

Reprezentarea sistemelor Lindenmayer ca documente XML

Sabin-Corneliu Buraga

Facultatea de Informatică, Universitatea „A.I.Cuza”, Iași

busaco@infoiasi.ro

Rezumat

În această lucrare se prezintă LSML (L-Systems Modeling Language) – un limbaj structurat de definire a sistemelor Lindenmayer (L-sisteme), ca aplicație a meta-limbajului XML (eXtensible Markup Language). Documentele LSML vor putea fi utilizate pe Web și integrate în alte documente XML, precum XHTML și Web Schematics sau în aplicații hipermedia distribuite.

Cuvinte cheie: XML, Web, sisteme dinamice, fractali, documente, grafica, limbaje formale

1. INTRODUCERE

Dezvoltarea accentuată a Internetului și mai ales a Web-ului a dus la apariția meta-limbajului structurat XML (*eXtensible Markup Language*), descendent al SGML (*Standard Generalized Markup Language*). Limbajul XML oferă suport pentru specificarea și validarea – printr-o construcție definind structura unui document denumită DTD (*Document Type Definition*) – a documentelor complexe. XML este independent de platforma hardware și software, fiind un meta-limbaj simplu de utilizat, folosit pentru definirea unor noi tipuri de documente prin intermediul unei specificații formale. Un document XML constă dintr-o secvență de *elemente* (marcatori delimitați de < și >), fiecare element putând include recursiv alte elemente sau text și putând avea atașate diverse *atribute*. Fiecare element va poseda un marcator de început și un marcator de sfârșit, regulile de imbricare a elementelor și numărul, modul de apariție și tipul atributelor fiind definite în mod formal de *definiția tipului de document* (DTD) atașată fiecărei clase de documente care se dorește a fi specificată.

Datorită flexibilității limbajului XML, în prezent există o multitudine de aplicații, una dintre cele mai importante fiind interschimbarea datelor electronice între diverse locații de pe Web. Alte direcții de cercetare includ: *MathML*, un limbaj menit să reprezinte formulele matematice într-un mod consistent și independent de editor, *QL* (*Query Language*), destinat modelării interogărilor de baze de date distribuite, sau *EFDL* (*Extensible Forms Description Language*), limbaj utilizat pentru descrierea formularelor electronice etc.

Vom defini un tip de document XML pentru reprezentarea sistemelor dinamice Lindenmayer (fractalii L-sistem), specificând DTD-ul pentru limbajul *LSML* (*L-Systems Modeling Language*) și vom ilustra o parte dintre posibilele lui utilizări viitoare. Se va urmări și definirea riguroasă a L-sistemelor pe baza căreia vom dezvolta elementele limbajului LSML.

2. L-SISTEME

2.1 Definierea formală

Un *sistem Lindenmayer* (*L-sistem*) este un tip particular de sistem dinamic simbolic în care evoluția sistemului posedă o interpretare geometrică bazată pe grafica *turtle* și a fost introdus de Aristid Lindenmayer (1968) pentru a modela dezvoltarea biologică. Un L-sistem poate fi definit formal de un 6-uplu:

$$Ls = \langle \textit{alphabet}, \textit{axiom}, \textit{rules}, \textit{order}, \textit{angle}, \textit{colour} \rangle$$

fiind construit din segmente de linii folosind un set de reguli (*rules*) care specifică diverse comenzi de desenare.

Pornind de la un șir inițial numit *axiomă* (*axiom*) sau inițiator, un șir de simboluri $w \in V^*$, unde V este un alfabet (*alphabet*) de simboluri, vor fi aplicate de un anumit număr de ori (definind ordinul fractalului, dat de $order \in \mathbb{N}^+$) anumite reguli de transformare pentru a produce un șir final conținând comenzi folosite pentru reprezentarea fractalului dorit. Se mai precizează valoarea unghiului $angle \in (1, 360)$ și culoarea inițială ($colour \in \mathbb{N}$) folosite în cadrul regulilor de desenare. Regulile sunt similare producțiilor gramaticilor din teoria limbajelor formale, putând conține atât comenzi de desenare, cât și neterminali folosiți în generarea șirului final. Regulile (producțiile sau legile de rescriere) au forma $n \rightarrow w$, unde $n \in V$, $w \in V^*$, cu $V = \mathbb{N} \cup C$, în care \mathbb{N} este mulțimea neterminalilor folosiți în definirea L-sistemului, iar C este mulțimea comenzilor de desenare permise (vezi secțiunea 2.4). Pot exista, de asemenea, și reguli de ștergere. Dacă un simbol $n \in V$ nu are asociată o producție dată în mod explicit, atunci se consideră că este asociat lui însuși și reprezintă o *constantă* a sistemului Lindenmayer. Axioma poate conține comenzi de desenare și măcar un simbol neterminal (se permite folosirea ca neterminali a numelor comenzilor de desenare).

Evoluția unui L-sistem este definită de o secvență (g_i) , $i = 0, 1, 2, \dots$, unde fiecare generație g_i este un cuvânt din V^* obținut prin aplicarea regulilor (în paralel) tuturor simbolurilor din cuvântul g_{i-1} , prima generație fiind axioma. Evoluția unui L-sistem modelează într-o manieră simplă și concisă dezvoltarea unei populații de forme biologice.

2.2 Tipuri

În funcție de tipul de reguli de producție, un L-sistem poate fi *independent* sau *dependent* de context. La fel, în funcție de numărul de reguli ce pot fi aplicate la un pas în evoluția unui sistem, un L-sistem poate fi *determinist* sau *nederminist*. Dacă există mai mult de o regulă de aplicare pentru un anumit simbol (de exemplu avem $a \rightarrow w_1$ și $b \rightarrow w_2$, cu $a, b \in V$, $w_1, w_2 \in V^*$), atunci este nevoie de un criteriu de alegere de aplicare a unei reguli. În cazul în care această alegere se realizează în funcție de anumite probabilități asociate fiecărei reguli în parte, sistemul Lindenmayer se numește *stochastic*. În continuare ne vom referi numai la L-sisteme deterministe, independente de context, denumite și *DOL-sisteme*. O clasă de DOL-sisteme este cea a *BDOL-sistemelor* (*bracketed L-Systems*), utile pentru modelarea matematică a structurilor (biologice) ramificate.

De fapt, limbajele generate de L-sisteme aparțin clasei limbajelor recursiv-enumerabile. Aspectele specifice L-sistemelor sunt acelea referitoare la evoluția paralelă a unui cuvânt (generații) pe baza generației precedente și la reprezentarea grafică a șirului rezultat în urma unui număr de generații succesive.

Radomir Mech și Przemyslaw Prusinkiewicz (1996) au propus un nou tip de L-sistem numit *Open L-system*, în vederea modelării formale a interacțiunii dintre plante și mediu prin intermediul proceselor comunicante.

2.2.1 Exemplu

Considerăm următorul sistem Lindenmayer denumit *L-sistem Fibonacci*:

$$F = \langle \{ a, b \}, a, \{ a \rightarrow b, b \rightarrow ba \}, \omega \rangle$$

unde am omis *angle* și *colour*. Putem pune în corespondență cu acest L-sistem relațiile recursive definind șirul Fibonacci ($f_0 = f_1 = 1, f_{n+2} = f_{n+1} + f_n$, cu $n \geq 0$). Se demonstrează ușor că numărul de organisme la a n -a generație g_n este $|g_n| = f_n$. Putem observa că $g_{n+2} = g_{n+1}g_n$ (de exemplu $g_7 = (babbababbabba)(babbabab) = g_6g_5$).

2.3 Utilizarea L-sistemelor

În cadrul unui L-sistem dinamic se poate pune în evidență un *proces local* (afectând fiecare organism individual și fiind reprezentat de aplicarea unei reguli de producție unui simbol particular) care duce la apariția în fapt a unui *proces global*, la nivelul întregii populații. Evoluția locală se desfășoară în paralel, în întreaga populație, similar procesului de dezvoltare observat la plante.

Din acest aspect derivă o parte din preocupările matematicienilor de a modela dezvoltarea biologică cu ajutorul L-sistemelor. O altă utilizare este studiul dinamic al suprafețelor geodezice (L-sisteme Thue-Morse) ori simularea țesuturilor nervoase (arbori de dendrite și axoni neuronali). Se cunosc și instanțe ale problemei comisului voiajor modelate ca sisteme Lindenmayer (Adrian Mariano, 1995). Diverse L-sisteme (e.g. *Penrose tiles*) au fost utilizate pentru rezolvarea problemei acoperirii planului.

2.4 Grafica turtle

Seymour Papert (1967) a inventat grafica *turtle* ca un sistem pentru translatarea unei secvențe de simboluri ale unui alfabet în mișcări ale unui automat, numit broască (*turtle*) pe un mediu de vizualizare (ecran grafic), oferind astfel copiilor un obiect programabil pentru facilitarea învățării informaticii (prin intermediul limbajului LOGO). Sistemul de bază este următorul: fixăm un pas d reprezentând distanța în pixeli parcursă de broască în cadrul unei operațiuni de mișcare. Se mai definește un unghi u , în mod uzual $u = 360/n$, cu $1 \leq n \leq 360$. Pentru reprezentări mai complexe se poate modifica și culoarea segmentelor de dreaptă figurate. În acest mod, L-sistemul este definit complet.

Comenzile standard de desenare (aparținând multimii de comenzi C) sunt inspirate din limbajul LOGO (grafica *turtle*). Iată o parte dintre ele:

F (*forward*) desenează trasând o linie înainte

G (*go*) deplasează broasca, fără desenare

+ (*increment*) rotește broasca cu unghiul u ales, în sens trigonometric

- (*decrement*) rotește broasca cu unghiul u ales, în sens anti-trigonometric

| (*turn*) încearcă să se întoarcă cu 180 de grade

! (*reverse*) inversează sensurile comenzilor **+** și **-**

\n incrementează unghiul cu n grade

/n decrementează unghiul cu n grade

Cn schimbă culoarea la culoarea n

<n incrementează culoarea cu n

>n decrementează culoarea cu n

@n multiplică lungimea segmentului de linie cu n

Starea broaștei este definită de tripletul (x, y, \mathbf{a}) , unde x și y reprezintă poziția curentă a broaștei, iar unghiul \mathbf{a} este interpretat ca direcție de translare a broaștei. Pentru generarea de forme grafice 3D, starea broaștei este dată de 6-uplul $(x, y, z, \mathbf{a}, \mathbf{b}, \mathbf{d})$, în care (x, y, z) sunt coordonate în spațiul cartezian tridimensional, iar unghiurile \mathbf{a} , \mathbf{b} și \mathbf{d} modelează orientarea curentă a broaștei.

2.4.1 Un exemplu

Considerăm următorul exemplu clasic (simbolul \rightarrow din cadrul regulilor îl vom înlocui cu $=$, iar comenzile

Angle și *Axiom* vor preciza unghiul inițial u și, respectiv, axioma):

```
Dragon {
  ; numele L-sistemului
  Angle 8 ; unghiul initial
           ; (360/8 = 45)
  Axiom Fx ; axioma, dupa care
           ; urmeaza regulile
  F=      ; se sterge F din sir
  y=+Fx--Fy+
  x=-Fx++Fy-
}
```

Șirul final, generat în urma aplicării regulilor, poate conține neterminali care vor fi eliminați în cadrul procesului de desenare. Pentru exemplul de mai sus, 6-uplul este $Dragon = \langle C \cup \{x, y\}, Fx, \{F=, y=+Fx--Fy+, x=-Fx++Fy-\}, 5, 8, 0 \rangle$ în care am considerat $order = 5$ și $colour = 0$ (negru). Înlăturând comenzile de desenare, L-sistemul constă din regulile de producție următoare: $a \rightarrow ab$ și $b \rightarrow ba$, având drept axiomă simbolul $a \in V$, cu $V = \{a, b\}$. Benoît Mandelbrot numește acest tip de L-sistem *dragonul Harter-Heighway*.

În urma aplicării regulilor de desenare, pentru exemplul de mai sus se va genera sistemul Lindenmayer din figura 1. Pentru un număr mare de iterații, reprezentarea grafică (denumită și *curba Dragon*) este ilustrată în figura 2. Pentru $order$ tinzând la infinit, reprezentarea L-sistemului se numește *attractor*. Mai mult, din figura 2 se poate remarca deja proprietatea de *auto-similaritate* a sistemului *Dragon*. Astfel, pentru majoritatea L-sistemelor se pot asocia două numere, unul desemnând numărul de forme „congruente” în care este divizată reprezentarea în întregul ei și celălalt definind un „factor de scalare” a fiecărei forme din cadrul reprezentării.

Datorită faptului că multe forme biologice au un aspect „fragmentat” sau „rămuros” în dezvoltarea lor, în grafica *turtle* se vor defini două noi comenzi *push* și *pop* care vor pune, respectiv vor extrage, de pe stivă starea unui L-sistem (clasa BDOL-sistemelor). Comanda *push* va fi reprezentată de simbolul [, iar *pop* va fi reprezentată de]. Pentru sistemele Lindenmayer tridimensionale, operațiunile *increment* și *decrement* vor fi înlocuite de șase comenzi de rotație 3D, câte o pereche pentru fiecare din cele trei axe Ox , Oy și respectiv Oz .

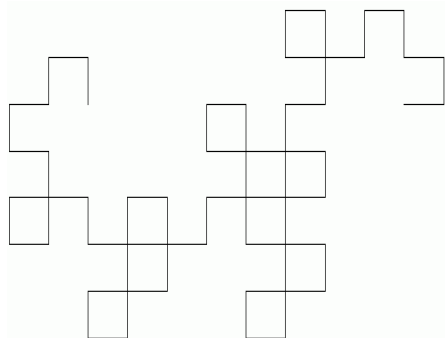


Figura 1: L-sistemul *Dragon* de ordinul 5

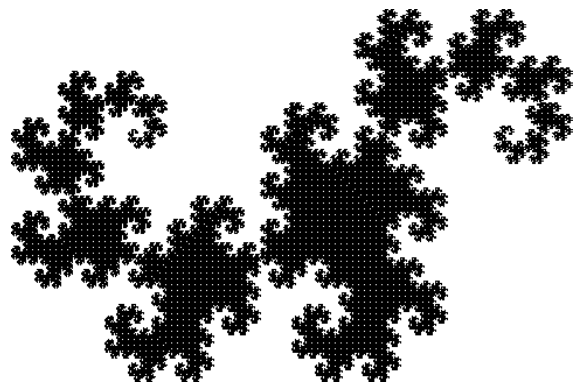


Figura 2: Curba *Dragon*

2.5 Reprezentarea grafică a sisteme-lor Lindenmayer

Există diverse aplicații destinate reprezentării grafice a L-sistemelor, fiecare având, din păcate, propriul format de stocare a datelor.

Unul dintre cele mai performante programe este *Lparser*, conceput de Laurens J.Lapre, care suportă definirea și reprezentarea L-sistemelor tridimensionale, formele grafice putând fi stocate în formate ca VRML, DXF sau POV. Drept componente suplimentare, bazate pe *Lparser*, se pot enumera *LMUSE* care generează, conform regulilor de rescriere, fișiere audio MIDI (*Music Instrument Digital Interface*) și *LPEvol* (*Lparser Evolution Engine*) – shell Java pentru analiza și reprezentarea grafică a L-sistemelor.

Ca alte exemple putem menționa: *Graptal* (sub UNIX) pentru simularea DOL-sistemelor parametrice, *Plant and Fractal Generator 2D* (pentru Macintosh), *FRACTINT* (sub DOS și Windows), *L-systems Applet* (pentru Web), toate fiind disponibile gratuit.



Figura 3: *Airhorse* (un L-sistem vizualizat într-un mediu VRML, parte din proiectul *Nerve Garden*)

Un proiect spectaculos al organizației Biota este *Nerve Garden*, bazat pe L-sisteme, VRML, rețele neuronale, tehnologii Web și tehnici de codificare genetică, al cărui scop este explorarea interactivă a biologiei digitale pe Internet și, într-un viitor mai depărtat, instituirea unei infrastructuri pentru medii virtuale biologice și crearea unor noi organisme (plante și animale) artificiale.

3. L-SYSTEMS MODELING LANGUAGE (LSML)

3.1 Filosofia XML

În continuare vom defini tipul de document LSML conform standardului XML, un tip de document fiind definit formal de părțile sale constituente și de structura acestora, prin intermediul *definiției tipului de document (Definition Type Document - DTD)*. DTD este exprimată ca un set de afirmații (instrucțiuni) declarative, utilizând o sintaxă specială. Fiecare definiție de element (marcator sau *tag*) al limbajului este compusă din trei părți: un nume sau un grup de nume (neterminali în jargonul limbajelor formale), două caractere specificând *regulile de minimizare* (vezi mai jos) și un conținut (neterminali și terminali), fiecare definiție putând fi asimilată cu o regulă de producție. Părțile componente ale unei declarații sunt separate de spații albe.

Regulile de minimizare determină dacă *tag*-urile de început și de sfârșit de element trebuie să fie prezente la fiecare apariție a respectivului element. Regulile sunt de fapt perechi de caractere: primul se referă la *tag*-ul de start, iar cel secund la *tag*-ul de sfârșit. Pot fi prezente numai caracterele "-" (marcator obligatoriu) și "o" (marcator opțional). În XML nu sunt permise *tag*-uri de sfârșit opționale, un element fără *tag* de sfârșit având o sintaxă modificată.

Închisă între paranteze, a treia parte a definiției unei element se numește *model-conținut*, specificând ce alte elemente pot apare în cadrul elementului definit. Cuvântul cheie #PCDATA (parsed character data) indică posibilitatea apariției oricărui caracter valid. Alte tipuri de date permise: CDATA (character data), NDATA (non-XML data), EMPTY (no data - element vid), ANY (orice tip de dată) etc.

În declarația unui model-conținut putem include un semn special, denumit *indicator de apariție*, care poate fi: semnul plus (elementul respectiv poate apare de una sau mai multe ori), semnul asterisc (elementul poate apare de zero, unu sau mai multe ori) sau semnul întrebării (elementul poate apare cel mult o dată).

Un model-conținut poate cuprinde mai multe componente într-o anumită ordine dictată de un *conector*. Un astfel de conector este unul din următoarele caractere: virgula (componentele trebuie să apară obligatoriu în ordinea specificată), ampersand (componentele trebuie să apară toate, dar în ordine aleatoare) ori bara verticală (doar una dintre componente poate apare).

Un anumit element poate poseda un set de *attribute* utilizate să descrie o anumită proprietate a unei apariții specifice (particulare) a acelu element. Atributele vor fi specificate numai în cadrul *tag*-ului de start, după numele elementului, ele declarându-se în cadrul DTD, tipurile valorilor acceptate fiind: ID (identificator unic pentru fiecare instanță a elementului), CDATA (orice șir de caractere valide), ENTITY (nume de entitate generală), NUMBER (doar numerele întregi sunt acceptate ca valori) etc.

A treia parte a definiției unui atribut specifică modul de interpretare a absenței atributului respectiv. Pot fi utilizate și aici o serie de cuvinte cheie predefinite: #REQUIRED (valoarea trebuie specificată în mod obligatoriu), #IMPLIED (valoarea poate lipsi) și altele.

Limbajul XML posedă o metodă flexibilă și facilă de codificare și referențiere a diferitelor părți ale conținutului unui document, într-un mod portabil, prin intermediul *entităților* (un fel de *macro*-uri). O formă specială de entități, numite *entități parametru*, pot fi utilizate în declarațiile XML din cadrul DTD, după cum vom vedea mai jos.

3.2 Definirea tipului de document pentru LSML

Conform definiției formale a unui sistem Lindenmayer, vor trebui specificate: numele L-sistemului, unghiul și culoarea inițiale folosite în reprezentarea grafică, axioma (compusă dintr-un șir de simboluri, neterminali sau comenzi de desenare), regulile de producție și numărul de aplicare în paralel a lor (gradul L-sistemului). Vom lua în considerație doar sistemele Lindenmayer deterministe și independente de context, având reprezentări grafice bidimensionale, deși modelul propus poate fi folosit și pentru alte tipuri de L-sisteme.

Vom defini pentru început elementele limbajului, apoi atributele aferente fiecărui element considerat.

```
<!ENTITY % command
"forward | left | right | colour |
reverse | turn | push | pop">
<!ELEMENT lsystem --
(axiom, rules)>
<!ELEMENT axiom --
((%command; | symbol)*)>
<!ELEMENT rules -- (rule*)>
<!ELEMENT rule --
((%command; | symbol)*)>
<!ELEMENT symbol --
(#PCDATA)>
<!ELEMENT forward -- EMPTY>
<!ELEMENT left -- EMPTY>
<!ELEMENT right -- EMPTY>
<!ELEMENT colour -- EMPTY>
<!ELEMENT reverse -- EMPTY>
<!ELEMENT turn -- EMPTY>
<!ELEMENT push -- EMPTY>
<!ELEMENT pop -- EMPTY>
```

După cum se remarcă din definițiile de mai sus, un document LSML va defini un L-sistem, fiind compus din elementele *axiom* și *rules*. Aceste elemente vor conține fie simboluri neterminale (date prin intermediul elementului *symbol*), fie comenzi de desenare (elemente de tip *command*). Elementul *left* substituie comanda *decrement*, iar elementul *right* înlocuiește *increment*. Comanda *go* se obține prin construcția *<forward*

draw="off">, iar comanda @ prin <forward length="..."> (vezi atributele de mai jos).

Pentru aceste elemente, atributele asociate sunt următoarele:

```
<!ATTLIST lsystem
  name      PCDATA  #REQUIRED
  author    PCDATA  #REQUIRED
  angle     PCDATA  #REQUIRED
  colour    PCDATA  #IMPLIED
>
<!ATTLIST rules
  repeat    NUMBER  #REQUIRED
>
<!ATTLIST rule
  start     PCDATA  #REQUIRED
>
<!ATTLIST forward
  draw      on | off  "on"
  length    NUMBER  #IMPLIED
>
<!ATTLIST left
  degree    NUMBER  #IMPLIED
  axis      PCDATA  #IMPLIED
>
<!ATTLIST right
  degree    NUMBER  #IMPLIED
  axis      PCDATA  #IMPLIED
>
<!ATTLIST colour
  value     PCDATA  #IMPLIED
  inc       NUMBER  #IMPLIED
  dec       NUMBER  #IMPLIED
>
<!ATTLIST push
  stack     PCDATA  #IMPLIED
>
<!ATTLIST pop
  stack     PCDATA  #IMPLIED
>
```

Atributele asociate elementelor LSML permit specificarea unui set mai larg de L-sisteme (am inclus atributul *axis* pentru specificarea axei pentru care se modifică unghiul broaștei, util pentru sisteme 3D, iar comenzile *push* și *pop* permit salvarea în și respectiv extragerea de informații din mai multe stive prin intermediul atributului *stack*, permițându-se astfel definirea de sisteme Lindenmayer cu stive multiple). În cadrul unui element *rule*, atributul *start* nu trebuie neapărat să conțină numai un neterminat, dându-se în acest mod posibilitatea specificării de L-sisteme dependente de context.

3.3 Exemplu

Vom rescrie definiția L-sistemului *Dragon* din secțiunea 2.4 ca document LSML:

```
<?xml version="1.0">
<!DOCTYPE lsystem PUBLIC "-//LSML
1.0//EN">
<lsystem name="dragon" author="Sabin
Corneliu Buraga" angle="8">
  <axiom>
    <forward />
    <symbol>x</symbol>
  </axiom>
  <rules repeat="5">
    <rule start="y">
      <right />
      <forward />
      <symbol>x</symbol>
      <left />
      <left />
      <forward />
      <symbol>y</symbol>
      <right />
    </rule>
    <rule start="x">
      <left />
      <forward />
      <symbol>x</symbol>
      <right />
      <right />
      <forward />
      <symbol>y</symbol>
      <left />
    </rule>
  </rules>
</lsystem>
```

Se putea renunța la elementul *symbol*, însă l-am preferat din motive de claritate a documentului-sursă.

3.4 Modelul orientat-obiect pentru documente LSML

Modelul orientat-obiect pentru documente (Document Object Model - DOM) reprezintă o interfață de programare a aplicațiilor destinate să prelucreze documentele XML și HTML, independentă de platformă și de limbaj, definind structura logică a documentelor și modalitățile de accesare și de modificare a lor. Această structură logică este una arborescentă, orientată-obiect: documentele sunt modelate utilizând obiecte, iar modelul nu oferă doar o vizualizare structurată a documentului, ci și o manieră de specificare a comportamentului său și a obiectelor componente. Fiecare element al unui document poate fi privit, deci, ca un obiect, fiecare obiect având identitate și propriile sale funcții. DOM identifică interfețele și obiectele utilizate să reprezinte și să manipuleze un document, semantica acestor interfețe și obiecte (inclusiv comportamentul și atributele lor) și relațiile și dependențele între aceste interfețe și obiecte.

LSML fiind derivat din meta-limbajul XML, pentru scrierea aplicațiilor poate fi folosit DOM deja specificat de *Consortiul Web* pentru XML, fiind standardizat doar primul nivel, urmând a fi supus

standardizării și nivelul secund al modelului. În plus, ne propunem ca în viitor să definim un model obiectual specific documentelor LSML.

3.5 Avantaje

Principalul avantaj al reprezentării L-sistemelor ca documente LSML este independența de platforma hardware și software, acest format putând fi folosit ca modalitate de interschimbare a definițiilor de sisteme Lindenmayer între diverse programe de modelare a lor.

Documentele LSML sunt mai lizibile și mai ușor de prelucrat decât fișierele de date (nestructurate) utilizate de diverse programe de reprezentare a sistemelor Lindenmayer ca *Lparser* sau *L-systems Applet*.

Fiind derivat din XML, limbajul LSML poate fi integrat în alte limbaje de adnotare, ca de exemplu în XHTML, succesor al HTML, în WebSchematics, utilizat în efectuarea de scheme și de ilustrații pe Web, în VML (*Vector Markup Language*) sau SVG (*Scalable Vector Graphics*), pentru reprezentarea imaginilor vectoriale.

Modelul obiectual pentru LSML oferă suport în dezvoltarea de aplicații de manipulare a documentelor, în limbaje precum Java sau C++.

Reprezentarea L-sistemelor în LSML oferă o mai mare ușurință de procesare în paralel a regulilor de producție, deoarece structura arborescentă a documentelor permite utilizarea unor tehnici de accesare și de căutare paralele. Aceasta poate duce la accelerarea reprezentării grafice a sistemelor Lindenmayer.

LSML poate fi integrat în proiectul *Nerve Garden* ca mijloc universal de reprezentare a formelor biologice descrise de L-sisteme în medii virtuale distribuite, L-sistemele 3D stocate ca documente LSML putând fi convertite ulterior în obiecte VRML.

4. CONCLUZII

În prezenta lucrare am definit un limbaj derivat din XML pentru reprezentarea elegantă, independentă de hardware și de software, a sistemelor Lindenmayer, având multiple utilizări în domenii ca biologia, simularea computerizată sau grafica de sinteză.

Limbajul specificat, *L-System Modeling Language*, poate fi folosit pentru interschimbarea reprezentărilor L-sistemelor în diverse programe de vizualizare sau pentru integrarea fractalilor L-sistem în alte documente XML ori în paginile Web. De asemenea, operațiile de căutare în astfel de structuri pot fi efectuate în paralel. Ne propunem ca pe viitor să oferim și un model orientat-obiect pentru documentele LSML în vederea conceperii de aplicații Web de prelucrare distribuită a L-sistemelor, să studiem posibilitățile de integrare completă în limbaje de specificare a graficii pe calculator precum *Web Schematics* sau VML și să realizăm o aplicație de vizualizare și prelucrare a documentelor LSML.

5. BIBLIOGRAFIE

- [1] T.Bray, J.Paoli, C.M.Sperberg-McQueen (eds.) – „Extensible Markup Language (XML)”, W3C, nov.1997: <http://www.w3.org/TR/REC-xml>
- [2] B.F.Damer – „Amoeba: a Simulator for Molecular Nanotechnology”, The Fourth Foresight Conference on Molecular Nanotechnology, Palo Alto, 1995
- [3] R.M.Dickau – „Three-dimensional L-Systems”: <http://forum.swarthmore.edu/advanced/robertd/lsys3d.html>
- [4] D.Duce, B.Hopgood – „Web Schematics on WWW”, Web Consortium’s Note, mart.1998: <http://www.w3.org/TR/1998/NOTE-WebSchematics>
- [5] M.Hammel – „L-Systems Software”: <http://www.cpsc.ucalgary.ca/~hammel/BioSim/Lsystems/software/software.html>
- [6] T.Jucan – „Limbaje formale si automate”, Ed. MatrixRom, Bucuresti, 1999
- [7] L.Lapre – „Lparser”: <http://www.xs4all.nl/~ljlapre>
- [8] T.Lin – „Animation of L-system based 3-D Plant Growing in Java”, Maryland Univ., nov.1996: <http://www.cs.umbc.edu/~ebert/693/TLin/>
- [9] B.Mandelbrot – „The fractal geometry of nature”, W.H. Freeman, San Francisco, 1982
- [10] B.Mandelbrot – „Obiectele fractale”, Ed.Nemira, Bucuresti, 1998
- [11] A.Mariano, P.Moscato, M.Norman – „Using L-Systems to generate arbitrarily large instances of the Euclidean Traveling Salesman Problem with known optimal tours”, Anales del XXVII Simposio Brasileiro de Pesquisa Operacional, Victoria, Brazil, nov.1995: <ftp://ftp.ing.unlp.edu.ar/pub/papers/memetic/final.ps.Z>
- [12] M.Masse – „L-system Applet”: <http://ugrad-www.cs.colorado.edu/masse/Java/Lsystem/Lsys.html>
- [13] S.Papert – „Mindstorms: Children, Computers, and Powerful Ideas”, Basic Books, New York, 1980
- [14] P.Prusinkiewicz, A. Lindenmayer – „The Algorithmic Beauty of Plants”, Springer-Verlag, New York, 1990
- [15] P.Prusinkiewicz, L.Kavi – „Subapical bracketed L-systems”, in Grammars and Their Application to Computer Science, LCNS 1073, Springer-Verlag, Berlin, 1996
- [16] P.Prusinkiewicz, M.Hammel – „Language-Restricted Iterated Function Systems, Koch Constructions, and L-systems”, in New Directions for Fractal Modeling in Computer Graphics, SIGGRAPH’94 Course Notes, ACM Press, 1994

- [17] G.Rozenberg, A.Salomaa – „Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology”, Springer-Verlag, Berlin, 1992
- [18] D.Skillicorn – „Structured Parallel Computation in Structured Documents”, Journal of Universal Computer Science, vol.3, no.1, 1997
- [19] M.Taylor, J.P.Louvet – „sci.fractals Frequently Asked Questions”, 1998: <http://www.mta.ca/~mctaylor/sci.fractals-faq/>
- [20] J.Vaario, K.Shimohara – „Modeling Environment Sensitive L-systems”, Kyoto Conference on Mathematical Biology'96, Japon, 1996
- [21] L.Wood (ed.) – „Document Object Model (DOM) Level 1 Specification”, Web Consortium, oct.1998: <http://www.w3.org/TR/REC-DOM-Level-1>
- [22] D.Wright – „Dynamical Systems and Fractals”: <http://www.math.okstate.edu/mathdept/dynamics/lecnotes/>
- [23] * * * - „FRACTINT Web Pages”: <http://spanky.triumf.ca/www/fractint/>
- [24] * * * - „Live Alife Page”: <http://www.fusebox.com/cb/alife.html>
- [25] * * * - „VRML Resources”: <http://www.vrml.com>
- [26] * * * - „World-Wide Web Consortium's Technical Reports”, Boston, 2000: <http://www.w3.org/TR/>