Parametroitujen stokastisten 2L-systeemien säännöt ovat täydellisimmillään muotoa

$$\text{id: } a_l < a > a_r : \text{ehto} \rightarrow \chi : \pi,$$

missä id tarkoittaa säännön tunnistetta ja π todennäköisyyskerrointa (*probability factor*). Todennäköisyyskerroin on ei-negatiivisen luvun palauttava aritmeettinen ilmaus. Symbolit a_l, a, a_r, ehto ja χ on määritelty kuten edellä kontekstisten, stokastisten sekä parametroitujen 0L-systeemien yhteydessä. Jos $P' \subseteq P$ on sääntöjoukko, jonka sääntöjä voidaan soveltaa tiettyyn moduuliin $A(a_1, a_2, \ldots, a_n) \in V \times \Re^*$, niin tietyn säännön $p_k \in P'$ soveltamisen todennäköisyys tähän moduuliin määritellään

$$prob(p_k) = \frac{\pi(p_k)}{\Sigma_{p_i \in P'} \pi(p_i)}. \qquad (7)$$

Yleisesti ottaen yllä esitetty todennäköisyys ei ole tiettyyn sääntöön liitetty vakio, vaan todennäköisyys voi riippua parametrien arvotuksista sekä moduulin kontekstista. [34]

Esimerkiksi seuraava parametroitu L-systeemi on sekä kontekstinen että stokastinen [34]:

$$\begin{aligned} &\omega : A(1)B(3)A(5) \\ &p_1 : A(x) \rightarrow A(x + 1) : 2 \\ &p_2 : A(x) \rightarrow B(x - 1) : 3 \\ &p_3 : A(x) : x > 3 \rightarrow C(x) : x \\ &p_4 : A(x) < B(y) > A(z) : y < 4 \rightarrow B(x + z)A(y). \end{aligned} \qquad (8)$$

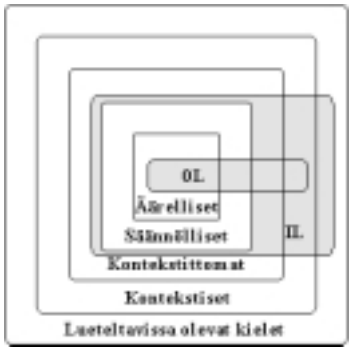
Moduuliin A(x) voidaan soveltaa tilanteesta riippuen sääntöjä p_1, p_2 tai p_3. Jos parametrin x arvo on pienempi tai yhtä suuri kuin 3, niin moduuliin A(x) voidaan soveltaa ainoastaan kahta ensimmäistä sääntöä. Sääntöjen soveltamisen todennäköisyys lasketaan tällöin kaavan (7) mukaan: $prob(p_1) = 2 / (2+3) = 0.4$ ja $prob(p_2) = 3 / (2+3) = 0.6$. Jos parametrin x arvo on suurempi kuin 3, niin tällöin myös sääntöä 3 voidaan soveltaa moduuliin A(x). Tässä tapauksessa sääntöjen soveltamisen todennäköisyys riippuu myöskin x:n arvosta. Jos x on esimerkiksi 5, niin todennäköisyydet lasketaan seuraavasti: $prob(p_1) = 2 / (2+3+5) = 0.2$, $prob(p_2) = 3 / (2+3+5) = 0.3$ ja $prob(p_3) = 5 / (2+3+5) = 0.5$. Kontekstista sääntöä p_4 voidaan soveltaa moduuliin B(y), jos ko. moduulin vasempana kontekstina on moduuli A(x), oikeana kontekstina moduuli A(z) ja jos ehto $y < 4$ arvottuu todeksi. Tällöin moduuli B(y) korvataan moduuleilla B(x + z)A(y). [34]

Jos L-systeemin (8) aksioomaan A(1)B(3)A(5) aletaan soveltaa ko. L-systeemin sääntöjä, niin ensimmäisen iteraation tuloksena voidaan saada parametroitu lause A(2)B(6)A(3)C(4). Tällöin moduuliin A(1) on sovellettu sääntöä p_1 (siihen olisi voitu soveltaa myös sääntöä p_2) ja moduuliin A(5) on sovellettu sääntöä p_3 (tässä olisi voitu soveltaa myöskin sääntöjä p_1 tai p_2). Sovellettavat säännöt on siis valittu satunnaisesti todennäköisyyskerrointa käyttäen. Moduuliin B(3) sovellettiin sääntöä p_4, koska ko. moduulin molemmat kontekstit sopivat tähän sääntöön, ja koska ehto $y < 4$ arvottuu todeksi. [34]

4. L-SYSTEEMIEN HIERARKIASTA

Lindenmayer esitteli L-systeeminsä siis vuonna 1968. Siitä lähtien L-systeemejä tutkittiin erittäin paljon myöskin teoreettisesta näkökulmasta. Näiden tutkimusten tuloksena syntyi ns. L-hierarkia, joka voidaan rinnastaa Chomskyn kielihierarkiaan. L-hierarkian ytimen muodostavat neljä L-systeemiperhettä, nimittäin D0L-, 0L-, E0L- ja ET0L-systeemiperheet. Näiden L-systeemiperheiden muodostamaa hierarkiaa käytetään nykyään apuna samaan tapaan kuin Chomskyn kielihierarkiaa, kun tutkitaan jotakin uutta mekanismia tai ilmiötä formaalien kielten näkökulmasta [20].

Chomskyn kieliopeissa sääntöjä sovelletaan siis peräkkäin ja L-systeemeissä rinnakkain. Tämä ero aiheuttaa olennaisia eroja myöskin L-systeemien ominaisuuksiin verrattuna vastaaviin Chomskyn kielioppeihin. Esimerkiksi on olemassa sellaisia kieliä, jotka voidaan tuottaa kontekstittomilla L-systeemeillä (siis 0L-systeemeillä), mutta ei kontekstittomilla Chomskyn kieliopeilla (ks. kuva 9) [36]. Esimerkiksi kieli $\{a^{2^n} \mid n \geq 0\}$ on tällainen kieli, kuten lähteessä [16] todetaan. Kuvasta 9 nähdään myös se, että on olemassa sellaisia kieliä, jotka voidaan tuottaa IL-systeemeillä, mutta ei kontekstisilla Chomskyn kieliopeilla (teoreema 10.7 lähteessä [16]).

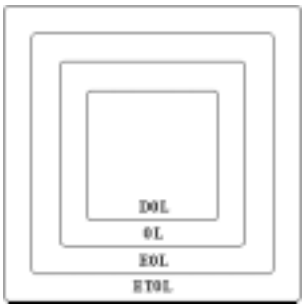


Kuva 9. 0L-systeemit ja IL-systeemit suhteessa Chomskyn kielihierarkiaan.

Varsinainen L-hierarkia on esitetty lauseessa 4.1 sekä kuvassa 10. Lausetta ei tässä työssä todisteta, mutta lauseen tulokset on todistettu lähteessä [16].

Lause 4.1.
$L_{D0L} \subsetneqq L_{0L} \subsetneqq L_{E0L} \subsetneqq L_{ET0L}$.

Todistus.
Sivuutetaan. ($L_{D0L} \subsetneqq L_{0L}$ [16, teoreema 2.8], loppuosa lauseesta [16, teoreema 10.10]).



Kuva 10. L-hierarkia.

Kuten kohdassa 2.1 perustellaan, L-systeemejä kannatta yrittää suhteuttaa Chomskyn kielihierarkiaan. Lähteen [16] teoreemassa 10.10 on tehty juuri näin. Ko. teoreeman sisältö on esitetty kuvassa 11. Kuvassa nuolella esitetään kieliperheen aito sisältyminen toiseen kieliperheeseen. Jos kahden kieliperheen välillä ei ole polkua, niin kumpikaan kieliperhe ei kokonaisuudessaan sisällä toista, mutta niillä on kuitenkin yhteisiä alkioita. Ennen kuvan 11 esittämistä annetaan kuitenkin kuvan kannalta tarpeelliset apulauseet 4.1 – 4.17.

Apulause 4.1.
$L_{RG} \subsetneqq L_{CF}$.

Todistus.
Tämän työn lause 2.1.

Apulause 4.2.
$L_{0L} \subsetneqq L_{T0L}$.

Todistus.
Sivuutetaan. ([16, s. 203]).

Apulause 4.3.
$L_{0L} \subsetneqq L_{IL}$.

Todistus.
Sivuutetaan. ([16, s. 203]).

Apulause 4.4.
$L_{0L} \subsetneqq L_{E0L}$.

Todistus.
Sivuutetaan. ([16, s. 203]).

Apulause 4.5.
$L_{CF} \subsetneq L_{E0L}$.

Todistus. [16]
Olkoon L kontekstiton kieli, joka on tuotettu kontekstittomalla kieliopilla $G = \langle N, T, P, S \rangle$, ja olkoon $H = \langle N \cup T, S, P', T \rangle$ sellainen E0L-systeemi, että $P' = P \cup \{a \rightarrow a \mid a \in N\}$. Nähdään helposti, että L(G) = L(H). Tästä seuraa, että jokainen kontekstiton kieli on myös E0L-kieli.

Lisäksi täytyy todistaa, että on olemassa sellainen E0L-kieli, joka ei ole kontekstiton. Tämä voidaan tehdä tarkastelemalla 0L-kieltä $\{a^{2^n} \mid n \geq 0\}$. Tämä kieli ei ole kontekstiton, mikä nähdään helposti Bar-Hillelin lemman avulla ([16, s. 60], [25, s. 48]). Apulauseen 4.4 perusteella tiedetään, että jokainen 0L-kieli on myös E0L-kieli. □

Apulause 4.6.
$L_{T0L} \subsetneq L_{ET0L}$.

Todistus.
Sivuutetaan. ([16, s. 203]).

Apulause 4.7.
$L_{IL} \subsetneq L_{EIL}$.

Todistus.
Sivuutetaan. ([16, s. 203]).

Apulause 4.8.
$L_{E0L} \subsetneq L_{ET0L}$.

Todistus.
Sivuutetaan. ([16, s. 203]).

Apulause 4.9.
$L_{ET0L} \subsetneq L_{CS}$.

Todistus.
Sivuutetaan. ([16, s. 203]).

Apulause 4.10.
$L_{CS} \subsetneq L_{RE}$.

Todistus.
Tämän työn lause 2.1.

Apulause 4.11.
$L_{EIL} = L_{RE.}$
Todistus.
Sivuutetaan. ([16, s. 205]).

Apulause 4.12.
$L_{RG} \subset L_{T0L.}$
Todistus.
Sivuutetaan. ([16, s. 202]). Tosin triviaaliratkaisuna voidaan todeta, että *tyhjä kieli* $\Phi \in L_{RG}$, mutta $\Phi \notin L_{RG}$.

Apulause 4.13.
$L_{RG} \subset L_{IL.}$
Todistus.
Sivuutetaan. ([16, s. 202]). Jälleen $\Phi \in L_{RG}$, mutta $\Phi \notin L_{IL}$.

Apulause 4.14.
$L_{0L} \subset L_{CF.}$
Todistus.
Ks. apulauseen 4.5 todistus.

Apulause 4.15.
$L_{T0L} \subset L_{IL.}$
Todistus. [16]
Väite voidaan ilmaista muodossa "on olemassa sellainen T0L-kieli, joka ei ole IL-kieli". Tarkastellaan kieltä $L = \{a^{2^u 3^v} \mid u, v \geq 0\}$. L on T0L-kieli, koska se voidaan tuottaa T0L-systeemillä $\langle\{a\}, a, \{a \rightarrow a^2, a \rightarrow a^3\}, \{\{a \rightarrow a^2\}, \{a \rightarrow a^3\}\}\rangle$.

Väitteen todistamiseksi täytyy vielä osoittaa, että L ei ole IL-kieli. Tehdään vastaoletus, että L on (k, l)-systeemin $G = \langle V, \omega, P, g\rangle$ tuottama IL-kieli. Tämä vastaoletus johtaa ristiriitaan seuraavasti:

(i) Mitä tahansa kokonaislukua K kohden on olemassa vain äärellinen määrä ei-negatiivisista luvuista koostuvia järjestettyjä pareja $\langle p, q\rangle$ joille pätee $2^q - 3^p = K$. (Tulos on peräisin lukuteoriasta).

(ii) Mitä tahansa positiivista kokonaislukua K kohden on olemassa vain äärellinen määrä ei-negatiivisista luvuista koostuvia järjestettyjä pareja ⟨p, q⟩, joille pätee $|2^q - 3^p| \leq K$. Tämä tulos on suora seuraus kohdasta (i).

(iii) Mitä tahansa positiivista kokonaislukua K kohden on olemassa vain äärellinen määrä kielen L lausepareja a^n ja a^t, joille pätee n - t = K. Tämä johtuu siitä, että jos a^n ja a^t ovat kielen L lauseita, niin $n = 2^{u_1}3^{v_1}$ ja $t = 2^{u_2}3^{v_2}$ joillakin u_1, u_2, v_1 ja v_2. Olkoot $u = \min\{u_1, u_2\}$ ja $v = \min\{v_1, v_2\}$. Kun n, t ja K jaetaan luvulla 2^u3^v, niin huomataan, että jos kielen L lausepareja a^n ja a^t, joille pätee n - t = K, olisi ääretön määrä, niin ainakin toinen seuraavista väittämistä olisi totta:

1. On olemassa ääretön määrä lukuja p ja q joille pätee $|2^q3^p - 1| \leq K$.
2. On olemassa ääretön määrä lukuja p ja q joille pätee $|2^q - 3^p| \leq K$.

Ensimmäinen väittämä on selvästi epätosi. Myös toinen väittämä on epätosi kohdan (ii) perusteella, eli kohdan (iii) väite on todistettu.

(iv) Olkoot m = k + l ja $L_m = \{a^{2^u3^v} \mid 2^u3^v > m\}$. Jos kahta sääntöä $\langle \alpha, a, \beta\rangle \rightarrow a^{z1}$ ja $\langle \alpha, a, \beta\rangle \rightarrow a^{z2}$ voidaan soveltaa L_m:n lauseeseen, niin z1 = z2. Tämä johtuu siitä, että jos z2 > z1 (tilanne z1 > z2 on analoginen tämän tilanteen kanssa), niin silloin olisi olemassa ääretön määrä sellaisia L:n lauseita a^y, joille pätee, että myös $a^{y+(z2-z1)}$ on L:n lause. Tämä on puolestaan ristiriidassa kohdan (iii) kanssa.

(v) On olemassa sellainen vakio F>1 jolle pätee, että jos $\langle a^k, a, a^l\rangle \rightarrow a^z$ on P:n sääntö, niin z = F. Tämä on suora seuraus kohdista (iii) ja (iv) sekä siitä, että L on ääretön kieli.

(vi) On olemassa sellainen vakio C jolle pätee, että jos a^u on johdettu G:ssä suoraan lauseesta a^{k+1}, niin u = C. Tämä on suora seuraus kohdasta (iv).

(vii) Jos lause a^y on johdettu L_m:ssä suoraan lauseesta $a^{2^u3^v}$, niin y on muotoa $(2^u3^v - m) * F + C$, missä m = k + l ja F sekä C ovat kohdissa (v) ja (vi) määritetyt vakiot. Jos joukko A määritellään

$$A = \begin{cases} \{\omega\},\ jos\ \mathrm{m} < |\,\omega\,|, \text{muutoin} \\ \{x \in L(G) \mid \mathrm{m} \leq |\,\mathrm{x}\,| \text{ ja L(G) : ssä on sellainen lause y, jolle pätee} \\ \qquad\qquad |\,\mathrm{y}\,| < \mathrm{m} \text{ ja x on johdettu G : ssä suoraan y : stä}\}, \end{cases}$$

niin tällöin jokainen L_m:n alkio on johdettu A:n alkiosta soveltamalla L_m:n lauseisiin toistuvasti sopivia (sovellettavissa olevia) G:n sääntöjä.

Näin ollen on olemassa äärettömän monta positiivista kokonaisluvuista p ja q koostuvaa lukuparia, joita kohti on olemassa sellaiset positiiviset kokonaisluvut u ja v, joille pätee $\langle u, v\rangle \neq \langle p, q\rangle$ ja $a^{2^p3^q}$ on johdettavissa suoraan $a^{2^u3^v}$:stä G:ssä. Näin ollen

$$(2^u3^v - m)F + C = 2^p3^q,$$

ja edelleen

$$F2^u3^v - 2^p3^q = mF - C.$$

Yllä olevassa yhtälössä joko u:n tai v:n täytyy kasvaa, jos 2^p3^q kasvaa. Mutta koska luvun $mF - C$ täytyy olla jaollinen luvulla 2^r3^s, missä $r = \min\{u, p\}$ ja $s = \min\{v, q\}$, täytyy sen samalla olla jaollinen äärettömän monella muotoa 2^r3^s olevalla luvulla. Näin ollen $mF - C = 0$, joten

$$2^p3^q = F2^u3^v.$$

Jokainen L_m:n alkio on johdettu joukon A alkiosta soveltamalla toistuvasti ko. alkioon sellaisia sääntöjä, joita voidaan soveltaa L_m:n lauseisiin. Joukko A on selvästikin äärellinen, mutta jos äärellisen joukon muotoa 2^u3^v olevia lukuja kerrotaan toistuvasti vakiolla F, eivät kaikki tulokset voi olla muotoa 2^p3^q, $p, g \geq 0$, olevia lukuja.

Näin on päädytty ristiriitaan vastaoletuksen kanssa, joten L ei ole IL-kieli. □

Apulause 4.16.

$L_{T0L} \not\subset L_{E0L.}$

Todistus. [16]

Väite voidaan ilmaista muodossa "on olemassa sellainen T0L-kieli, joka ei ole E0L-kieli". Väitteen todistamiseksi tarkastellaan kieltä $K = \{x \in \{a, b\}^* \mid |x|_a = 2^i, i \geq 0\}$. Tämä kieli kuuluu perheeseen $L_{CS} \setminus L_{E0L}$ ([16, s. 104]). Siis kieli K ei ole E0L-kieli.

K on kuitenkin T0L-kieli, sillä K = L(G), kun G on T0L-systeemi $\langle\{a, b\}, a, P, \{P_1, P_2\}\rangle$, missä $P_1 = \{a \rightarrow a^2, b \rightarrow b\}$ja $P_2 = \{a \rightarrow ba, a \rightarrow ab, a \rightarrow a, b \rightarrow b\}$. □

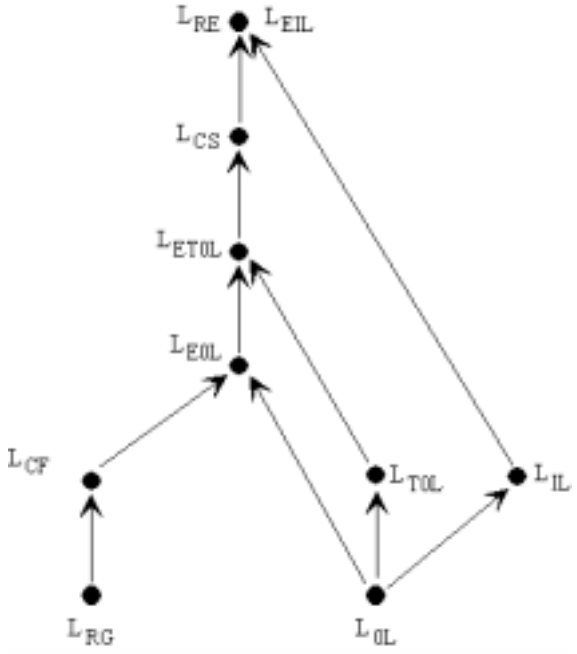
Apulause 4.17.

$L_{IL} \not\subset L_{CS}$.

Todistus. [16]

Nyt väite voidaan ilmaista muodossa "on olemassa sellainen IL-kieli, joka ei ole kontekstinen". Tämän työn lauseen 2.1 mukaan on olemassa sellainen lueteltavissa oleva kieli L, joka ei ole kontekstinen. Apulauseen 4.11 perusteella $L \in L_{EIL}$. Siis on olemassa sellainen IL-kieli K ja aakkosto V, että $K \cap V^* = L$. Jos K olisi kontekstinen kieli, niin silloin myöskin L olisi kontekstinen kieli, koska V* on kontekstinen kieli ja L_{CS} on suljettu leikkauksen suhteen [18, teoreema 9.6]. Siis $K \in L_{IL} \setminus L_{CS}$. □

Kuvan 11 oikeellisuus seuraa nyt apulauseista 4.1 - 4.17.



Kuva 11. L-hierarkia suhteessa Chomskyn kielihierarkiaan.

5. L-SYSTEEMIEN PÄÄTÖSONGELMISTA

Tässä luvussa käsitellään L-systeemeihin liittyviä päätösongelmia. Päätösongelman ratkeavuutta ja ratkeamattomuutta havainnollistetaan osoittamalla tietty päätösongelma ratkeavaksi tietyssä kieliperheessä (D0L-systeemit, kohta 5.1) ja osoittamalla saman päätösongelman muuttuvan ratkeamattomaksi kun kieliperhettä laajennetaan (0L-systeemit, kohta 5.2).

Kuten lähteessä [25] todetaan, kiinnostavia päätösongelmia ovat vain sellaiset ongelmat, joissa sekä karekteristinen joukko että sen komplementti ovat äärettömiä. Näin ollen tässäkin työssä esiteltävillä päätösongelmilla on ääretön määrä tapauksia, joihin siis jokaiseen voidaan vastata joko "kyllä" tai "ei".

Ennen kuin siirrytään käsittelemään esimerkkejä L-systeemien päätösongelmista lienee paikallaan esitellä joitakin formaalien kielten tavallisimpia päätösongelmatyyppejä. Olkoon M jokin tietty tapa määritellä kieli. (M voi tarkoittaa siis esimerkiksi kaikkia tietyntyyppisiä kielioppeja, tietyntyyppisiä automaatteja, tai vaikka D0L-systeemejä). Tässä esimerkissä M:llä tarkoitetaan vaikkapa kohdassa 2.1 määriteltyjä kontekstittomia kielioppeja. Tällöin kielen ekvivalenssiongelmalla (*language equivalence problem*) tarkoitetaan päätösongelmaa, jonka tehtävänä on päättää, määrittelevätkö M:ään kuuluvat kieliopit m_1 ja m_2 saman kielen. Kielen sisältymisongelmalla (*language inclusion problem*) tarkoitetaan päätösongelmaa, jolla päätetään, sisältyykö kieliopin m_1 tuottama kieli kieliopin m_2 tuottamaan kieleen. Kielen tyhjyysongelman (*language emptiness problem*) ratkaisu määrittää, onko M:ään kuuluvan kieliopin m tuottama kieli tyhjä. Vastaavasti kielen äärellisyysongelman (*finitiness problem*) ratkaisu määrittää, onko tiettyyn kieliperheeseen kuuluva kieli L äärellinen vai ei. Lopuksi kielen kuulumisongelman (*membership problem*) ratkaisu kertoo, kuuluuko mielivaltainen lause w kieleen m (joka siis kuuluu M:ään). Tässä työssä annetaan seuraavaksi L-systeemeihin liittyen esimerkit kielen ekvivalenssiongelmasta.

Kannattaa huomata, että jos jokin päätösongelma on ratkeamaton jossakin kieliperheessä, niin se on ratkeamaton myös kaikissa ko. kieliperheen sisältävissä muissa kieliperheissä. Vastaavasti jos jokin päätösongelma on ratkeava jossain kieliperheessä, niin se on ratkeava myös kaikissa ko. kieliperheeseen sisältyvissä kieliperheissä. Tästä syystä ratkeamattomien päätösongelmien tapauksessa etsitään mahdollisimman pientä kieliperhettä, ja ratkeavien päätösongelmien tapauksessa vastaavasti mahdollisimman suurta kieliperhettä. [39, 40]

Aiemmin esitettyihin Chomskyn kielioppeihin liittyen päätösongelmista voi hieman yksinkertaistaen todeta, että säännöllisiin kieliin liittyvät päätösongelmat ovat ratkeavia ja lueteltavissa oleviin kieliin liittyvät päätösongelmat ovat ratkeamattomia. Monet kontekstittomia kieliä koskevat päätösongelmat ovat ratkeavia ja valtaosa kontekstisia kieliä koskevista päätösongelmista ovat ratkeamattomia. E0L-systeemit muistuttavat varsin paljon kontekstittomia kieliä päätösongelmien ratkeavuuden suhteen. [39]

5.1 D0L-SYSTEEMIEN EKVIVALENSSIONGELMA

Vaikka D0L-systeemien kieli- ja sekvenssiekvivalenssiongelmat eroavat jonkin verran toisistaan, ovat ne silti varsin läheistä sukua keskenään. Ero johtuu siitä, että kaksi D0L-systeemiä voivat tuottaa saman kielen eri järjestyksessä. Seuraavan esimerkin D0L-systeemit tuottavat saman kielen [40]:

$$G_1 = \langle \{a, b\}, b, \{a \rightarrow b^2, b \rightarrow a\} \rangle \text{ ja } G_2 = \langle \{a, b\}, a, \{a \rightarrow b, b \rightarrow a^2\} \rangle .$$

Kun G_1:n aksioomaan b aletaan soveltaa systeemin sääntöjä, saadaan lausesekvenssi $b \Rightarrow a \Rightarrow bb \Rightarrow aa \Rightarrow bbbb \Rightarrow aaaa \ldots$ ja kun G_2:lle tehdään sama, niin saadaan $a \Rightarrow b \Rightarrow aa \Rightarrow bb \Rightarrow aaaa \Rightarrow bbbb \ldots$. D0L-systeemit G_1 ja G_2 tuottavat siis saman kielen eri järjestyksessä, eli ne ovat kieliekvivalentteja, mutta eivät sekvenssiekvivalentteja. D0L-systeemien kieli- ja sekvenssiekvivalenssiongelmien sukulaisuus on helppo perustella sillä, että jo vuosia ennen kuin kumpaakaan ongelmaa pystyttiin todistamaan ratkeavaksi tiedettiin, että toisen ongelman todistaminen ratkeavaksi johtaisi heti myös toisen ongelman ratkeavuuden todistamiseen [40].

Ennen D0L-systeemejä koskevan sekvenssiekvivalenssiongelman ratkeavuuden todistamista esitetään kaksi esimerkkiä, jotka tuovat esiin aiheeseen liittyviä ongelmia [40].

Esimerkki 5.1.
Tarkastellaan aluksi kahta D0L-systeemiä:

$$G_1 = \langle \{a, b\}, aba, \{a \rightarrow aba, b \rightarrow \Lambda\} \rangle,$$
$$G_2 = \langle \{a, b\}, aba, \{a \rightarrow a, b \rightarrow ba^2b\} \rangle.$$

Esimerkin D0L-systeemeistä nähdään helposti, että $L(G_1) = L(G_2) = \{(aba)^{2n} \mid n \geq 0\}$ ja että sekvenssit $S(G_1)$ ja $S(G_2)$ ovat samat. Esimerkin kieliopit tuottavat siis samat kielet, ja vieläpä siten, että kieleen kuuluvat lauseet tuotetaan samassa järjestyksessä. Näin ollen G_1 ja G_2 ovat sekä kieli- että sekvenssiekvivalentteja.

Ekvivalenssiongelman yleisen tapauksen kanssa edellisellä esimerkillä ei kuitenkaan ole paljonkaan tekemistä. Jos puhutaan sekvenssiekvivalenssiongelmasta, niin eräs keino ongelman ratkaisemiseksi on muodostaa yksi kerrallaan sekvenssien $S(G_1)$ ja $S(G_2)$ lauseita, ja testata kullakin iteraatiokierroksella täsmäävätkö lauseet. Tämä proseduuri sisältää itse asiassa puolet sekvenssiekvivalenssiongelman ratkaisualgoritmista: jos G_1 ja G_2 *eivät* ole sekvenssiekvivalentteja, niin proseduuri pysähtyy ja antaa oikean vastauksen ("ei"). Ongelmana on kuitenkin se, että mikäli G_1 ja G_2 ovat sekvenssiekvivalentteja, niin proseduurin suoritus ei pääty koskaan.

Em. epätäydellisen algoritmin muuttamiseksi täydelliseksi pitäisi pystyä määrittämään sellainen luku $C(G_1, G_2)$, että jos $C(G_1, G_2)$ ensimmäistä sekvenssien $S(G_1)$ ja $S(G_2)$ termiä ovat samat, niin $S(G_1) = S(G_2)$. Tämän luvun määrittäminen on osoittautunut varsin hankalaksi tehtäväksi. Lähteen [40] mukaan näyttää kuitenkin siltä, että G_1:n (tai G_2:n) aakkoston merkkien lukumäärä kahdella kerrottuna olisi riittävä luku takaamaan sekvenssiekvivalenttiuden. Ainakaan vastaesimerkkiä ei ole keksitty. Tämä tulos on pystytty todistamaan korkeintaan kahdesta merkistä koostuville aakkostoille [19], mutta tätä tulosta ei ole pystytty yleistämään koskemaan kaikkia aakkostoja [4]. Esimerkki 5.2 on hankalin tunnettu esimerkki kahdesta sellaisesta D0L-systeemistä G_1 ja G_2, joille pätee $S(G_1) \neq S(G_2)$, mutta mahdollisimman monta (suhteessa aakkoston merkkien määrään) lausesekvenssin alkupään lausetta ovat samat [40].

Esimerkki 5.2.
Tarkastellaan D0L-systeemejä

$G_1 = \langle \{a, b\}, ab, \{a \rightarrow abb, b \rightarrow aabba\} \rangle$,
$G_2 = \langle \{a, b\}, ab, \{a \rightarrow abbaabb, b \rightarrow a\} \rangle$.

Kun yo. D0L-systeemien aksioomiin aletaan soveltaa ko. systeemin sääntöjä, saadaan seuraavat lausesekvenssit:

$S(G_1)$: ab $\Rightarrow$ abbaabba $\Rightarrow$ abbaabbaaabbaabbabbaabbaaabbaabbabbabbaabbaaabbaabbabbaabbaaabbaabbaabbaaabbaa
bbabbaabbaaabbaabbabbabbaabbaaabbaabbabbaabbaaabba $\Rightarrow$ …

$S(G_2)$: ab $\Rightarrow$ abbaabba $\Rightarrow$ abbaabbaaabbaabbabbaabbaaabbaabb $\Rightarrow$
abbaabbaaabbaabbabbaabbaaabbaabbabbaabbabbaabbaaabbaabbabbaabbaaabbaabbaaabbaa
bbabbaabbaaabbaabbabbaabbabbaabbaaabbaabbabbaabbaa $\Rightarrow$ …

Sekvensseistä nähdään, että niiden kolme ensimmäistä lausetta ovat samat, mutta sen jälkeen muodostettavat lauseet ovat keskenään erilaisia.

Näiden esimerkkien jälkeen siirrytään D0L-systeemien yleisen sekvenssiekvivalenssi-ongelman ratkaisun pariin. Määritellään aluksi ns. vastinjoukko (*equality set*) ja elementaarinen homomorfismi [40]:

Määritelmä 5.1.
Kahden homomorfismin g ja h, g, h: $V^* \rightarrow V_1^*$, *vastinjoukko* on joukko
$E(g, h) = \{w \in V^* \mid g(w) = h(w)\}$.

Huomaa, että mahdollisesti $V^* = V_1^*$.

Olkoot g ja h määritelty joukossa {a,b}* esimerkiksi seuraavasti:

$g(a) = h(b) = a$, $g(b) = h(a) = aa$.

Nyt vastinjoukko E(g, h) koostuu kaikista sellaisista lauseista, joissa on yhtä monta a ja b kirjainta.

Kahden homomorfismin g ja h (jotka on määritelty V*:ssä) sanotaan olevan samat kielessä $L \subseteq V^*$, jos ehto g(w) = h(w) pätee kaikille L:n lauseille w. Selvästi g ja h ovat samat kielessä L jos ja vain jos $L \subseteq E(g, h)$.

Määritelmä 5.2.
Homomorfismi h: $V^* \rightarrow V_1^*$ (V ja V_1 ovat siis aakkostoja, lisäksi on mahdollista että $V = V_1$) on ositettavissa (*simplifiable*), jos on olemassa kardinaalisuudeltaan V:tä pienempi aakkosto V_2 sekä sellaiset homomorfismit f: $V^* \rightarrow V_2^*$ ja g: $V_2^* \rightarrow V_1^*$, että h = gf. Muussa tapauksessa h on *elementaarinen* homomorfismi.

Seuraavaa kolmea apulausetta (5.1 – 5.3) tarvitaan tämän kohdan ensimmäisen lauseen todistuksessa. Kaikki apulauseet ovat lähteestä [40].

Apulause 5.1.
Jos homomorfismit g ja h ovat elementaarisia, niin E(g, h) on säännöllinen.

Todistus.
Sivuutetaan. ([40, s. 74]).

Apulause 5.2.
On olemassa algoritmi, joka päättää, ovatko kaksi homomorfismia g ja h samat säännöllisessä kielessä R.

Todistus.
Sivuutetaan. ([40, s. 83]).

Apulause 5.3.
D0L-systeemin tuottaman kielen L sisältyminen säännölliseen kieleen R on ratkeava päätösongelma.

Todistus.
Sivuutetaan. ([40, s. 84]).

Lause 5.1.
Kahden elementaarisen homomorfismin samuus annetussa D0L-kielessä on ratkeava päätösongelma.

Todistus. [39, 40]
Tehtävänä on siis tutkia, ovatko elementaariset homomorfismit g_1 ja g_2 samat kielessä L(G), missä G = ⟨V, w, h⟩ on D0L-systeemi. Apulauseen 5.1 mukaan $E(g_1, g_2)$ on säännöllinen. (Huom. Äärellisten automaattien hyväksymä kielten perhe on sama kuin säännöllisten kielten perhe [25]). Seuraavaksi kaikki säännölliset kielet (aakkostossa V) luetellaan seuraavasti: R_0, R_1, R_2, … . (Tämä toimenpide on tehtävissä deterministisellä algoritmilla.)

Muodostetaan seuraavaksi algoritmi kahdesta proseduurista: ensimmäisen avulla todetaan homomorfismien samuus, toisen avulla todetaan jos ne eivät ole samat. Jälkimmäisen proseduurin konstruointitapa on ilmeinen: tarkastellaan sekvenssiä

$$w = h^0(w), h^1(w), h^2(w), \ldots,$$

ja verrataan, pitääkö paikkaansa, että

$$g_1(h^i(w)) = g_2(h^i(w)).$$

Jos morfismit g_1 ja g_2 eivät ole samat L(G):ssä, niin proseduuri pysähtyy ja antaa oikean vastauksen ("ei").

Ensimmäisen proseduurin (jonka avulla siis todetaan morfismien samuus) i+1:s askel rakentuu seuraavasti. Ensin tarkistetaan apulauseen 5.2 avulla, ovatko morfismit g_1 ja g_2 samat R_i:ssä. Jos ne ovat, niin seuraavaksi tarkistetaan apulauseen 5.3 avulla, sisältyykö L(G) R_i:hin. Jos L(G) sisältyy R_i:hin, niin ko. morfismit ovat samat L(G):ssä. Mikäli ko. morfismit eivät ole samat R_i:ssä tai L(G) ei sisälly R_i:hin, niin algoritmin suorituksessa siirrytään askeleeseen i+2. Jos morfismit g_1 ja g_2 ovat samat L(G):ssä, niin proseduuri pysähtyy ja antaa oikean vastauksen ("kyllä").

Lauseen 5.1 todistava algoritmi saadaan nyt aikaiseksi suorittamalla molemmat proseduurit samanaikaisesti, siis suorittamalla vuorotellen yksi askel kummastakin proseduurista. Toinen proseduuri pysähtyy ja antaa oikean vastauksen, jolloin myös koko algoritmi pysähtyy ja antaa oikean vastauksen. □

D0L-systeemien sekvenssiekvivalenssiongelma on redusoitavissa lauseeseen 5.1. Perusajatuksena tässä redusoinnissa on muokata mielivaltaisen D0L-systeemin homomorfismi sellaiseen muotoon, että riittää testata elementaaristen homomorfismien samuus ko. D0L-kielessä. Seuraavaa apulausetta käytetään hyväksi ko. redusoinnissa.

Apulause 5.4.
Oletetaan, että homomorfismit h_1 ja h_2 on määritelty $V^* \rightarrow V^*$. Tällöin on olemassa sekvenssi $i_1, \ldots, i_m$, jonka elementit ovat joukosta {1, 2}, sekä sellaiset homomorfismit f, p_1 ja p_2, että

$$h_j h_{i_1} \ldots h_{i_m} = p_j f, \qquad j = 1, 2,$$

ja homomorfismit p_j ovat elementaarisia. Lisäksi sekvenssi $i_1, \ldots, i_m$ ja homomorfismit f, p_1 ja p_2 ovat algoritmisesti muodostettavissa.

Todistus.
Sivuutetaan. ([40, s. 86]).

Lause 5.2.
D0L-systeemien sekvenssienkvivalenssiongelma on ratkeava.

Todistus. [40, 39]
Yleisyyttä loukkaamatta voidaan olettaa, että käsiteltävät D0L-systeemit ovat seuraavat:

$$G_i = \langle V, w, h_i \rangle, \; i = 1, 2,$$

ts. niiden aksioomat ja aakkostot ovat samat. Merkitään lausesekvenssien $S(G_i)$, $i = 1, 2$, lauseita seuraavasti:

$$w = w_0^{(i)}, w_1^{(i)}, w_2^{(i)}, \ldots .$$

Olkoon sekvenssi $i_1, \ldots, i_m$ ja homomorfismit f, p_1 ja p_2 sellaiset, että ne täyttävät apulauseen 5.4 ehdot homomorfismeille h_1 ja h_2. Merkitään homomorfismeja

$$g_i = h_i h_{i_1} \ldots h_{i_m} = p_i f, \;\; i = 1, 2,$$

ja D0L-systeemejä

$$G_{ij} = \langle V, w_j^{(i)}, g_i \rangle, \; 1 \leq i \leq 2, \; 0 \leq j \leq m.$$

(Kannattaa huomata, että V saattaa sisältää sellaisia merkkejä, jotka eivät esiinny $S(G_{ij})$:ssä. Morfismit g_i ovat kuitenkin määriteltyjä koko aakkostossa V.)

Nähdään helposti, että jos $S(G_1) = S(G_2)$, niin myös

$$S(G_{1j}) = S(G_{2j}), \text{ kun } 0 \leq j \leq m. \qquad \text{(i)}$$

Yo. implikaatio pätee myös toiseen suuntaan. Tämä nähdään epäsuorasti seuraavalla tavalla:

Oletetaan, että (i) pätee, mutta oletetaan myös, että on olemassa sellainen kokonaisluku n, että $w_n^{(1)} \neq w_n^{(2)}$. Valitaan n siten, että se on pienin kokonaisluku jolla edellisessä lauseessa esitetty yhtälö pätee. Tilanne $n \leq m$ ei selvästikään ole mahdollinen, sillä siinä tapauksessa sekvenssien $S(G_{1n})$ ja $S(G_{2n})$ ensimmäiset elementit olisivat toisistaan poikkeavat, mikä on

ristiriidassa oletuksen kanssa. Joten on oltava $n \geq m+1$. Jos valitaan uusi sellainen kokonaisluku t, että

$$0 \leq n_1 = n - t(m + 1) \leq m,$$

niin (t+1):s lause sekvensseissä $S(G_{1n_1})$ ja $S(G_{2n_1})$ eroavat toisistaan, mikä on myöskin ristiriidassa oletuksen kanssa. Näin ollen lauseen 5.2 todistamiseksi riittää todistaa yhtälön (1) sisältämä päätösongelma ratkeavaksi. Tätä tarkoitusta varten määrätään indeksi j kiinteäksi, ja merkitään

$$H_i = G_{ij} = \langle V, w_j^{(i)}, g_i \rangle, \; i = 1, 2.$$

Nyt voidaan tehdä oletus $w_j^{(1)} = w_j^{(2)}$, mikä puolestaan tarkoittaa sitä, että $f(w_j^{(1)}) = f(w_j^{(2)})$, sillä muutenhan $S(H_1) \neq S(H_2)$. (Homomorfismi f, kuten myös seuraavaksi tarvittavat homomorfismit p_1 ja p_2, määriteltiin todistuksen alussa).

Tarkastellaan lopuksi D0L-systeemejä

$$K_i = \langle V_i, f(w_j^{(i)}), fp_i \rangle, \; i = 1, 2,$$

missä V_i on aakkoston V osajoukko, jossa g_i on ositettu. (Tässä yhteydessä täytyy käyttää V:n osajoukkoa, koska fp_i ei ole välttämättä määritelty koko V:ssä.)

Lauseen 5.1 perusteella voidaan ratkaista, ovatko p_1 ja p_2 samat $L(K_1)$:ssä. Tässä todistuksessa käytetyistä merkinnöistä seuraa kuitenkin suoraan, että p_1 ja p_2 ovat samat $L(K_1)$:ssä jos ja vain jos $S(H_1) = S(H_2)$. □

Kuten tämän kohdan alussa todettiin, niin ennen kuin kumpaakaan ongelmaa (siis D0L-systeemejä koskevaa kieli- ja sekvenssiekvivalenssiongelmaa) pystyttiin todistamaan ratkeavaksi tiedettiin, että toisen ongelman todistaminen ratkeavaksi johtaisi heti myös toisen ongelman ratkeavuuden todistamiseen. Niinpä ei ole vaikea arvata, että myös D0L-systeemejä koskeva kieliekvivalenssiongelma on ratkeava.

Lause 5.3.
D0L-systeemien kieliekvivalenssiongelma on ratkeava.
Todistus.
Sivuutetaan. [39, s.136]

5.2 0L-SYSTEEMIEN EKVIVALENSSIONGELMA

Kohdassa 5.1 todistettiin D0L-systeemien kieliekvivalenssiongelman olevan ratkeava päätösongelma. Tässä kohdassa todistetaan, että jos kieliperhettä laajennetaan D0L-systeemeistä 0L-systeemeihin, niin sama päätösongelma muuttuu ratkeamattomaksi.

Ns. Postin vastaavuusongelmaa (*Post correspondence problem*), tai paremminkin sen ratkeamattomuutta, käytetään varsin usein hyödyksi formaalien kielten teoriassa, kun pyritään todistamaan, ettei jotakin tiettyä algoritmia ole olemassa. Tavallinen tapa tietyn ongelman P ratkeamattomuuden todistamiseen on redusoida P ongelmaksi P_1, joka tiedetään ratkeamattomaksi, ja osoittamalla sitten, että jos P olisi ratkeava, niin myös P_1 olisi ratkeava. Lähteessä [40] Postin vastaavuusongelma on esitetty vastinjoukon määritelmää hyväksikäyttäen seuraavasti: vastinjoukon (*equality set*, tässä tilanteessa voidaan käyttää ehkä mieluummin termiä vastinkieli, *equality language*) E(g, h) tyhjyysongelma on ratkeamaton päätösongelma. (Homomorfismit g ja h on määritelty kuten vastinjoukon määritelmässä 5.1).

Postin vastaavuusongelma esitetään kuitenkin "aukikirjoitettuna" apulauseessa 5.5 seuraavasti:

Apulause 5.5. [25, 16]
Olkoon V vähintään kahdesta merkistä koostuva aakkosto. Tällöin ei ole olemassa sellaista algoritmia, joka kykenisi päättelemään syötteeksi saamistaan ei-tyhjiä lauseita sisältävistä merkkijonolistoista $w_1, w_2, \ldots, w_n$ ja $x_1, x_2, \ldots, x_n$ ($n \geq 1$), että onko olemassa sellaista indeksijonoa $i_1, i_2, \ldots, i_m$ ($m \geq 1$), että $w_{i_1}w_{i_2}\ldots w_{i_m} = x_{i_1}x_{i_2}\ldots x_{i_m}$.

Todistus.
Sivuutetaan. [18, lause 14.1]

Postin vastaavuusongelman *tapaus* (*instance*) koostuu siis kahdesta merkkijonolistasta $w_1, w_2, \ldots, w_n$ ja $x_1, x_2, \ldots, x_n$ ($n \geq 1$). Tapauksella sanotaan olevan ratkaisu, jos ja vain jos on olemassa sellainen indeksijono $i_1, i_2, \ldots, i_m$ ($m \geq 1$), että $w_{i_1}w_{i_2}\ldots w_{i_m} = x_{i_1}x_{i_2}\ldots x_{i_m}$.

Apulauseen 5.5 avulla voidaan todistaa seuraava lause:

Lause 5.4.
0L-systeemien kieliekvivalenssiongelma on ratkeamaton päätösongelma.

Todistus. [16]
Väite voidaan muotoilla tätä todistusta varten seuraavasti: ei ole olemassa algoritmia, joka osaisi päätellä, päteekö kahdelle mielivaltaiselle 0L-systeemille G ja H, että L(G) = L(H). (Vaikka tässä todistuksessa käytetään Chomskyn kielioppien yhteydessä määriteltyjä merkintätapoja, niin todistuksen 0L-systeemin aakkostoa ei kuitenkaan ole jaettu apu- ja perusmerkkeihin. Kaikki ko. 0L-systeemillä tuotetut lauseet kuuluvat siis 0L-systeemin tuottamaan kieleen).

Olkoon V vähintään kahdesta merkistä koostuva aakkosto, ja olkoon $K = \alpha_1, \alpha_2, \ldots, \alpha_n$, $L = \beta_1, \beta_2, \ldots, \beta_n$ Postin vastaavuusongelman tapaus aakkostossa V. Olkoon $V_1 = \{S,A,B,C,D,E,F\}$ sellainen aakkosto, että $V \cap V_1 = \varnothing$. Olkoon $G_{K,L} = \langle V \cap V_1, S, P\rangle$ 0L-systeemi, jonka sääntöjoukko P koostuu seuraavista säännöistä (merkintä $\overline{x}$ tarkoittaa lausetta x käännetyssä järjestyksessä kirjoitettuna):

$S \rightarrow \alpha_i E \overline{\beta}_i$, $i \in \{1, \ldots, n\}$,
$S \rightarrow xAx$, $x \in V$,
$S \rightarrow xBy$, $x, y \in V$ ja $x \neq y$,
$A \rightarrow xAx$, $x \in V$,
$A \rightarrow xBy$, $x, y \in V$ ja $x \neq y$,
$A \rightarrow xC$, $x \in V$,
$A \rightarrow Fx$, $x \in V$,
$B \rightarrow xB$, $x \in V$,
$B \rightarrow Bx$, $x \in V$,
$B \rightarrow D$,
$C \rightarrow Cx$, $x \in V$,
$C \rightarrow D$,
$E \rightarrow \alpha_i E \overline{\beta}_i$, $i \in \{1, \ldots, n\}$,
$F \rightarrow Fx$, $x \in V$,
$F \rightarrow D$,
$x \rightarrow x$, $x \in V$.

Olkoon $H_{K,L} = \langle V \cup V_1, S, P_1\rangle$ 0L-systeemi, jonka sääntöjoukko $P_1 = P \cup \{E \rightarrow D\}$. Näiden määritelmien perusteella nähdään, että

$$
\begin{aligned}
L(G_{K,L}) = \{S\} \; & \cup \{wE\overline{w} \mid w \in V^+\} \cup \{wD\overline{u} \mid w, u \in V^+, w \neq u\} \\
& \cup \{wazBub\overline{w} \mid w \in V^*, u, z \in V^*, a, b \in V, a \neq b\} \\
& \cup \{wzC\overline{w} \mid w, z \in V^+\} \cup \{wFz\overline{w} \mid w, z \in V^+\} \\
& \cup \{wazBub\overline{w} \mid w \in V^*, u, z \in V^*, a, b \in V, a \neq b\} \\
& \cup \{wE\overline{u} \mid w = \alpha_{i_1}\alpha_{i_2}\ldots\alpha_{i_m}, u = \beta_{i_1}\beta_{i_2}\ldots\beta_{i_m}, \\
& \qquad m \geq 1, i_1, \ldots, i_m \in \{1, \ldots, n\}\}
\end{aligned}
$$

ja

$$
\begin{aligned}
L(H_{K,L}) = L(G_{K,L}) \cup \{wD\overline{u} \mid & w = \alpha_{i_1}\alpha_{i_2}\ldots\alpha_{i_m}, u = \beta_{i_1}\beta_{i_2}\ldots\beta_{i_m}, \\
& m \geq 1, i_1, \ldots, i_m \in \{1, \ldots, n\}\}.
\end{aligned}
$$

Näin ollen $L(G_{K,L}) = L(H_{K,L})$ jos ja vain jos Postin vastaavuusongelman tapauksella K,L on ratkaisu. Eli jos olisi olemassa sellainen algoritmi, joka osaisi päätellä, päteekö kahdelle mielivaltaiselle 0L-systeemille G ja H, että L(G) = L(H), niin silloin olisi olemassa myöskin sellainen algoritmi, joka osaisi päätellä, onko mielivaltaisella Postin vastaavuusongelman tapauksella ratkaisu vai ei. Näin ollen lause 5.4 seuraa apulauseesta 5.5. □

Pienenä L-systeemien päätösongelmiin liittyvänä yksityiskohtana todettakoon, että vaikka 0L-systeemien ekvivalenssiongelma onkin ratkeamaton päätösongelma, niin 0L- ja D0L-systeemien *välinen* ekvivalenssiongelma on ratkeava päätösongelma [4].

6. L-SYSTEEMEILLÄ TUOTETTUJEN MERKKIJONOJEN ERÄS TULKINTA

Merkkijonon $b_v a_o b_v a_o a_o b_v a_o a_o$ tulkinta kuvassa 6 (kohdassa 3.1) on melko alkeellinen. Mutkikkaampien organismien mallintamiseen tarvitaan kehittyneempiä tulkintatapoja L-systeemien avulla tuotettujen merkkijonojen esittämiseen graafisessa muodossa. Lähteissä [36, 38] kerrotaan, kuinka mm. Frijters ja Lindenmayer, Hogeweg ja Hesper, Szilard ja Quinton sekä Prusinkiewicz ovat omilla tutkimuksillaan myötävaikuttaneet tällaisen kehittyneemmän tulkintatavan (*turtle interpretation of strings*) syntyyn. Tässä työssä tästä tulkintatavasta käytetään nimitystä piirtopää-tulkinta.

Piirtopää-tulkintaa käyttäen on mahdollista tulkita myös kolmiulotteisia kuvioita, mutta tässä työssä tarkastellaan ainoastaan kaksiulotteisia kuvioita.

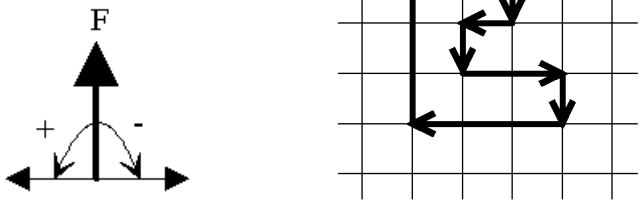
Piirtopää-tulkinnan idea on seuraava [36, 38]: piirtopään tila esitetään kolmikon (x, y, α) avulla. Karteesiset koordinaatit (x, y) esittävät piirtopään sijainnin ja kulma α ilmaisee piirtopään kulkusuunnan. Piirtopää liikkuu piirtoalustalla piirtopäälle annettujen komentojen perusteella. Komennot esitetään symbolien F, f, + ja – avulla. Näitä symboleja kutsutaan tässä työssä perussymboleiksi. Kun ensin määritellään askeleen pituus d ja kulkusuunnan muutoskulma δ, voidaan piirtopäälle annettaville komennoille antaa seuraavat määritelmät (ks. kuva 12):

F Piirtopää liikkuu d:n pituisen askeleen eteenpäin. Piirtopään uusi tila on (x', y', α), missä $x' = x + d \cos\alpha$ ja $y' = y + d \sin\alpha$. Pisteiden (x, y) ja (x', y') välille piirretään jana.

f Piirtopää liikkuu d:n pituisen askeleen eteenpäin piirtämättä janaa.

\+ Piirtopää kääntyy vasemmalle δ astetta. Piirtopään uusi tila on $(x, y, \alpha + \delta)$.

– Piirtopää kääntyy oikealle δ astetta. Piirtopään uusi tila on $(x, y, \alpha - \delta)$.



ν = FFF-FF-F-F+F+FF-F-FFF

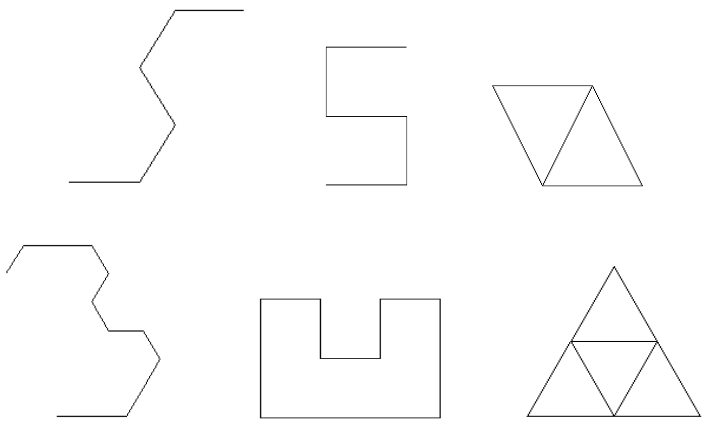
Kuva 12. (a) Symbolien F, + ja – tulkinta. (b) Merkkijonon tulkinta. Kulkusuunnan muutoskulma δ on 90°. Alkutilassaan piirtopää on syntyneen kuvion vasemmassa alalaidassa, ja sen kulkusuunta on suoraan ylöspäin.

Piirtopää pystyy tulkitsemaan merkkijonon ν, kun piirtopään alkutila (x_0, y_0, α_0) sekä parametrit d ja δ on määritelty. Piirtopää tulkitsee merkkijonon ν kuvan 12b esittämällä tavalla. Tässä kuvassa tosin piirtopää on havainnollisuuden vuoksi komennettu piirtämään nuolia, eikä janoja kuten ylempänä esitetyssä yleisessä määritelmässä kerrotaan. Haluttaessa piirtopää voi piirtää janojen tai nuolien sijaan jonkin muunkin muotoisen objektin tulkitessaan jotakin merkkijonoa, riippuen siitä, mitä L-systeemillä halutaan mallintaa.

Kuten jo aiemmin todettiin, parametrit d ja δ pitää määritellä ennen merkkijonon graafisen tulkinnan aloittamista. Erityisesti muutoskulmalla δ on hyvin merkittävä vaikutus syntyvän kuvion muotoon, kun taas askeleen pituus d vaikuttaa pelkästään syntyvän kuvion lopulliseen kokoon. Muutoskulman δ vaikutusta syntyvään kuvioon on demonstroitu kuvassa 13, jossa on esitetty kahden merkkijonon graafinen tulkinta erilaisilla muutoskulman arvoilla [28]:

1. merkkijono: F+F+F–F–F
2. merkkijono: FF+FF+F+F–F–F+F+FF+F.

Graafinen tulkinta em. merkkijonoille on esitetty kolmella eri muutoskulman arvolla (60°, 90° ja 120°) kuvassa 13 [28]. Kuvasta nähdään, että pelkästään muutoskulmaa muuttamalla saadaan aikaan varsin radikaaleja eroja syntyvään kuvioon.



Kuva 13. Kahden merkkijonon graafinen tulkinta: F+F+F–F–F (ylemmät kolme kuvaa) ja FF+FF+F+F–F–F+F+FF+F (alemmat kuvat). Vasemmanpuolisissa kuvissa δ = 60°, keskimmäisissä 90° ja oikeanpuolisissa kuvissa 120°.

Piirtopäälle voidaan antaa myöskin yhdistettyjä komentoja, kuten lähteessä [28] on esitetty. Piirtopää pystyy tulkitsemaan myös symboleja, jotka tulkittuna koostuvat erilaisista perussymbolisekvensseistä. Seuraavassa esimerkissä käytetään symboleja L, R, S ja Z. Näiden symbolien tulkinta on esitetty taulukossa 1:

Taulukko 1. Symbolien L, R, S ja Z tulkinta.

Symboli	**Tulkinta**
L	+F–F–F+
R	–F+F+F–
S	FF+F+FF–F–FF
Z	FF–F–FF+F+FF

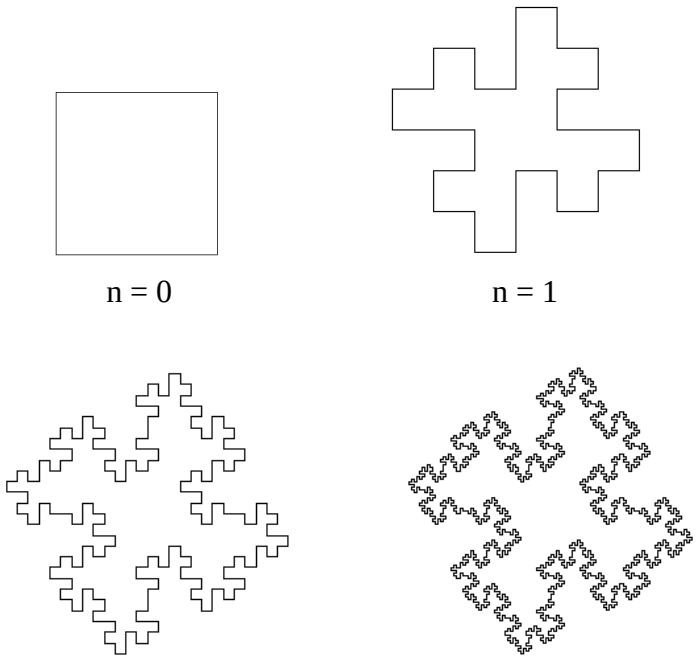
Piirtopää tulkitsee esimerkiksi merkkijonon FLRF–S samoin kuin se tulkitsisi merkkijonon F+F–F–F+–F+F+F–F–FF+F+FF–F–FF.

Piirtopää-tulkinta sopii menetelmänä hyvin L-systeemeillä tuotettujen merkkijonojen graafiseen tulkintaan. Esimerkiksi kuvassa 14 on esitetty 4 ensimmäistä iteraatiota L-systeemillä 2 tuotetusta Kochin käyrän variaatiosta (*quadratic Koch island*, [22, 38]). Kuvan 14 neljä käyrää on saatu tulkitsemalla graafisesti L-systeemin

$$\omega : F–F–F–F$$
$$p : F \rightarrow F–F+F+FF–F–F+F \qquad (9)$$

avulla tuotetut merkkijonot.

Muutoskulma δ on 90°. Askeleen pituutta d on pienennetty jokaisen iteraatiokierroksen välillä siten, että säännön oikean puolen muodostama lohko (F–F+F+FF–F–F+F) on yhtä pitkä kuin säännön vasemman puolen (F) muodostama lohko. Käyrän kehittyminen iteraatiokierrosten 0 ja 1 välillä on demonstroitu tarkemmin kuvassa 15. Siinä esitetään vaiheittain, kuinka aksioomasta F–F–F–F muodostetaan ensimmäisen iteraatiokierroksen jälkeen syntyvä merkkijono F–F+F+FF–F–F+F–F–F+F+FF–F–F+F–F–F+F+FF–F–F+F–F–F+F+FF–F–F+F, sekä ko. merkkijonojen graafinen tulkinta.



Kuva 14. L-systeemin (9) avulla tuotettujen merkkijonojen graafinen tulkinta iteraatiokierroksilla 0-3.

Kuvasta 15 nähdään, kuinka aksioomasta F–F–F–F kehittyy vaiheittain merkkijono F–F+F+FF–F–F+F–F–F+F+FF–F–F+F–F–F+F+FF–F–F+F–F–F+F+FF–F–F+F.

(F–F–F–F

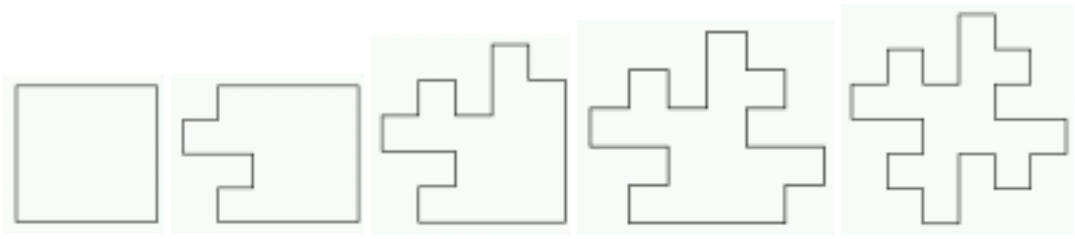
⇒ F–F+F+FF–F–F+F–F–F–F

⇒ F–F+F+FF–F–F+F–F–F+F+FF–F–F+F–F–F

⇒ F–F+F+FF–F–F+F–F–F+F+FF–F–F+F–F–F+F+FF–F–F+F–F

⇒ F–F+F+FF–F–F+F–F–F+F+FF–F–F+F–F–F+F+FF–F–F+F–F–F+F+FF–F–F+F).

Merkkijonon vaiheittainen kehittyminen on esitetty tässä havainnollisuuden vuoksi, itse asiassahan aksiooman kaikki F merkit korvataan L-systeemin (9) ainoan säännön oikealla puolella (F–F+F+FF–F–F+F–F–F–F) samanaikaisesti.



Kuva 15. Ensimmäinen iteraatio vaiheittain esitettynä.

Kuvan 15 esimerkki on piirretty fraktaaligeneraattorilla (tai tarkasti ottaen L-systeemeillä tuotettujen merkkijonojen tulkintaan suunnitellulla ohjelmalla), joka löytyy osoitteesta `http://www.ee.uwa.edu.au/~braunl/lgrammar/java/`. Sen avulla lukija voi halutessaan muodostaa kaikki tässä työssä esitettävät L-systeemien avulla muodostetut kuvat.

Kohdassa 2.3 esitetty kuvan 4 Kochin lumihiutale voidaan muodostaa samaan tapaan seuraavan PD0L-systeemin avulla:

$$\begin{aligned} &\omega : \text{F++F++F} \\ &p : \text{F} \rightarrow \text{F–F++F–F}. \end{aligned} \qquad (10)$$

Muutoskulman täytyy tietenkin olla tässä tapauksessa 60°. Lisäksi askeleen pituutta täytyy pienentää iteraatioiden välillä samaan tapaan kuin alla esitetyssä esimerkissä (parametroitu D0L-systeemi), eli siirryttäessä seuraavaan iteraatioon askeleen pituus pienenee aina kolmasosaan edellisen iteraation tilanteesta.

Sama Kochin lumihiutale voidaan muodostaa myöskin parametroidulla D0L-systeemillä. Parametroidun D0L-systeemin

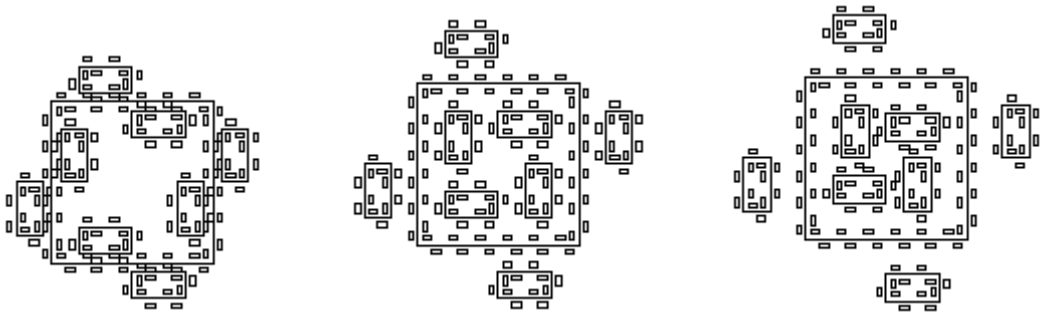
$$\begin{aligned} &\omega : \text{F(1)+(120)F(1)+(120)F(1)} \\ &p : \text{F(s)} \rightarrow \text{F(s/3)–(60)F(s/3)+(120)F(s/3)–(60)F(s/3)} \end{aligned} \qquad (11)$$

tuottamat merkkijonot tulkitaan samalla tavalla kuin edelläkin, mutta piirtopäälle annettavien komentojen syntaksi on tässä tapauksessa hieman erilainen verrattuna edellä esitettyihin komentoihin. Parametroidun L-systeemin tapauksessa komento F korvataan komennolla F(s) (piirtopää piirtää s:n mittaisen janan), + korvataan komennolla +(θ)

(piirtopää kääntyy vasemmalle θ astetta) ja − korvataan komennolla −(θ) (piirtopää kääntyy oikealle θ astetta). Nyt parametroitu D0L-systeemi (11) tuottaa myöskin Kochin lumihiutaleen (ks. kuva 4). [32, 33, 34]

Lähteissä [32, 33, 34] vastaavissa esimerkeissä aksiooma oli F(1)−(120)F(1)−(120)F(1) ja ainut sääntö F(s) → F(s/3)+(60)F(s/3)−(120)F(s/3)+(60)F(s/3). Merkkien "+" ja "−" tulkinta on siis vaihdettu tähän työhön keskenään. Tämä on tehty siksi, että lukijan olisi helpompi nähdä yhteneväisyydet L-systeemeissä (11) ja (10). Syntyvä kuvio on kuitenkin täsmälleen sama riippumattta siitä, käytetäänkö lähteiden [32, 33, 34] tapaa vai tässä työssä esitettyä tapaa.

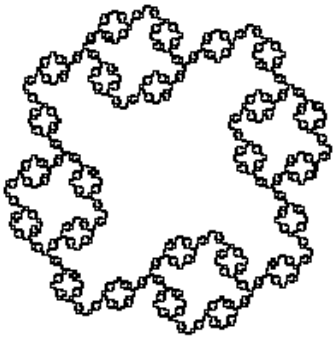
Varsin yksinkertaisten L-systeemien avulla voidaan siis muodostaa mutkikkaankin näköisiä Kochin käyriä. Tilanne vaikeutuu hieman, jos halutaan muodostaa käyrä, joka ei ole yhtenäinen. Tällöin käytettävään L-systeemiin täytyy lisätä toinen sääntö, joka sisältää perussymbolin f. Tätä toista sääntöä (ks. kuva 16) tarvitaan, jotta syntyvän kuvan eri komponentit pysyvät sopivalla etäisyydellä toisistaan. Kuvassa 16 demonstroidaan millaisia vaikutuksia L-systeemin toisen säännön muuttaminen aiheuttaa syntyvään kuvioon [38].



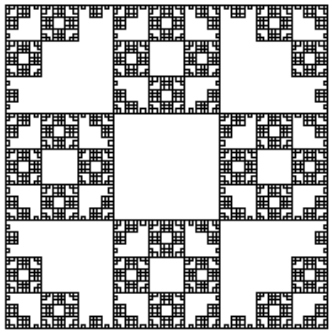
f → ff f → ffffff f → ffffffff

Kuva 16. Kaikissa kuvioissa n = 2, δ = 90°, ω = F+F+F+F ja F → F+f-FF+F+FF+Ff+FF-f+FF-F-FF-Ff-FFF.

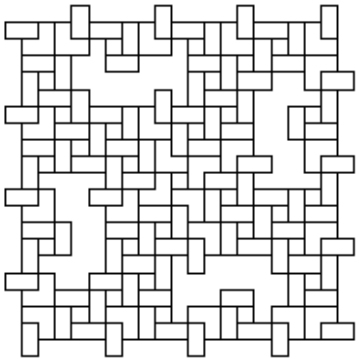
Kaikki tähän asti esitetyt esimerkit osoittavat, kuinka helppoa erilaisia L-systeemejä ja niissä esiintyviä sääntöjä on muutella, ja kuinka radikaaleja muutoksia näennäisen pienet muutokset L-systeemissä saattavat aiheuttaa syntyvään kuvioon. Juuri sääntöjen muokkaamisen helppouden takia L-systeemit soveltuvat hyvin erilaisten uusien Kochin käyrien muodostamiseen. Kuvassa 17 demonstroidaan, kuinka oleellisesti samasta aksioomasta muodostetut kuviot voivat erota toisistaan, kun L-systeemin ainoaan sääntöön tehdään pieniä muutoksia [38].



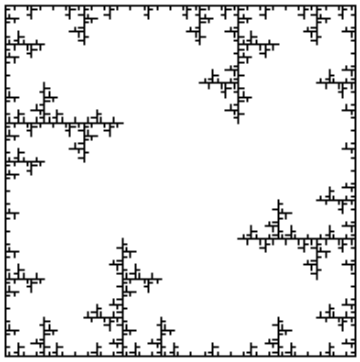
n = 4, δ = 90°
F-F-F-F
F → FF-F-F-F-F-F+F



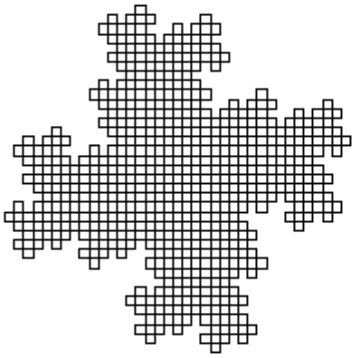
n = 4, δ = 90°
F-F-F-F
F → FF-F-F-F-FF



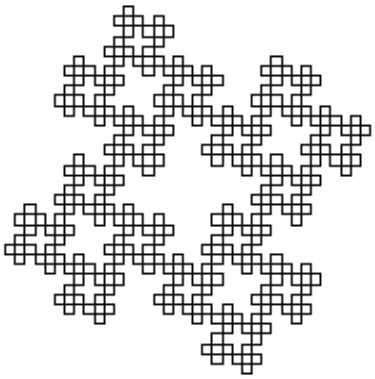
n = 3, δ = 90°
F-F-F-F
F → FF-F+F-F-FF



n = 4, δ = 90°
F-F-F-F
F → FF-F--F-F



n = 4, δ = 90°
F-F-F-F
F → F-FF--F-F



n = 4, δ = 90°
F-F-F-F
F → F-F+F-F-F

Kuva 17. Erilaisia Kochin käyriä.

Kuvia 13, 16 ja 17 tarkastelemalla lukija saa jonkinlaisen käsityksen L-systeemien ja niiden perusteella muodostettujen kuvioiden suhteesta, sekä siitä, kuinka sääntöjen muokkaaminen

tietyssä L-systeemissä vaikuttaa syntyvään kuvioon. Vaikka onkin mahdollista saada aikaan varsin mielenkiintoisia kuvioita tietyn L-systeemin sääntöjä muuntelemalla, niin ongelmana on usein kuitenkin se, miten jokin tietty olemassa oleva rakenne tai rakenteen kehittyminen voidaan mallintaa L-systeemin avulla. Tätä ongelmaa kutsutaan inferenssiongelmaksi (*inference problem*) L-systeemien teoriaa käsittelevissä lähteissä. Kirjallisuudessa on esitetty joitakin algoritmeja inferenssiongelman ratkaisemiseksi (ks. [38, s. 11]), mutta niiden avulla ei kuitenkaan pystytä mallintamaan esimerkiksi kasveja riittävän tarkasti.

7. HAARAUTUVIEN RAKENTEIDEN MALLINTAMINEN L-SYSTEEMIEN AVULLA

Tähän asti esitetyissä esimerkeissä L-systeemien avulla tuotetut merkkijonot on muunnettu graafiseen muotoon piirtopäätulkinnan mukaisesti. Syntyneet kuviot muodostuvat eripituisista janasekvensseistä. Janojen pituudet ja niiden väliset kulmat vaikuttavat syntyvän kuvion muotoon, jolloin kuviossa janat saattavat leikata toisiaan, jotkut janat saattavat tulla piirretyiksi moneen kertaan, ja toiset janat saattavat olla näkymättömiä (käytettäessä perussymbolia "f"). Periaatteessa syntyvät kuviot kuitenkin muodostuvat aina yhdestä ja ainoasta viivasta. Voidaan kuvitella, että ne on piirretty yhdellä vedolla kynää välillä paperista nostamatta, tosin musteensyöttö katkaistaan perussymboleja f tulkittaessa.

Haarautuvien rakenteiden sekä niitä tuottavien menetelmien mallintamiseen tarvitaan omat matemaattiset menetelmät. Rungollisen puun (*axial tree*) määritelmä on esitetty kohdassa 7.1, ja joitakin menetelmiä haarautuvien rakenteiden tuottamiseen on esitetty kohdissa 7.1 ja 7.2.

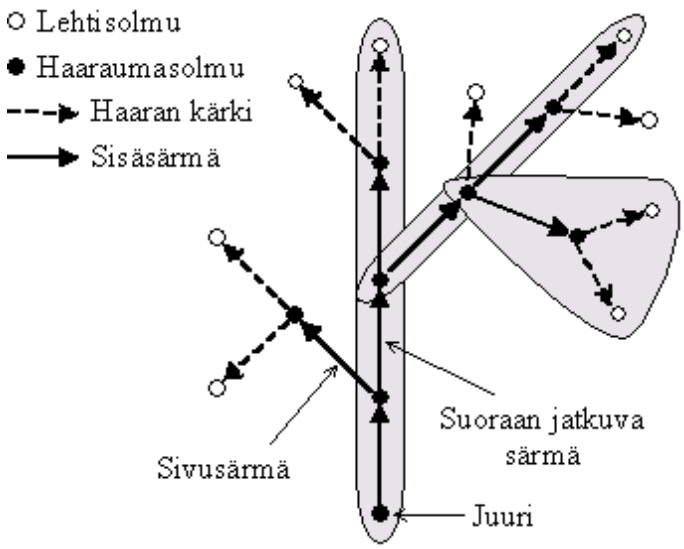
7.1 PUU 0L -SYSTEEMIT

Juurellinen puu muodostuu suunnatuista särmistä. (Graafiteoriassa juurellinen puu määritellään seuraavasti: Suunnattu graafi G on juurellinen puu, jos graafissa G on juuri ja vastaava suuntaamaton graafi on puu [42]). Juurellisella puulla on kolmenlaisia solmuja, nimittäin juuri, haaraumasolmuja (*branching point*, tutumpi termi tälle lienee sisäsolmu) ja lehtiä (*terminal node*). Juurellisen puun särmät muodostavat polkuja haaraumasolmuista (*base*) lehtiin (*terminal node*). Biologisessa kontekstissa tällaisia polkuja kutsutaan puun haaroiksi (*branch segment*). Lehteen päättyvää särmää kutsutaan haaran kärjeksi (*apex*). Kaikki muut särmät ovat sisäsärmiä (*internode*). [33]

Rungollinen puu on juurellisen puun erikoistapaus. Sen kaikilla solmuilla on korkeintaan yksi suoraan jatkuva särmä (*straight segment*, ks. kuva 18). Kaikki jäljelle jäävät särmät ovat ns. sivusärmiä (*lateral segment*). Särmäjonoa kutsutaan rungoksi (*axis*), jos on voimassa ehdot

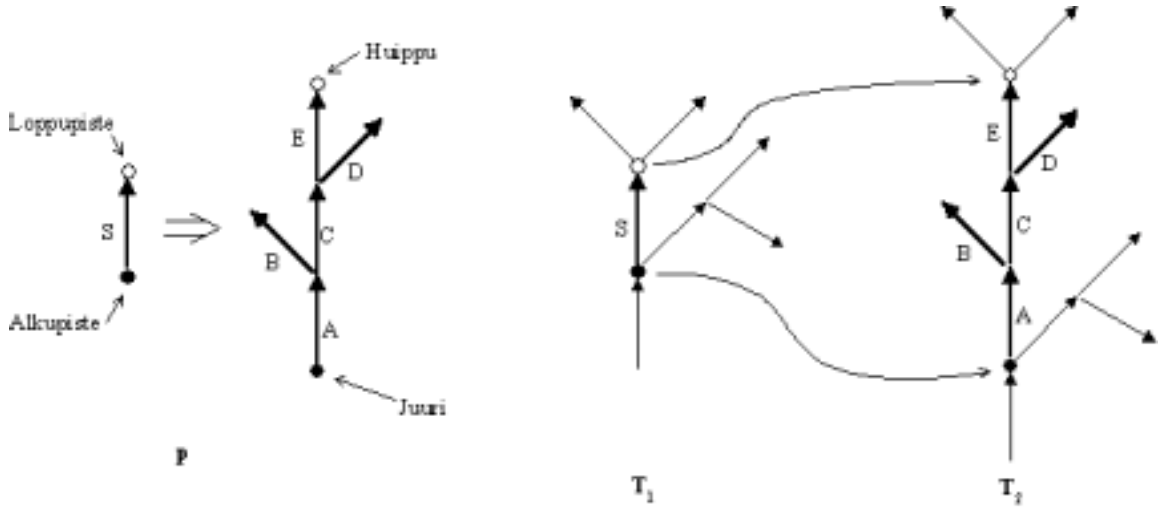
- särmäjonon ensimmäinen särmä alkaa puun juuresta, tai se on jonkin solmun sivusärmä,
- särmäjonon kaikki peräkkäiset särmät ovat suoraan jatkuvia särmiä, ja
- särmäjonon viimeistä särmää ei seuraa enää muita suoraan jatkuvia särmiä.

Puun haara (*branch*) muodostuu rungosta, ja kaikista siinä kiinni olevista särmäjonoista. Kukin puun haara on rungollinen (ali)puu. [38, 33]



Kuva 18. Rungollinen puu.

Haarautuvien rakenteiden mallintamisessa voidaan käyttää menetelmää, jossa L-systeemin säännöissä käsitellään suoraan rungollisia puita. Tässä tapauksessa säännössä korvataan sen vasemman puolen särmä säännön oikean puolen rungollisella puulla kuvan 19 osoittamalla tavalla [38].



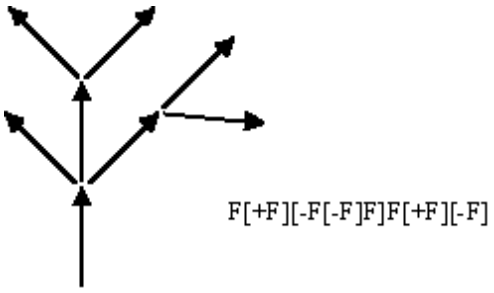
Kuva 19. Vasemmalla esitettyä sääntöä p sovelletaan oikeanpuoleisessa kuviossa puun T_1 särmään S.

Puu 0L -systeemi G koostuu kolmesta komponentista: särmäjoukosta V, alustetusta puusta ω (puun särmät kuuluvat joukkoon V) sekä sääntöjoukosta P (*tree productions*). L-systeemin G tuottama puu T_2 on johdettu suoraan puusta T_1 (merk. $T_1 \Rightarrow T_2$), jos T_2 on saatu T_1:stä korvaamalla samanaikaisesti T_1:n jokainen särmä P:hen kuuluvien sääntöjen oikeilla puolilla. Puu T on tuotettavissa G:n avulla n:n pituisella johdolla, jos on olemassa sellainen puusekvenssi $T_0, T_1, ..., T_n$, että $T_0 = \omega$, $T_n = T$ ja $T_0 \Rightarrow T_1 \Rightarrow ... \Rightarrow T_n$. [38]

Määritelmästä nähdään, että se ei sisällä rungollisten puiden määritelmää. Erityisesti rungollisten puiden mallintamisessa onkin käytännöllisempää käyttää pino 0L -systeemejä.

7.2 PINO L -SYSTEEMIT

Pino 0L -systeemien säännöt muistuttavat D0L-systeemien sääntöjä. Lisänä D0L-systeemeihin verrattuna on haarautuvien rakenteiden kuvaamiseen tarkoitettu elementti, jonka avulla on mahdollista mallintaa rungollisia puita. Pino 0L -systeemissä haaran alkaminen kuvataan "["-merkillä, ja haaran loppuminen kuvataan vastaavasti "]"-merkillä. Kun mallissa suljetaan haara "]"-merkillä, mallinnus jatkuu mallin rungosta (ks. kuva 20) [38].



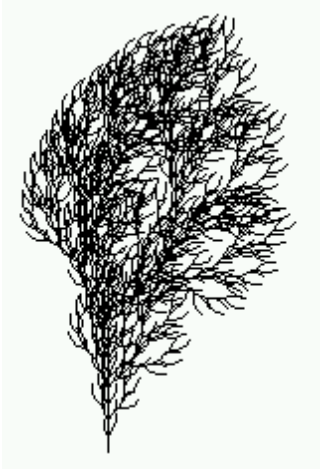
Kuva 20. Rungollinen puu esitettynä pino 0L –systeemin avulla.

Piirtopää tulkitsee merkit "[" ja "]" siten, että [-merkkiin törmätessään se tallentaa piirtopään tilan pinoon, ja jatkaa sitten merkkijonon tulkitsemista. Kun piirtopää kohtaa]-merkin, niin piirtopään tila palautetaan pinosta. Tässä vaiheessa tavallisesti piirtopään sijainti muuttuu, mutta janaa vanhan ja uuden sijaintipisteen välille ei piirretä.

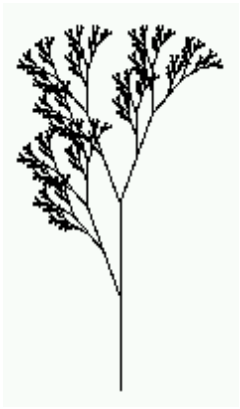
Merkkijonoja tuotetaan pino 0L -systeemeillä samaan tapaan kuin D0L-systeemeilläkin. Hakasulut korvaavat itsensä merkkijonoja tuotettaessa, eli säännöt [→[ja]→] oletetaan kuuluvaksi kaikkiin pino 0L -systeemeihin. Kuvassa 21 on esimerkkejä pino 0L -systeemeillä tuotetuista kuvista [38].

n = 5, δ = 25.7°
F
F→F[+F]F[-F]F

n = 5, δ = 20°
F
F→F[+F]F[-F][F]

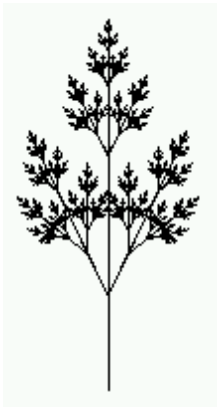


n = 4, δ = 22.5°
F
F→FF-[-F+F+F]+
[+F-F-F]



n = 7, δ = 20°
X
X→F[+X]F[-X]+X

F→FF



n = 7, δ = 25.7°
X
X→F[+X][-X]FX

F→FF

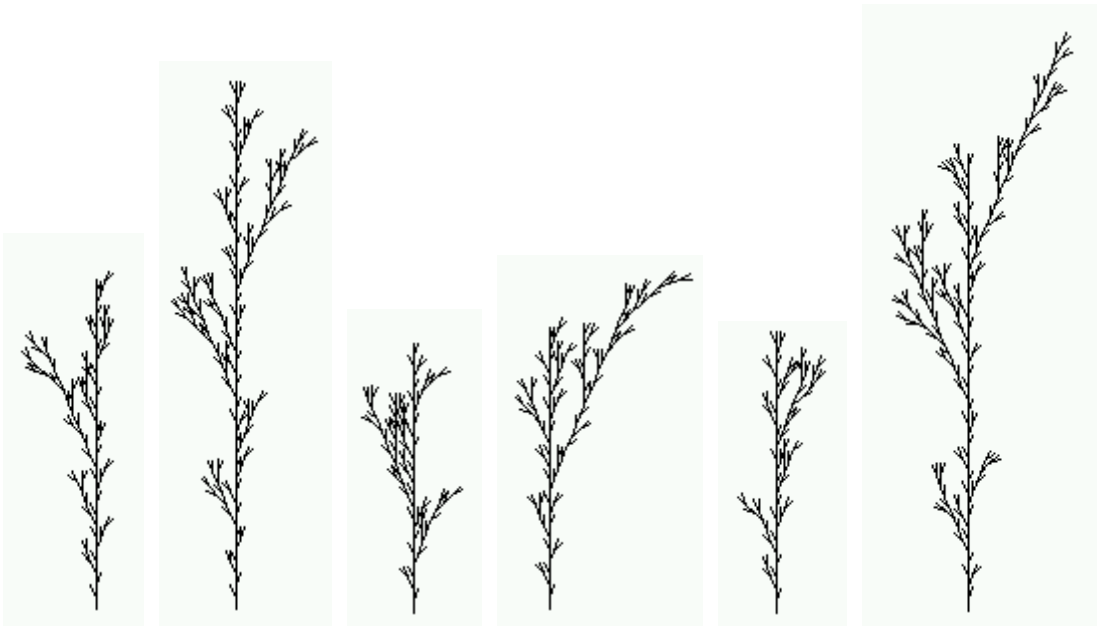
n = 5, δ = 22.5°
X
X→F-[[X]+X]+F
[+FX]-X

F→FF

Kuva 21. Pino 0L –systeemeillä tuotettuja kasveja.

Kuvan 21 kasvit generoituvat aina samanlaisiksi, kun ne muodostetaan kuvassa esitettyjen L-systeemien avulla. Kasvin eri yksilöiden välille ei siis saada aikaan minkäänlaista vaihtelua, vaan kaikki kasvin yksilöt ovat keskenään identtisiä. Jos tällä tavalla halutaan generoida esimerkiksi jokin fraktaalimaisema, niin maiseman kasvit muodostuvat yksitoikkoisen samankaltaisiksi.

L-systeemien avulla mallinnettavien kasvien (tai muiden objektien) eri yksilöiden välille saadaan eroavaisuuksia käyttämällä esim. kohdassa 3.4 esitettyä menetelmää, eli ottamalla käyttöön todennäköisyydet L-systeemin sääntöjen soveltamisessa. Kuvassa 22 on havainnollistettu kuinka erilaisia, mutta kuitenkin toisiaan muistuttavia kuvia voidaan muodostaa stokastisella pino 0L -systeemillä (12).



Kuva 22. Stokastisella L-systeemillä tuotettuja haarautuvia rakenteita.

Kuvan 22 kuviot on tuotettu stokastisen pino 0L –systeemin (12) avulla [38]:

$$
\begin{aligned}
&\omega : F \\
&p_1 : F \xrightarrow{.33} F[+F]F[-F]F \\
&p_2 : F \xrightarrow{.33} F[+F]F \qquad\qquad (12) \\
&p_3 : F \xrightarrow{.34} F[-F]F.
\end{aligned}
$$

Kuvan kasvit on tuotettu soveltamalla L-systeemin (12) aksioomaan ω 5 kertaa systeemin sääntöjoukkoa P ja tulkitsemalla näin syntynyt merkkijono graafisesti. Kuvan 22 kasvit muistuttavat toisiaan, ts. ne voisivat esittää jonkin tietyn kasvilajin eri yksilöitä.

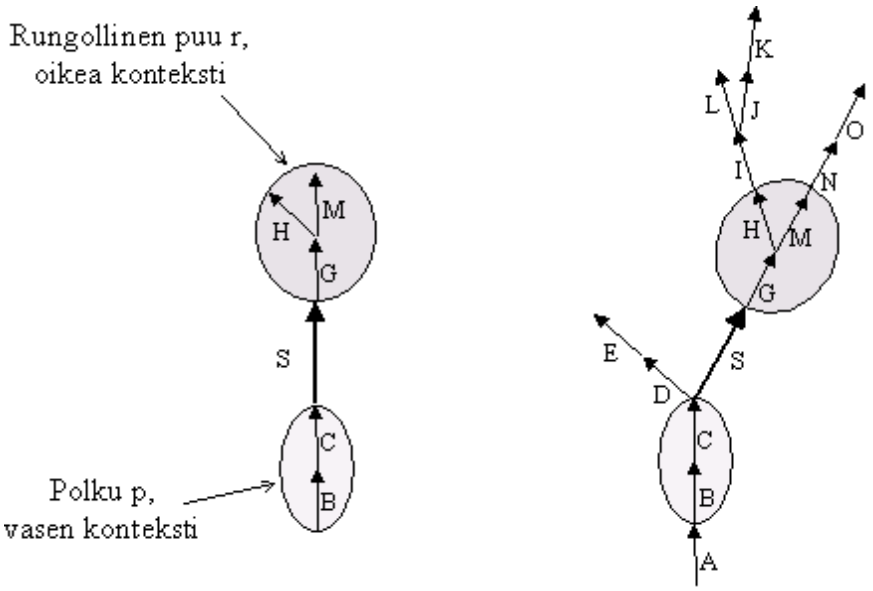
Jos kasvin kehittymistä ajatellaan prosessina, niin tämän prosessin kontrollimekanismina toimii kommunikointi kasvin eri osien välillä. Lindenmayerin mukaan kasvien sisäinen kommunikointi voi olla joko hierarkkista tai vuorovaikutteista (Lindenmayer käytti termejä

lineage ja *interaction*). Hierarkkisessa kommunikointitavassa informaatio kulkee kasvinosien välillä samaan tapaan kuin esimerkiksi hakupuissa eli vanhempisolmusta lapsisolmuihin. Vuorovaikutteisessa kommunikointitavassa informaatio kulkee puolestaan joko (topologisesti ajatellen) peräkkäisten moduulien välillä (*endogenous information exchange*) tai minkä tahansa kasvin moduulien välillä (*exogenous information exchange*). Esimerkkinä peräkkäisten moduulien välillä tapahtuvasta informaation kulusta voidaan mainita veden ja ravinteiden kulku kasvin juuresta aina sen latvaan asti. Vastaavasti esimerkki kasvin minkä tahansa moduulien välillä tapahtuvasta informaation vaihdosta on kasvin ylempien haarojen tai oksien toimiminen varjostimena, joka estää auringon valon pääsyn kasvin alempiin oksiin. [33]

Kontekstittomissa 0L-systeemeissä sääntöjä voidaan soveltaa riippumatta säännön vasemman puolen kontekstista. Tästä syystä 0L-systeemeillä voidaan mallintaa ainoastaan em. hierarkkiseen kommunikointitapaan perustuvia mekanismeja. Vuorovaikutteisen informaatiovaihdon mahdollistamiseksi L-systeemin sääntöjen soveltaminen täytyy tehdä riippuvaiseksi säännön vasemman puolen kontekstista, jolloin voidaan mallintaa myös peräkkäisten moduulien väliseen kommunikointiin (*endogenous*) perustuvia mekanismeja. Kontekstisia L-systeemejä hyödynnetään nimenomaan kasveja mallintavissa sovelluksissa. [33]

Kuvassa 23 on esitetty kontekstin tarkistus puu 2L -systeemissä. Puu 2L -systeemin säännön p vasen puoli (*predecessor*) koostuu kolmesta osasta:
1. polusta p, joka muodostaa säännön vasemman kontekstin,
2. särmästä S (*strict predecessor*), ja
3. rungollisesta puusta r, joka muodostaa säännön oikean kontekstin (ks. kuva 23). [38]

Suurin periaatteellinen ero säännön vasemman ja oikean kontekstin välillä on se, että puun juuresta on täsmälleen yksi polku annettuun särmään S, kun puolestaan särmästä S voi olla useita polkuja puun eri lehtisolmuihin. Säännön p osalta kontekstin tarkistus puussa T onnistuu, jos p on T:n sellainen polku, joka päättyy särmän S alkupisteeseen ja r on sellainen T:n alipuu, että se alkaa särmän S loppupisteestä. Kuvassa 23 on esimerkki siitä, kun kontekstin tarkistus onnistuu [38, 33].



Kuva 23. Onnistunut kontekstin tarkistus.

Edellä todettiin kontekstittomista L-systeemeistä, että rungollisia puita on helpompi mallintaa pino L -systeemien kuin puu L-systeemien avulla. Kontekstisten L-systeemien osalta tilanne on kuitenkin toisenlainen. Tämä johtuu siitä, että kontekstisten pino L -systeemien tapauksessa kontekstintarkistusproseduuri voi joutua hyppäämään puun haaraa tai sen osaa esittävien symbolien yli. Esimerkki tästä tilanteesta löytyy kuvasta 23 [38, 33]. Siinä kontekstisen pino L -systeemin sääntöä p, jonka vasen puoli on BC < S > G[H]M, voidaan soveltaa lauseeseen ABC[DE][SG[HI[JK]L]MNO], mutta kontekstintarkistusproseduuri joutuu hyppäämään symbolien [DE] yli etsiessään vasenta kontekstia ja symbolien I[JK]L yli etsiessään oikeanpuoleista kontekstia.

Kontekstisen pino L -systeemin säännön vasemman puolen vasenta kontekstia voidaan käyttää simuloimaan signaalin propagointia puun juuresta kohti puun huippua, ja säännön vasemman puolen oikeaa kontekstia voidaan vastaavasti käyttää simuloimaan signaalin propagointia huipusta kohti puun juurta. Esimerkiksi L-systeemi

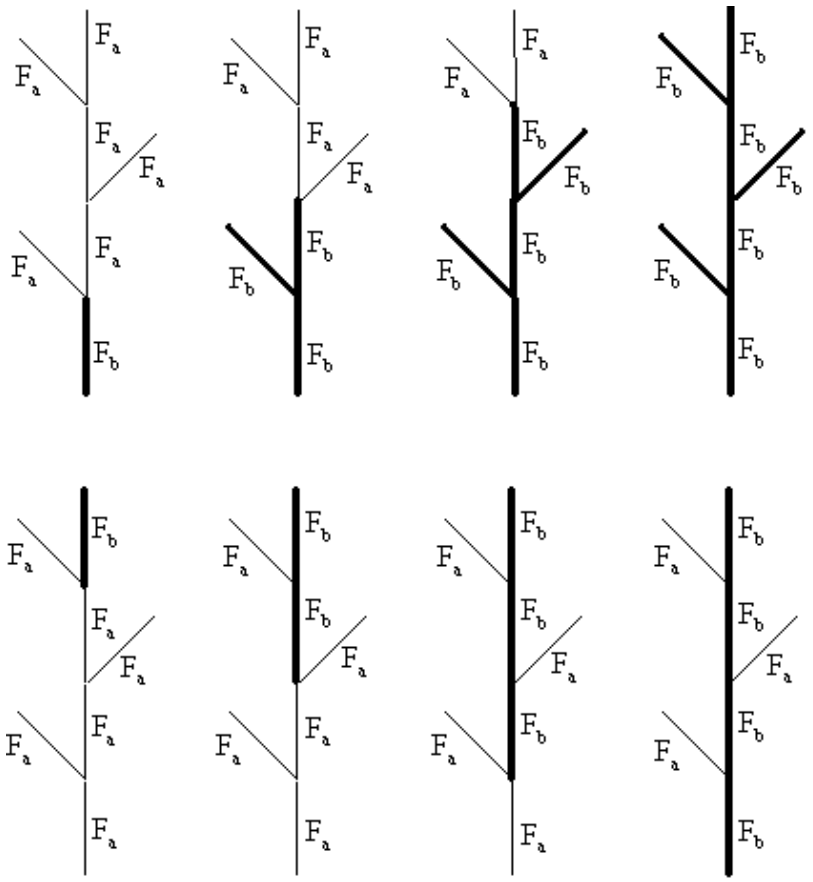
$$\begin{aligned} &\omega : F_b[+F_a]F_a[-F_a]F_a[+F_a]F_a \\ &p_1 : F_b < F_a \rightarrow F_b \end{aligned} \qquad (13)$$

simuloi signaalin propagointia puun juuresta puun huippuun sellaisessa haarautuvassa rakenteessa, joka ei enää kasva (ks. kuva 24, ylempi kuvasarja). [38, 33]

Symboli F_b tarkoittaa sellaista segmenttiä, jonka signaali on jo tavoittanut, ja symboli F_a puolestaan sellaista segmenttiä, jota signaali ei vielä ole saavuttanut. (Kannattaa huomata, että kontekstia tarkistettaessa symboleja "+" ja "-" ei huomioida mitenkään).

Toiseen suuntaan etenevää signaalia (ks. kuva 24, alempi kuvasarja) voidaan simuloida samaan tapaan L-systeemillä

$$\omega : F_a[+F_a]F_a[-F_a]F_a[+F_a]F_b$$
$$p_1 : F_a > F_b \rightarrow F_b. \qquad (14)$$



Kuva 24. Propagoiva signaali haarautuvassa rakenteessa.

Kuvasta 24 nähdään, että ylemmässä kuvasarjassa signaali (*acropetal signal propagation*) tavoittaa myös kaikki sivusärmät, kun taas alemmassa kuvasarjassa (*basipetal signal propagation*) näin ei tapahdu. Tämä ero johtuu tavasta, jolla juurellinen puu on määritelty. Määritelmän mukaan juurellisessa puussa on suunnattu polku juuresta mihin tahansa puun haaran kärkeen, mutta kahden haaran kärjen välillä ei ole suunnattua polkua. [33]

Yleisesti ottaen peräkkäisten moduulien väliseen kommunikointiin (*endogenous*) perustuvia kehitysprosesseja voidaan analysoida ja mallintaa seuraavien prosesseissa ilmenevien ominaisuuksien avulla [33]:

- *Informaation kulun suunta*. Kaksi perustapausta informaation kulun suunnasta prosessissa on esitetty kuvassa 24. Informaatio voi kulkea prosessissa tietenkin myös molempiin suuntiin, ja samaan prosessiin voi sisältyä monia propagoivia signaaleja, jotka voivat liikkua prosessissa joko samanaikaisesti tai peräkkäin.
- *Haaraumasolmuissa tapahtuvat prosessit*. Haaraumasolmuihin voi tulla informaatiota useista eri lähteistä, ja tätä informaatiota voidaan yhdistellä ja käsitellä haarauma-solmussa ja lähettää käsiteltyä informaatiota edelleen useisiin eri kohteisiin.
- *Informaation esitystapa*. Moduulien välillä tapahtuva informaatiovaihto voi sisältää esim. diskreettejä signaaleja tai muuttujien arvoja.

Tässä työssä ei esitetä enempää yo. ominaisuuksiin perustuvia esimerkkejä, mutta esim. lähteessä [33] on esitetty useita esimerkkejä liittyen näihin ominaisuuksiin.

8. MUITA SOVELLUSALUEITA

Lindenmayer kehitti L-systeemit alunperin mallintamaan kasvua biologisessa kontekstissa [30, 21]. Myöskin Prusinkiewicz ja Hanan etsivät alunperin vastausta siihen, voidaanko L-systeemien avulla mallintaa luonnosta löytyviä erilaisia kasvilajeja, sekä voidaanko L-systeemien avulla tuottaa kuvia erilaisista fraktaaleista. Tässä työssä tähän mennessä esitetyt L-systeemien sovellusalueet on valittu nimenomaan em. historiallisista syistä, eli siis pitäen mielessä, mistä kaikki sai alkunsa.

Vaikka L-systeemeille keksitään nykyään jatkuvasti uusia sovelluskohteita, edelleenkin L-systeemejä hyödynnetään eniten sellaisissa tietokonegrafiikkasovelluksissa, joissa pyritään mallintamaan kasveja ja animoimaan niiden kehittymistä mahdollisimman realistisesti. Varsinaisten animaatioiden ja simulaatioiden lisäksi (ks. esim. [35, 26, 13]) L-systeemejä on käytetty myös eri kasvinosien mallintamisessa ([10, 14]) ja nyttemmin myös kokonaisten ekosysteemien mallintamisessa ([6]). Lisäksi L-systeemien avulla on mallinnettu myös muita biologisia organismeja, kuten jopa ihmisen vatsalaukkua ([9, 8]).

Muista sovellusalueista voidaan tässä yhteydessä esimerkkinä mainita hahmontunnistus ([41, 17]), eri kulttuurien kuva- tai käsityötaiteessa esiintyvien kuvioiden mallintaminen ([36, 7]) sekä fraktaalimusiikki ([36]).

Kaikista yllämainituista sovellusalueista voisi kirjoittaa laajasti, mutta näiden sijaan kohdassa 8.1 esitetään itse kehittämäni sovellus.

8.1 L-SYSTEEMIT RODUNJALOSTUKSESSA

Tässä itselaaditussa esimerkissä mallinnetaan L-systeemin avulla koiran turkin värityksen periytymistä colliekannassa.

Suomessa käytössä oleva rotumääritelmä hyväksyy collieille kolme väriä. Värit ovat soopeli-valkoinen, kolmivärinen (trikolor) ja blue merle. Rotumääritelmä viittaa silmin nähtäviin väreihin. Perimältään nämä koirat voivat kuitenkin olla neljää eri tyyppiä; soopelinvärinen koira voi olla ns. dominanttisoopeli tai ns. trikolor-tekijäinen soopeli. Dominoivasti soopelia väriä periyttävä koira voi saada ainoastaan soopelinvärisiä jälkeläisiä. Trikolor-tekijäinen soopeli voi olla ulkonäöltään kuten dominanttisoopeli, tai ns. tumma soopeli, mahonkisoopeli. Dominanttisoopelin ja trikolor-tekijäisen soopelin erottaminen toisistaan ei ole aina silmämääräisesti mahdollista. [3]

Collieiden sallitut väriyhdistelmät (Suomessa) ja värien periytyminen on esitetty taulukoissa 8.1 – 8.3. Blue merle -koira voitaisiin tietenkin yhdistää toiseen blue merle tai soopelinväriseen koiraan. Käytäntö on kuitenkin osoittanut, että jo pelkästään värien puhtaudenkin takia blue merle -koira kannattaa yhdistää ainoastaan kolmivärisen koiran kanssa. Lisäksi muihin yhdistelmiin liittyy eettisiä ongelmia. Kun yhdistetään kaksi merle geeniä, saadaan väistämättä aikaan 25 % merlehomotsygootteja eli ns. viallisia valkoisia. Homotsygootti-muodossa MM (merle-merle) pennut ovat valkoisia ja syntyvät näkö- ja kuulovammaisina. Koiranjalostukseen liittyviä asioita ei tässä työssä tämän enempää esitetä, mutta erityisesti collieiden värien periytymiseen liittyvät asiat löytyvät esimerkiksi lähteestä [3].

Taulukko 8.1. Sallitut väriyhdistelmät ja värien periytyminen (osa 1).

Toinen vanhemmista	**Toinen vanhemmista**	**Jälkeläiset**
 Dominanttisoopeli	 Dominanttisoopeli	 100 % dominanttisoopeli
	 Trikolor-tekijäinen soopeli	 50 % dominanttisoopeli  50 % trikolor-tekijäinen soopeli
	 Trikolor	 100 % trikolor-tekijäinen soopeli

Taulukko 8.2. Sallitut väriyhdistelmät ja värien periytyminen (osa 2).

Toinen vanhemmista	**Toinen vanhemmista**	**Jälkeläiset**
 Trikolor-tekijäinen soopeli	 Trikolor-tekijäinen soopeli	 25 % dominantti-soopeli  50 % trikolor-tekijäinen soopeli  25 % trikolor
	 Trikolor	 50 % trikolor-tekijäinen soopeli  50 % trikolor

Taulukko 8.3. Sallitut väriyhdistelmät ja värien periytyminen (osa 3).

Toinen vanhemmista	**Toinen vanhemmista**	**Jälkeläiset**	
 Trikolor	 Trikolor	 100 % trikolor	
	 Blue merle	 50 % trikolor	 50 % blue merle

Taulukoissa 8.1 – 8.3 esitetty värien periytyminen mallinnetaan seuraavaksi L-systeemin (15) avulla. Tässä mallissa oletetaan jalostuksessa käytettävien uroskoirien määrä riittäväksi, eli halutun värisiä ja ikäisiä uroskoiria oletetaan olevan aina käytettävissä. Tästä syystä aksioomasta on voitu jättää uroskoirat pois. Tämä on tehty pelkästään sen takia, että malli pysyisi mahdollisimman yksinkertaisena. Mallin voisi tarvittaessa muodostaa vastaamaan myös tarkemmin todellisuutta. Esimerkiksi jos kannassa ei olisi perimältään halutunlaista uroskoiraa, niin ko. perimäyhdistelmää koskevien sääntöjen soveltaminen voitaisiin estää mallissa siihen asti, kunnes sopiva koira kantaan ilmaantuu. Jos tämä malli implementoitaisiin sovellukseksi, jonka avulla koirakannan kehittymistä voitaisiin seurata, niin jalostuksessa käytettävät uroskoirat voitaisiin valita mallin sisällä koirakannasta löytyvien sopivien (väriltään ja iältään) uroskoirien joukosta joko satunnaisesti, tai vaihtoehtoisesti sovelluksen käyttäjä voisi tehdä itse nämä valinnat.

Toisaalta tarvittavat uroskoirat olisi voitu yhtä hyvin lisätä myöskin aksioomaan, koska säännöissä uroskoirien syntyminen on kuitenkin mukana. Sääntöihin uroskoirat on sisällytetty sen takia, että on liian voimakas oletus, että kaikki syntyvät koirat olisivat naaraita. Tässä esimerkissä mallinnetaan siis itse asiassa sellaisen kennelin koirakannan kehitystä, jossa on alunperin kolme narttukoiraa aksiooman mukaan, ja jalostustyö tehdään satunnaisesti uroskoirien valinnan suhteen.

Muodostettavaan L-systeemiin on tehty muitakin yksinkertaistavia oletuksia. Mallissa oletetaan esimerkiksi, että jos narttukoira saa pentuja, niin pentuja kuuluu pentueeseen vain yksi. Pentujen lukumääränkin voi haluttaessa satunnaistaa, kuten myöhemmin esitetään. Narttukoiran täytyy olla vähintään yhden vuoden ikäinen, ennen kuin se voi saada pentuja. Lisäksi kielletyt yhdistelmät on jätetty mallista pois (esimerkiksi aiemmin selitetyt blue merle – blue merle –pennut). Nämä kielletytkin yhdistelmät voitaisiin ottaa malliin mukaan, mutta koska (ainakaan osasta) näin syntyvistä pennuista ei tule elinkelpoisia koiria, on ne jätetty tästä mallista pois. Näiden kiellettyjen yhdistelmien poisjättäminen

mallista vastaa luonnollista karsintaa, koska eivät ko. yksilöt luonnossakaan pärjäisi. Kiellettyjen yhdistelmien poisjättämisen voi perustella silläkin tosiasialla, että jalostustyö nimenomaan pyrkii välttämään tilanteita, jotka saattavat aiheuttaa elinkelvottomien koirien syntymisen.

Viimeinen rajoitus mallissa on koirien elinikä; tässä mallissa kaikkien koirien oletetaan elävän kolmivuotiaiksi. Tämä rajoitus on tehty sen takia, että mallia voidaan hieman myös alustavasti testata ajamalla esimerkkiajot käsin. Myös eliniän voi satunnaistaa, kuten myöhemmin esitetään.

Kun käytetään symboleja S_D (dominanttisoopelinarttu), s_D (dominanttisoopeliuros), S_T (trikolor-tekijäinen soopelinarttu), s_T (trikolor-tekijäinen soopeliuros), T (trikolornarttu), t (trikoloruros), B (blue merle narttu), b (blue merle uros) ja x (koiran ikä), niin malli voidaan esittää muodossa

$\omega : S_D(0)\ S_T(0)\ T(0)$

$p_1 : S_D(x): 0 < x < 3 \rightarrow S_D(x + 1)S_D(0) : 1$
$p_2 : S_D(x): 0 < x < 3 \rightarrow S_D(x + 1)s_D(0) : 1$
$p_3 : S_D(x): 0 < x < 3 \rightarrow S_D(x + 1)S_T(0) : 1$
$p_4 : S_D(x): 0 < x < 3 \rightarrow S_D(x + 1)s_T(0) : 1$
$p_5 : S_D(x): x = 3 \rightarrow \Lambda$
$p_6 : S_D(x): x < 3 \rightarrow S_D(x + 1) : 4$

$p_7 : S_T(x): 0 < x < 3 \rightarrow S_T(x + 1)S_D(0) : 1$
$p_8 : S_T(x): 0 < x < 3 \rightarrow S_T(x + 1)s_D(0) : 1$
$p_9 : S_T(x): 0 < x < 3 \rightarrow S_T(x + 1)S_T(0) : 2$
$p_{10}: S_T(x): 0 < x < 3 \rightarrow S_T(x + 1)s_T(0) : 2$
$p_{11}: S_T(x): 0 < x < 3 \rightarrow S_T(x + 1)T(0) : 1$
$p_{12}: S_T(x): 0 < x < 3 \rightarrow S_T(x + 1)t(0) : 1$
$p_{13}: S_T(x): x = 3 \rightarrow \Lambda$
$p_{14}: S_T(x): x < 3 \rightarrow S_T(x + 1) : 8$

$p_{15}: T(x): 0 < x < 3 \rightarrow T(x + 1)T(0) : 4$
$p_{16}: T(x): 0 < x < 3 \rightarrow T(x + 1)t(0) : 4$
$p_{17}: T(x): 0 < x < 3 \rightarrow T(x + 1)B(0) : 1$
$p_{18}: T(x): 0 < x < 3 \rightarrow T(x + 1)b(0) : 1$

p_{19}: T(x): $0 < x < 3 \rightarrow T(x + 1)S_T(0) : 3$ (15)
p_{20}: T(x): $0 < x < 3 \rightarrow T(x + 1)s_T(0) : 3$
p_{21}: T(x): $x = 3 \rightarrow \Lambda$
p_{22}: T(x): $x < 3 \rightarrow T(x + 1) : 16$

p_{23}: B(x): $0 < x < 3 \rightarrow B(x + 1)B(0) : 1$
p_{24}: B(x): $0 < x < 3 \rightarrow B(x + 1)b(0) : 1$
p_{25}: B(x): $0 < x < 3 \rightarrow B(x + 1)T(0) : 1$
p_{26}: B(x): $0 < x < 3 \rightarrow B(x + 1)t(0) : 1$
p_{27}: B(x): $x = 3 \rightarrow \Lambda$
p_{28}: B(x): $x < 3 \rightarrow B(x + 1) : 4$

p_{29}: $s_D(x)$: $x = 3 \rightarrow \Lambda$
p_{30}: $s_D(x)$: $x < 3 \rightarrow s_D(x + 1)$
p_{31}: $s_T(x)$: $x = 3 \rightarrow \Lambda$
p_{32}: $s_T(x)$: $x < 3 \rightarrow s_T(x + 1)$
p_{33}: t(x): $x = 3 \rightarrow \Lambda$
p_{34}: t(x): $x < 3 \rightarrow t(x + 1)$
p_{35}: b(x): $x = 3 \rightarrow \Lambda$
p_{36}: b(x): $x < 3 \rightarrow b(x + 1)$.

L-systeemin (15) säännöt on ryhmitelty siten, että säännöt 1-6 koskevat dominanttisoopelinarttuja, säännöt 7-14 trikolor-tekijäisiä soopelinarttuja, säännöt 15-22 trikolornarttuja, säännöt 23-28 blue merle –narttuja ja loput säännöistä uroskoiria. Ehto $0 < x < 3$ narttukoiria koskevissa säännöissä varmistaa sen, että vain 1- tai 2-vuotiaat koirat voivat saada pentuja tässä mallissa. Säännöt, joissa esiintyy ehto $x = 3$, tarkoittavat ko. koiran kuolemaa ja ne säännöt, joissa esiintyy ehto $x < 3$, tarkoittavat ko. koiran "täyttävän vuosia" (myös narttukoiran tapauksessa ilman, että se saa pentuja).

Taulukossa 8.4 on esitetty, kuinka todennäköisyydet syntyvien pentujen väritykselle muodostuvat. Taulukossa on oletettu, että on yhtä todennäköistä, että syntyvä pentu on narttu tai uros. Tätä todennäköisyyttä voidaan muuttaa vaihtamalla taulukon "todennäköisyys"-sarakkeessa olevien kaavojen jakajaa (tässä esimerkissä siis 2) halutulla tavalla. Jos esimerkiksi haluttaisiin mallintaa tilanne, jossa urospennun syntymisen todennäköisyys on 45 %, olisi jakajana urospennun tapauksessa 1/0,45, ja narttupennun tapauksessa 1/0,55 (nythän se on itse asiassa 1/0,5). Lisäksi taulukossa on oletettu, että kaikkien värin suhteen

mahdollisten "isäkoirien" todennäköisyydet ovat samat. Myöskin tätä todennäköisyyttä voidaan muuttaa, jos jotkin tietyt värit halutaan mallintaa todennäköisemmiksi kuin toiset. Jos esimerkiksi haluttaisiin mallintaa taulukon 8.4 ylin tilanne (dominanttisoopelinarttu) siten, että *sopivista* uroskoirista 30 % olisi dominoivasti soopelia väriä periyttäviä koiria, 50 % trikolor-tekijäisiä koiria ja loput 20 % uroskoirista olisivat trikoloreja, niin tällöin todennäköisyydet muodostuisivat seuraavasti:

$$S_{D,}\, s_D = \frac{30\% * 100\% + 50\% * 50\%}{2} \quad \text{ja} \quad S_T, s_T = \frac{50\% * 50\% + 20\% * 100\%}{2}.$$

Jos vielä lisäksi yo. tilanteessa haluttaisiin mallintaa tilanne, jossa tietyllä väriperimällä varustetun narttukoiran syntyvä pentu on 45 % todennäköisyydellä uros, niin todennäköisyydet laskettaisiin kaavoilla

$$S_D = \frac{30\% * 100\% + 50\% * 50\%}{1/0.55}, \quad s_D = \frac{30\% * 100\% + 50\% * 50\%}{1/0.45},$$

jne., eli yleisesti ottaen kaavalla

$$\frac{\sum(pi * pj)}{1/ps},$$

missä pi tarkoittaa isän väriperimän todennäköisyyttä, pj syntyvän pennun väriperimän todennäköisyyttä ja ps pennun sukupuolen todennäköisyyttä.

Taulukko 8.4. Syntyvän pennun sukupuolen ja väriperimän todennäköisyydet.

'Äiti'	'Isä'	Jälkeläiset	Todennäköisyys
		 100 %	S_D, $s_D = \frac{1/3*100\%+1/3*50\%}{2} = 0,25$
		 50 % 50 %	S_T, $s_T = \frac{1/3*100\%+1/3*50\%}{2} = 0,25$
		 100 %	
		 50 % 50 %	S_D, $s_D = \frac{1/3*50\%+1/3*25\%}{2} = 0,125$
		 25 % 50 % 25 %	S_T, $s_T = \frac{1/3*50\%+1/3*50\%+1/3*50\%}{2} = 0,25$
		 50 % 50 %	T, $t = \frac{1/3*25\%+1/3*50\%}{2} = 0,125$
		 100 %	S_T, $s_T = \frac{1/4*100\%+1/4*50\%}{2} = 0,1875$
		 50 % 50 %	T, $t = \frac{1/4*50\%+1/4*100\%+1/4*50\%}{2} = 0,25$
		 100 %	B, $b = \frac{1/4*50\%}{2} = 0,0625$
		 50 % 50 %	
		 50 % 50 %	B, $b = \frac{1*50\%}{2} = 0,25$ T, $t = \frac{1*50\%}{2} = 0,25$

Taulukon 8.4 todennäköisyydet on siirretty L-systeemiin (15) seuraavalla tavalla. Käsitellään tässä yhteydessä esimerkkinä trikolor-nartun tapaus. Tässä tapauksessa todennäköisyydet vaihtelevat taulukossa 0,0625 ja 0,25 välillä. Nämä todennäköisyydet pitäisi siis siirtää L-systeemiin (15) siten, että kunkin säännön soveltamisen todennäköisyys L-systeemissä vastaa taulukon 8.4 vastaavaa todennäköisyyttä. Kaikessa yksinkertaisuudessaan tämä saadaan aikaan muuttamalla taulukon todennäköisyydet kokonaisluvuiksi siirtämällä pilkkua riittävän monta dekadia oikealle, ja etsimällä tämän jälkeen syntyneiden kokonaislukujen suurin yhteinen tekijä. Kun nämä kokonaisluvut jaetaan löydetyllä

suurimmalla yhteisellä tekijällä, on samalla muodostettu L-systeemin säännöille todennäköisyyskertoimet. Tämän esimerkin tapauksessa syntyneet todennäköisyyskertoimet säännöille p15 - p20 ovat siis 4, 4, 1, 1, 3 ja 3.

L-systeemiin (15) on mallinnettu sellainen tilanne, että jos narttukoira on synnytysikäinen (1-2 vuotias), niin on yhtä todennäköistä, että narttukoira ei synnytä kuin että se synnyttäisi. Tämän takia L-systeemiin on lisätty sääntöihin p6, p14, p22 ja p28 todennäköisyys-kertoimet siten, että se on yhtä suuri kuin kunkin sääntöryhmän pennun syntymää mallintavien sääntöjen todennäköisyyskertoimet yhteenlaskettuna.

Todellisuudessa kovin vanhoilla koirilla ei enää teetetä pentuja, eli sääntöihin p6, p14, p22 ja p28 liitettävä todennäköisyyskerroin kannattaisi käytännössä muodostaa sopivan ikään perustuvan funktion avulla. Tämä esimerkki on kuitenkin luonteeltaan vain periaatteellinen, joten esimerkin L-systeemi on tarkoituksella pidetty pelkistetyn yksinkertaisena.

Jos syntyvien pentujen lukumäärä halutaan satunnaistaa, niin jokaiselle pentuekoolle tarvitaan oma sääntöryhmä L-systeemissä (15). Näin toimien L-systeemin (15) sääntöjen lukumäärä kasvaa nopeasti turhan suureksi. Tämä puolestaan johtaa siihen, että L-systeemin (15) esitystapaa täytyisi muuttaa tiivistetymmäksi siirtämällä informaatiota eksplisiittisistä säännöistä parametreiksi sääntöjen moduuleihin. Esimerkiksi koiran sukupuolen tai väriperimän voisi aivan hyvin esittää moduulin sisällä parametrin avulla. Mutta kuten jo aiemmin on useaan kertaan todettu, tämä esimerkki on vain periaatteellinen, joten L-systeemin (15) esitystapakin on valittu mahdollisimman havainnolliseksi.

Jos kuitenkin nykyiseen L-systeemiin haluttaisiin lisätä dominanttisoopelinarttua koskevat säännöt pentuekoolle kaksi, niin sääntöryhmään p1-p6 pitäisi lisätä yhteensä yhdeksän sääntöä (kaikki vaihtoehdot pentujen sukupuolen ja väriperimän suhteen):

$S_D(x)\colon 0 < x < 3 \rightarrow S_D(x+1)S_D(0)S_D(0)$,
$S_D(x)\colon 0 < x < 3 \rightarrow S_D(x+1)S_D(0)s_D(0)$,
$S_D(x)\colon 0 < x < 3 \rightarrow S_D(x+1)s_D(0)s_D(0)$,
$S_D(x)\colon 0 < x < 3 \rightarrow S_D(x+1)S_T(0)S_T(0)$,
$S_D(x)\colon 0 < x < 3 \rightarrow S_D(x+1)S_T(0)s_T(0)$,
$S_D(x)\colon 0 < x < 3 \rightarrow S_D(x+1)s_T(0)s_T(0)$,
$S_D(x)\colon 0 < x < 3 \rightarrow S_D(x+1)S_D(0)S_T(0)$,
$S_D(x)\colon 0 < x < 3 \rightarrow S_D(x+1)S_D(0)s_T(0)$,
$S_D(x)\colon 0 < x < 3 \rightarrow S_D(x+1)S_T(0)s_D(0)$.

Samalla kun sääntöjen lukumäärä kasvaa voimakkaasti, niin myöskin todennäköisyyskertoimien laskeminen muuttuu työläämmäksi, riippuen siitä, miten eri pentuekokojen ja sukupuolijakauman todennäköisyyksiä halutaan painottaa. Kuten aiemmin on todettu, käytännössä päästään helpommalla, kun koiran sukupuoli ja väriperimä esitetään parametreina moduulien sisällä. Esimerkiksi aiemmin esitetyt 15 sääntöä dominanttisoopelinartulle (pentuekoot yksi ja kaksi) voitaisiin esittää seuraavien 4 säännön avulla:

$X(x, y, z)$): $0 < x < 3 \rightarrow X(x + 1, y, z)X(0, f(y), g(z))$,
$X(x, y, z)$): $0 < x < 3 \rightarrow X(x + 1, y, z)X(0, f(y), g(z))\ X(0, f(y), g(z))$,
$X(x, y, z)$: $x = 3 \rightarrow \Lambda$,
$X(x, y, z)$: $x < 3 \rightarrow X(x + 1, y, z)$,

missä X tarkoittaa pelkistetysti koiraa, x koiran ikää, y koiran sukupuolta ja z koiran väriperimää. Funktioita f(x) ja g(x) koskevat merkinnät ovat ehkä hieman harhaanjohtavia, käytännössä em. funktioiden tulee olla syntaktisesti oikeinmuodostettuja aritmeettisia ilmauksia (ks. kohta 3.5). Käytännössä f(x) määrittäisi syntyvän koiran sukupuolen esimerkiksi siten, että 0 tarkoittaa narttua ja 1 urosta, ja g(x) määrittäisi koiran väriperimän sopivalla tavalla koodattuna. Todennäköisyyskertoimet on tässä tapauksessa kohtuullisen helppo määrittää, käytännössä eri pentuekokoja olisi varmastikin syytä painottaa todellisuutta vastaavalla tavalla.

Eli tällä tavalla määriteltynä L-systeemiin tarvitaan vain yksi sääntö kullekin pentuekoolle. Jos tällainen koiran turkin värin (tai jonkin muun perinnöllisen tekijän) periytymistä mallintava sovellus haluttaisiin implementoida, niin luonnollisesti yo. tapa sääntömäärittelyyn on huomattavasti järkevämpi kuin L-systeemissä (15) esitetty tapa.

Implementoitavassa mallissa myös koirien elinikä täytyisi satunnaistaa. Eliniän satunnaistaminen tapahtuisi käytännössä sopivan funktion h(x) avulla, jolloin L-systeemin (15) säännöissä esiintyvä ehto $0 < x < 3$ voitaisiin korvata ehdolla $0 < x$ ja ehdot $x = 3$ ja $x < 3$ voitaisiin korvata ehdolla $x < h(x)$. Funktio h(x) määriteltäisiin siten, että se palauttaa lukua x suuremman arvon sitä todennäköisemmin, mitä suurempi luku x on. Tämä tarkoittaa sitä, että koiran vanhetessa sen kuolemisen todennäköisyys kasvaa. (Käytännössä on harvinaista, että collie eläisi yli 15 vuotiaaksi).

Liitteessä 1 on esitetty esimerkkinä yksi syntynyt derivaatiopuu, kun L-systeemin (15) sääntöjä on alettu soveltaa aksioomaan. Koska esimerkkiajot suoritettiin pelkkänä

pöytätestinä, testiajoja suoritettiin ainoastaan 20 kertaa. Näin ollen saatujen tulosten perusteella ei kannata tehdä mitään kovin pitkälle meneviä johtopäätöksiä tai tilastollisia analyysejä, mutta jotain tuloksista kuitenkin uskaltanee päätellä.

Ensinnäkin voidaan todeta, että uroskoirat ovat tässä mallissa hieman turhaan mukana, koska niillä ei ole mitään käytännön merkitystä iteraatioiden etenemisessä. Uroskoirillekin on käyttöä siinä vaiheessa, kun mallia kehitetään siten, että lisääntymisessä tarvittavat uroskoirat tulevat mallin sisältä. Silloin tosin mallinnetaan periaatteessa jo koko colliekantaa; tässä mallissahan mallinnetaan colliekannan osajoukkoa.

Jo ennen mallin testausta oli selvää, että koirien elinikä ja pentuekoko oli mallinnettu liian pieniksi. Testiajot todistavat samoin, sillä vaikka testiajoista kaikkein pisin kestikin 23 iteraatiokierrosta ennen kuin viimeinenkin koira oli kuollut mallista, oli testiajojen pituuden keskiarvo 9,9 ja mediaani 8. Valmiiksi implementoidulla sovelluksella olisikin varsin mielenkiintoista tutkia, kuinka pentuekoko tai aksioomana olevien koirien lukumäärä vaikuttaa iteraatioiden kulkuun. Pentuekoolle voisi esimerkiksi yrittää määrittää jonkinlaisen raja-arvon, jota pienemmällä määrällä koirakanta kuolee sukupuuttoon, ja jota suuremmalla kanta pysyy hengissä loputtomiin. Tätä raja-arvoa voisi sitten verrata reaalimaailman arvoon keskimääräisestä pentuekoosta, ja miettiä sen perusteella, onko malli laadittu siinä mielessä todellisuutta vastaavaksi.

Yhteenvetona varsin suppeista testeistä voidaan todeta, että syntyneiden pentujen sukupuolijakauma (n. 49 % uroksia ja 51 % narttuja) sekä todennäköisyys nartun synnyttämiselle (n. 47 % synnytysikäisistä nartuista synnytti, 53 % ei synnyttänyt) näyttäisi mallissa vastaavan sitä tilannetta, jota L-systeemillä haluttiin mallintaa. (Tämäkin seikka tosin tiedettiin etukäteen, käytännössä em. testeissä tuli testattua käytetyn satunnaislukugeneraattorin oikeellisuus).

Uusien parametrien lisääminen malliin on varsin helppoa. Esimerkiksi jonkin tietyn perinnöllisen sairauden aiheuttava geeni voitaisiin lisätä malliin, ja tutkia mallin avulla sen esiintymistä koirakannassa. Perinnöllisissä sairauksissa tyypillinen sukupolvien yli hyppiminen on helposti mallinnettavissa parametroitujen L-systeemien avulla, joten reaalimaailmaa vastaava tilanne on mallinnettavissa tällä tavalla. Lisäksi näin kloonauksen aikakaudella saattaisi olla mielenkiintoista tutkia, kuinka esimerkiksi jonkin mallinnetun koirakannan geeniperimä (esim. juuri jonkin perinnöllisen sairauden suhteen) kehittyy, kun kloonattavaksi valitaan eri geeniperimällä varustettuja yksilöitä.

8.2 L-SYSTEEMEIHIN LIITTYVIÄ ONGELMIA

L-systeemit sopivat periaatteessa hyvin kasvin kehittymistä mallintavien tietokoneanimaatioiden tekemiseen, koska L-systeemit kehitettiin alunperin mallintamaan nimenomaan kasvuprosessia. Tietokoneanimaatiossa peräkkäisten kuvien (frame) täytyy kuitenkin olla riittävän samanlaisia, jotta animaatio näyttäisi etenevän jouheasti. L-systeemeillä tuotettujen merkkijonojen avulla toteutetut kasvia kuvaavat kuviot eivät yleensä täytä tätä ehtoa. Ts. jonkin kasvin kehittymistä kuvaavan L-systeemin avulla tuotetut kuviot ovat liian erilaisia iteraatiokierroksilla n ja n+1 käytettäväksi sellaisenaan jossakin animaatiossa. [29]

Ns. differentiaalisten L–systeemien (*differential L-systems, dL-systems*) avulla voidaan välttää yllä kuvattu ongelma [31]. Differentiaalisissa L-systeemeissä aika ei ole diskreetti vaan jatkuva muuttuja, jolloin L-systeemin moduulit kehittyvät jatkuvasti myös iteraatiokierrosten välillä. Differentiaalisten L-systeemien määritelmä löytyy mm. lähteestä [35].

Jos L-systeemejä tarkastellaan puhtaasti biologisesta näkökulmasta, niin niiden avulla voisi kuvitella olevan mahdollista muotoilla kasvin kehittymiseen liittyviä kysymyksiä myöskin (matemaattisen) formaalisti. Käytännössä tämä tilanne tuntuu kuitenkin melko kaukaiselta, ensinnäkin sen takia, että kasveille ei ole olemassa tarkkaa matemaattista mallia. Toinen ongelma on se, että L-systeemien teoriaa koskeva tutkimustyö ei ole juurikaan ottanut huomioon biologisen mallintamisen tarpeita. Pääosa L-systeemien teoriaa koskevista tuloksista koskee nimittäin ei-parametroituja 0L-systeemejä (ilman haarautuvien rakenteiden kuvaamiseen tarvittavia mekanismeja, ja ilman tuotettujen merkkijonojen geometrista tulkintaa), kun taas biologisten ilmiöiden (kasvien) mallintamisessa hyödynnetään parametreja, eri kasvinosien välistä vuorovaikutusta sekä mallinnettavien rakenteiden geometrisia ominaisuuksia. [32]

Tämä tilanne ilmenee itse asiassa myös tässä työssä, kun tarkastelee työn teoriaosuudessa käsiteltyjä tuloksia suhteessa työn sovelluksia käsittelevässä osuudessa esitettyihin L-systeemeihin.

Alunperin L-systeemit suunniteltiin *suljetuiksi* malleiksi siinä mielessä, että niiden avulla ei ollut tarkoitus pystyä mallintamaan vuorovaikutusta kasvin ja sitä ympäröivän elinympäristön välillä. Kun tämä vuorovaikutus haluttiin myöhemmin ottaa mukaan L-systeemeihin, muodostui samalla uusia, mutkikkaampia L-systeemityyppejä. T0L-systeemit (ks. kohta 3.2) oli ensimmäinen L-systeemityyppi, johon oli sisällytetty elinympäristön vaikutus mallinnettavaan kasviin. T0L-systeemeissä elinympäristön muutos mallinnetaan käytännössä kvalitatiivisella tavalla, eli eri olosuhteita kuvaamaan käytetään eri

sääntöjoukkoja. Seuraava kehitysaskel L-systeemityyppien kehityksessä oli parametroidut L-systeemit (kohta 3.5). Niiden avulla elinympäristön muutosta voidaan mallintaa myöskin kvantitatiivisesti, kuten esimerkiksi lähteessä [15] on esitetty. Ko. lähteessä mallinnettiin tilanne, kuinka päivittäinen lämpötilan vaihtelu vaikuttaa kasvin kasvunopeuteen.

Vaikka tässä työssä L-systeemityyppien esittely lopetettiin parametroituihin L-systeemeihin, ei niiden kehittely kuitenkaan loppunut vielä siihen. Ympäristösensitiiviset L-systeemit (*environmentally-sensitive L-systems*) lisäsivät malleihin ympäristön lokaalien ominaisuuksien vaikutukset [37]. Kun parametroiduissa L-systeemeissä ympäristöä mallintavat parametrit koskevat globaalisti kaikkia objekteja mallin sisällä, voidaan ympäristösensitiivisissä L-systeemeissä mallintaa ympäristöä koskeviin parametreihin eroja mallin sisällä ympäristön lokaalia tilaa kuvaavien funktioiden avulla.

Ympäristösensitiivisissä L-systeemeissä ympäristön lokaali tila on kuitenkin aina vakio, mikä käytännössä tarkoittaa sitä, että informaatio kulkee mallissa vain ympäristöstä mallinnettavaan objektiin. Ns. avoimet L-systeemit (*open L-systems*) lisäsivät ympäristösensitiivisiin L-systeemeihin myös toisen informaation kulun suunnan, eli mallinnettavasta objektista sitä ympäröiviin ympäristötekijöihin. Avoimissa L-systeemeissä ympäristöä ei enää kuvata yksinkertaisen funktion avulla, vaan ympäristö mallinnetaan aktiiviseksi mallinnettavia objekteja ympäröiväksi prosessiksi, joka reagoi objekteista tulevaan informaatioon. [24]

Uusia L-systeemityyppejä syntyy edelleen, kun L-systeemejä aletaan soveltaa uusilla sovellusalueilla. Tässä työssä ei ole edes yritetty esitellä kaikkia kirjallisuudesta löytyviä L-systeemityyppejä, mutta kaikkein yleisimmät L-systeemityypit tästä työstä löytyvät.

Lopuksi voidaan vielä todeta käytetystä lähdemateriaalista sen verran, että lähteinä on käytetty ehkä hieman korostetusti Prusinkiewiczin (et. al.) tutkimuksia varsinkin L-systeemien sovelluksia koskevassa osuudessa. Myös muiden tutkijoiden kirjoittamaa L-systeemejä koskevaa materiaalia on käytettävissä erittäin paljon, mutta mielestäni Prusinkiewicz edustaa L-systeemien soveltamista biologisessa kontekstissa käsittelevän tutkimuksen ehdotonta huippua.

LÄHTEET

[1] Alfonseca, M., Ortega, A., Representation of fractal curves by means of L systems. *APL Quote Quad*, **26**, 4 (1996), 13-21.

[2] Alfonseca, M., Ortega, A., A study of the representation of fractal curves by L systems and their equivalences. *IBM Journal of Research & Development*, **41**, 6 (1997), 727-736.

[3] Alvasto, A., Hirvonen, S., *Collie*. Tammi, 1998.

[4] Culik, K., Karhumäki, J., A New Proof For The D0L Sequence Equivalence Problem And Its Implications, University of Waterloo, Department of Computer Science, Research Report CS-85-30, 1985.

[5] Dai, M., Ozawa, K., Simulation of worn-out cloth textures by doubly stochastic L-systems. *Image and Vision Computing*, **16**, 5 (1998), 363-371.

[6] Deussen, O., Hanrahan, P., Lintermann, B., Mech, R., Pharr, M., Prusinkiewicz, P., Realistic modeling and rendering of plant ecosystems. In *Proceedings of the 25th Annual Conference on Computer Graphics*, 275-286, 1998.

[7] Drewes, F., Klempien-Hinrichs, R., Picking knots from trees. The syntactic structure of Celtic knotwork. In Anderson, M., Cheng, P. and Haarlev, V. (eds.*), Theory and Application of Diagrams. Lecture Notes in Artificial Intelligence 1889*, Springer, 2000, 89-104.

[8] Durikovic, R., Kaneda, K., Yamashita, H., Visual modeling of stomach growth on the basis of L-systems. In *Proceedings of 1997 International Conference on Shape Modeling and Applications, 121-128*, 1997.

[9] Durikovic, R., Kaneda, K., Yamashita, H., Animation of biological organ growth based on L-systems. *Computer Graphics Forum*, **17**, 3 (1998), C1-13, C365.

[10] Fowler, D., Prusinkiewicz, P., Battjes, J., A collision-based model of Spiral Phyllotaxis. *Proceedings of SIGGRAPH '92,* In *Computer Graphics, 26, 2, (1992),* ACM SIGGRAPH, New York, 361-368.

[11] Gleick, J., *Kaaos*. Art House, 1990.

[12] Green, D., Fractals and scale. http://life.csu.edu.au/complex/tutorials/tutorial3.html (14.9.2000)

[13] Hammel, M., Prusinkiewicz, P., Remphrey, W., Davidson, C., Simulating the development of Fraxinus pennsylvanica shoots using L-systems. In *Proceedings of the Sixth Western Computer Graphics Sympsium,* 49-58, 1995.

[14] Hammel, M., Prusinkiewicz, P., Wyvill, B., Modelling compound leaves using implicit countours. In Tosiyasu L. Kunii (ed.), Visual Computing: Integrating Computer Graphics with Computer Vision, 119-212. Springer-Verlag, 1992.

[15] Hanan, J., Virtual plants - integrating architectural and physiological models. *Environmental Modelling & Software,* **12**, 1 (1997), 35-42.

[16] Herman, G. T., Rozenberg, G., *Developmental Systems and Languages.* North-Holland / American Elsevier, 1975.

[17] Holliday, D. J., Samal, A., Object recognition using L-system fractals. *Pattern Recognition Letters,* **16**, 1 (1995), 33-42.

[18] Hopcroft, J. E., Ullman, J. D., *Formal Languages and Their Relation to Automata*. Addison-Wesley, 1969.

[19] Karhumäki, J., On the equivalence problem for binary D0L systems, *Information and Control*, **50**, 276-284, 1981.

[20] Kari, L., Rozenberg, G., Salomaa, A., L Systems. In Rozenberg, G and Salomaa, A., (eds), *Handbook of Formal Languages.* Springer-Verlag, 1996.

[21] Lindenmayer, A., L systems in their biological context. *Proceedings of the 1974 Conference on Biologically Motivated Automata Theory,* 65-69, 1974.

[22] Mandelbrot, B., *The Fractal Geometry of Nature.* W. H. Freeman, 1982.

[23] Maurer, H.A., Salomaa, A., Wood, D., Pure grammars, *Information and Control* 44, 47-72. 1980.

[24] Mech, R., Prusinkiewicz, P., Visual models of plants interacting with their environment. In *Proceedings of SIGGRAPH 96,* 379-410. 1996.

[25] Mäkinen, E., Laskennan teorian perusteet. Tampereen yliopisto, tietojenkäsittelyopin laitos, raportti C-1996-2, 1996.

[26] Noser, H., Thalmann, D., Turner, R., Animation based on the interaction of L-systems with vector force fields. *Proceedings Computer Graphics International 1992.* Springer-Verlag, 1992.

[27] Peitgen, H-O., Saupe, D. (eds), *The Science of Fractal Images.* Springer-Verlag, 1988.

[28] Peitgen, H-O., Jürgens, H., Saupe, D., *Chaos and Fractals. New Frontiers of Science.* Springer-Verlag, 1992.

[29] Prusinkiewicz, P., Applications of L-systems to computer imagery. *Graph-Grammars and their Application to Computer Science. 3rd International Workshop,* 534-548. Springer-Verlag, 1987.

[30] Prusinkiewicz, P., Graphical applications of L-systems. *Proceedings of Graphics Interface '86 and Vision Interface '86,* 247-253. 1986.

[31] Prusinkiewicz, P., Modeling and visualization of biological structures. In *Proceeding of Graphics Interface '93,* 128-137. 1993.

[32] Prusinkiewicz, P., Hammel, M., Hanan, J., Mech, R., L-systems: from the theory to visual models of plants. In Michalewich, M. T., (ed), *Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences.* 1996.

[33] Prusinkiewicz, P., Hammel, M., Hanan, J., Mech, R., Visual models of plant development. In Rozenberg, G and Salomaa, A., (eds), *Handbook of Formal Languages.* Springer-Verlag, 1996.

[34] Prusinkiewicz, P., Hammel, M., Mech, R., Hanan, J., The artificial life of plants. In *Artificial life for graphics, animation, and virtual reality,* SIGGRAPH ‘95 Course Notes, 1-1 - 1-38. 1995.

[35] Prusinkiewicz, P., Hammel, M., Mjolsness, E., Animation of plant development. In *Proceedings of SIGGRAPH 93*, 351-360. 1993.

[36] Prusinkiewicz, P., Hanan, J., Lindenmayer systems, fractals and plants. *Lecture Notes in Biomathematics*, **79** (1989).

[37] Prusinkiewicz, P., James, M., Mech, R., Synthetic topiary. In *Computer Graphics Proceedings*, 351-358. 1994.

[38] Prusinkiewicz, P., Lindenmayer, A., *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

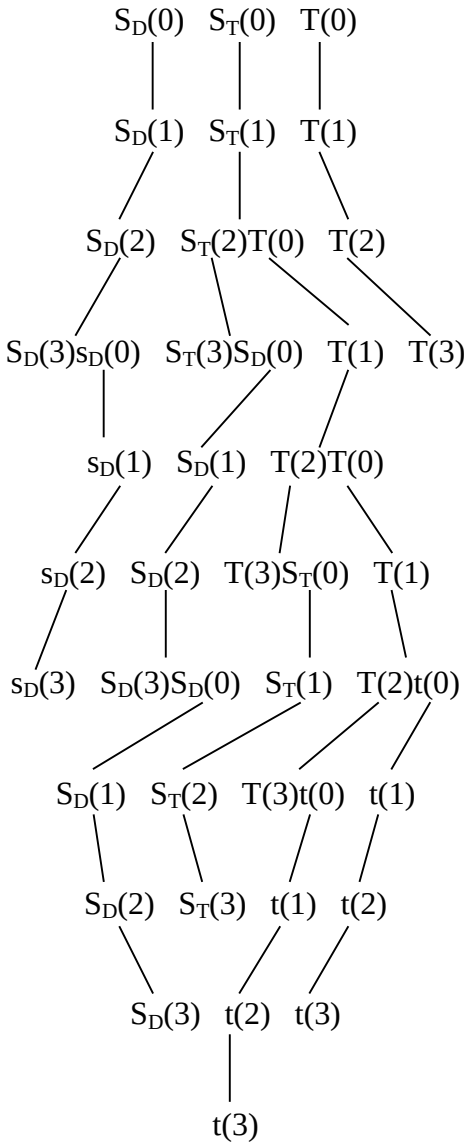
[39] Rozenberg, G., Salomaa, A., *The Mathematical Theory of L Systems*. New York Academic Press, 1980.

[40] Salomaa, A., *Jewels of Formal Language Theory.* Computer Science Press, 1981.

[41] Samal, A., Peterson, B., Holliday, D. J., Recognizing plants using stochastic L-systems. In *Proc. of ICIP-94*, **1**, 183-187.

[42] Thulasiraman, K., Swamy, M., N., S., *Graphs: Theory and Algorithms*. New York Wiley Corporation, 1992.

[43] Vaario, J., Ohsuga, S., Hori, K., Connectionist modeling using Lindenmayer systems. *Information Modeling and Knowledge Bases: Foundations, Theory, and Applications,* 1991, 496-510.



L-systeemin (15) tuottama esimerkkiderivaatiopuu.