SPECIAL SECTION

Non-Standard Sound Synthesis with L-Systems

Stelios Manousakis

ABSTRACT

This paper presents a new non-standard technique for waveform synthesis in the time domain using L-systems, a formalism related to grammars, fractals and automata. This technique, developed as part of a larger-scale compositional system, is based on waveform segmentation and offers various methods for generating wavetables. The paper first introduces L-systems and some specifics of their interpretation and discusses extensions such as incorporating genetic algorithms and designing hierarchical L-systems and L-system networks. The second half describes the implementation model in detail, proposes some sound synthesis strategies and presents paths for further work.

Non-standard sound synthesis is a term introduced by Holtzman [1] to refer to specific types of abstract sound synthesis that describe sound as amplitude and time values. It is an experimental field of synthesis, idiomatic to the digital domain. Non-standard techniques are based on mathematical models and compositional abstraction rather than the human ear (as in spectral synthesis), physical properties of objects (physical modeling) or reproduction of actual sound sources (sampling-based synthesis).

Non-standard models approach sound synthesis as a type of micro-composition, usually merging the worlds of sound synthesis and score creation in one system as a strategy for achieving coherence between form, structure and material. With such techniques, many aspects of score composition are reduced to sound synthesis, with rules applied in the lowest levels leading to larger-scale morphogeneses.

There are a number of non-standard techniques that can be roughly categorized into: rule-based systems, such as Koenig's *SSP* [2], Berg's *ASP* and *PILE* [3], Brün's *SAWDUST* [4] and Holtzman's digital instrument [5]; stochastic systems, such as Xenakis's *Stochastic Synthesis* and *GENDY* [6–7]; other waveform segmentation algorithms, such as *EWSS* by Valsamakis and Miranda [8], Chandra's *Wigout* and *Triktrak* [9] and the "fractal interpolation" techniques by Yadegari [10], Monro [11] and Dashow [12]; and lastly, non-linear oscillators, such as Collins's *SLUGens* [13].

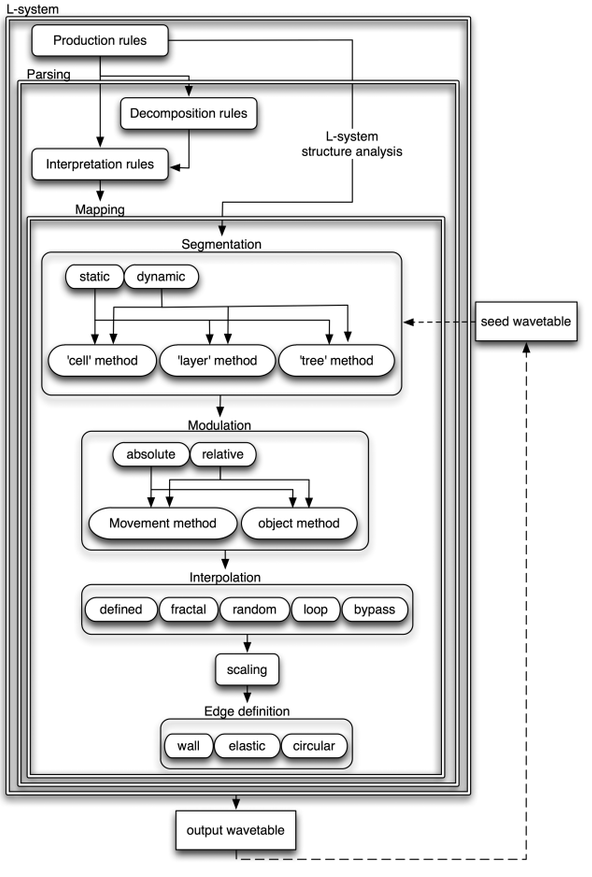
The sound synthesis model described in this paper is part of a larger compositional system first presented in 2006 [14], hereafter referred to as Musical L-systems, which aims to create a rigorous and comprehensive musical method that unifies compositional procedure in all time-levels through the use of L-systems, an algorithm modeling biological growth.

L-SYSTEMS

Lindenmayer systems, or L-systems, are named after Aristid Lindenmayer, who introduced them in 1968. They are related to formal grammars, iterative systems, cybernetics and automata theory. An L-system is a form of string rewriting grammar that works with symbols. Rewriting is a widely used technique for transforming a given input according to a set of rules or productions. This can happen recursively with a feedback loop. Complex objects can be defined by successively replacing parts of a simple initial object, which makes rewriting a very compact and powerful technique. Due to the recursive nature of the algorithm, local behaviors designed at a particular level result in forms and structures emerging in higher levels. L-systems were originally conceived for modeling cell development and plant growth but are increasingly used in various other fields (e.g. computer graphics, architecture, robotics), with very convincing results. A few musical implementations have been proposed for generating MIDI scores.

An L-system is a generative grammar that consists of an alphabet, an optional set of constants,

Fig. 1. L-system sound synthesis flowchart. (© Stelios Manousakis)



Stelios Manousakis (composer, researcher), Center for Digital Arts and Experimental Media (DXARTS), Box 353414, University of Washington, Seattle, WA 98195-3414, U.S.A. E-mail: <stm@modularbrains.net>. Web site: <www.modularbrains.net>.

Sound samples related to this article are available at <modularbrains.net/DoDigitalMonkeysInhabitVirtualTrees.html> and <modularbrains.net/UndercoverHarpsichordAgentsTerrorizeTheCourt.html>.

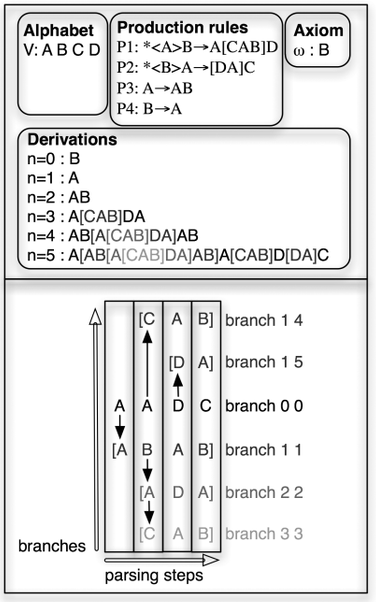


Fig. 2. In bracketed L-systems every branch is tagged with its distance in number of branches from the trunk and its index in the tree. (© Stelios Manousakis)

an axiom (i.e. a string of symbols from the alphabet defining the system's initial state) and a set of rules for iteratively replacing symbols with other symbols. A complete L-system contains three or four types of rules, describing separate levels of data process: the production, (optional) decomposition, interpretation and mapping rules.

For a flowchart of the implementation presented here, see Fig. 1.

Production Rules

The production rules form the transformational core of the algorithm, representing structural development at the most abstract level. L-system productions are specified using the standard notation of formal language theory. Different types of production rules exist, defining the type, character and output of the L-system, with different rules generating different formal languages. An L-system can be:

- Context-free (OL-system) or context-sensitive (IL-system)
- Deterministic (DL-system) or stochastic
- Bracketed
- Propagative (PL-system) or non-propagative
- Inclusive of tables (TL-system)
- Parametric
- Inclusive of extensions (EL-system). For example, multi-set, environmentally sensitive, open

Nonexclusive types of production rules can be combined in an L-system. For instance, a simple example of a deterministic, context-free system (DOL-system):

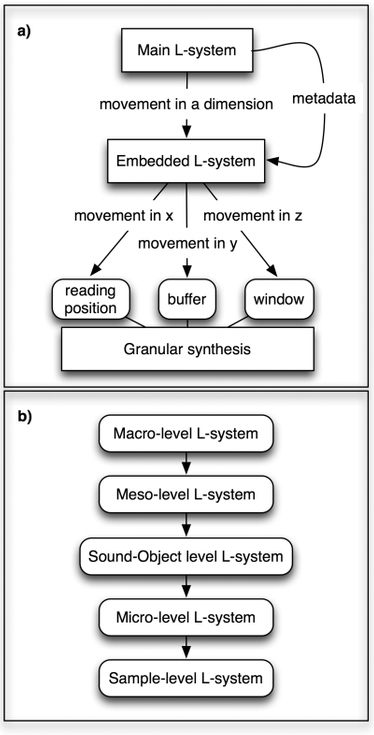
Alphabet:
V: A B
Production rules:
P1: A → AB
P2: B → A
Axiom:
ω: B
will produce for derivation step *n*:
n = 0: B
n = 1: A
n = 2: AB
n = 3: ABA
n = 4: ABAAB
n = 5: ABAABABA.

For a more extensive presentation of the various types of L-systems see Manousakis [15] and Prusinkiewicz and Lindenmayer [16].

Decomposition Rules

The fundamental concept of the algorithm is that of development over time, with symbols representing simple structural modules. However, compound modules consisting of several elements can also be defined, then decomposed into their constituents using decomposition rules. These are context-free rules, applied after each transformation of the string resulting from the production rules.

Fig. 3. Hierarchical L-systems modeling: a) the timbre of a granular engine, b) time-level hierarchies. (© Stelios Manousakis)



Interpretation Rules

The interpretation rules are the part of the algorithm that parses and translates the string output. Without them, an L-system remains a theoretical abstraction of structural transformation. The interpretation rules have the form of context-free productions and are applied recursively after each derivation step, following the production and decomposition rules.

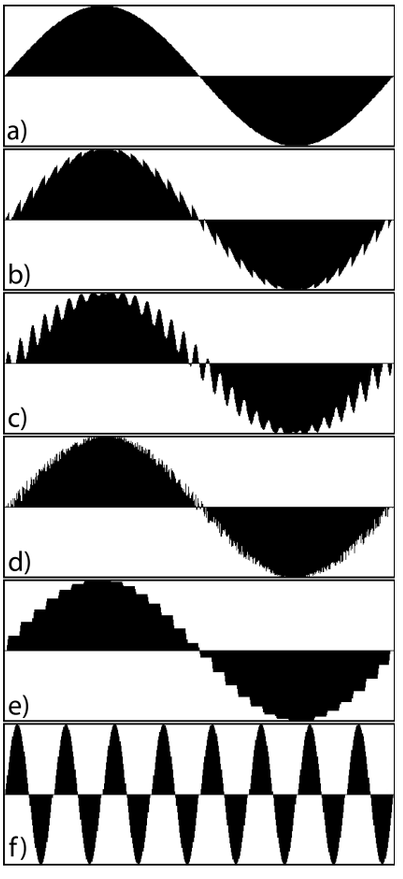
Mapping Rules

These rules determine how to use the interpreted output of the algorithm, mapping the product data to the actual application. They are specific to each implementation and may differ greatly between disciplines (e.g. graphics, music) or depending on the results desired.

AUTOMATA

L-systems were first used as an abstract mathematical tool with very simple graphical interpretations. Later, a LOGO-style "turtle graphics" interpretation was incorporated. The turtle, in its simplest form, is a two-dimensional au-

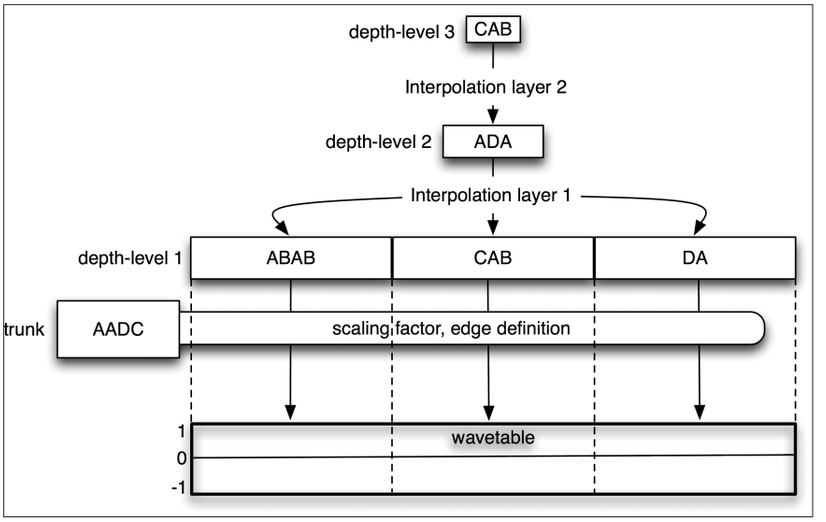
Fig. 4. Interpolation types: (a) Defined: linear; (b) defined: exponential; (c) fractal; (d) random; (e) None: bypass; (f) None: loop. (© Stelios Manousakis)



tomaton whose state is defined as the set {*x, y,* α}, where (*x, y*) are the Cartesian coordinates representing the turtle's position in space and (α) is the angle representing its heading, that is, which way it faces. The L-system string is interpreted as a sequence of commands moving the automaton in space.

In Musical L-systems, the "turtle" is an automaton that moves and rotates in time, in an infinite floating-point space of up to three dimensions. It can have an additional predefined number of dynamic qualities represented by autonomous noninteracting dimensions that can be finite or infinite, integer or floating-point. It can also perform various other tasks encoded as metadata. All dimensions—including rotations and qualities—and metadata may be mapped to musical parameters.

The symbols represent actions. When the product string is parsed, each symbol commands the automaton to perform a specific task. A symbol may command it to move or rotate in {*x, y, z*}, to change a quality, to scale the output of a dimension or offset itself in it, to start a new data stream (branch) or end the current one, to change a global variable of the L-system—or to act upon any parameter of the compositional environment. In order to delineate an action, the following must be set: the dimension, value and parameter upon which it will be applied; the algebraic function to use when applying the action (add/subtract/multiply/divide); the value/variable to apply.



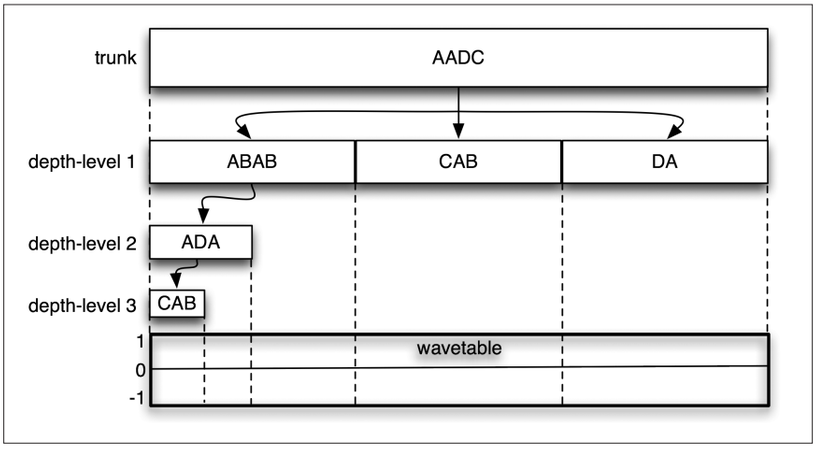


Fig. 5. (a) "Layer" segmentation using the bracketed L-system from Fig. 2. (b) "Tree" segmentation using the bracketed L-system from Fig. 2. (© Stelios Manousakis)

Seeding

Seeding is a very important aspect for the interpretation of an L-system. The same grammar gives different results with different seeds. Prior to parsing, a set of initialization values is defined, each being the seed for a dimension. The automaton is initialized in the state dictated by these values.

Thresholds

A low and/or high threshold can be set within a dimension to alter the interpretation of a symbol if the current state of the automaton surpasses that threshold. The change may involve using a different value or variable or even a different function. This gives control over the range of movement of the automaton from within the rules, limiting the space without compression.

Parsing: Static versus Dynamic Interpretations

During parsing, the string generated by an L-system is interpreted in sequence from beginning to end, with symbols following each other as instructions to the automaton for generating musical data. In Musical L-systems, there are two options for timing the parsing procedure for sample-level synthesis, depending on whether the focus is on the result or the process:

1. Static interpretation in sample-and-hold time: The interpreted outcome of the entire production is output at once after each derivation (similarly to graphic interpretations). In this case the goal is to create one static sound-object for each generation.
2. Dynamic interpretation in continuous time: The string is treated as a continuous growth process, with symbols parsed sequentially in real time. Symbols can be interpreted as "active time-cells" triggering a parsing step, or as hidden accumulative commands. This is a much more powerful and musically appropriate interpretation, first introduced in Prusinkiewicz [17]. In Musical L-systems, the temporal flow of parsing can be controlled by the automaton itself, either by using metadata actions to control an external clock, or by incorporating an internal feedback clock, with the time delay between parsing steps represented by an internal time vector within the automaton. Integrating the control of the timing of parsing in the automaton itself enforces the grammatical unity between structure and development over time, allowing for spatiotemporal patterns to emerge.

Multiple Automata in Musical L-Systems

In Musical L-systems, the output of certain types of L-systems can be used to generate multiple automata. Their amount can be predefined or set by the L-system.

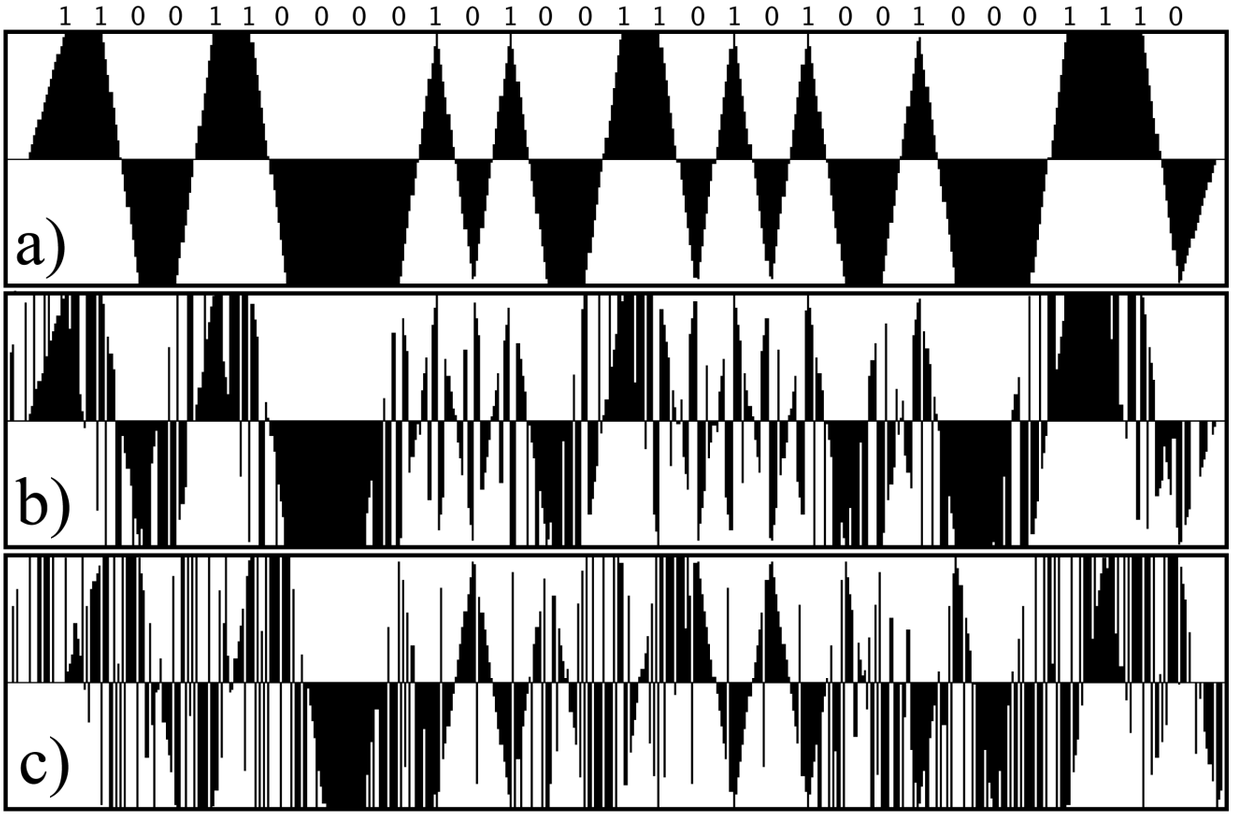


Fig. 6. "Cell" segmentation and movement modulation using a non-propagative L-system (512 samples); interpolation is: (a) linear, (b) defined by another L-system, (c) fractal. (© Stelios Manousakis)

Bracketed L-systems

Bracketed (branched) L-systems are parallel processing models that generate multiple branches, each representing a different data stream. One or more sets of special symbols are used to start and end branches ("push"/"pop").

Branching Structure. Before a bracketed L-system is parsed, a data map is made for the current generation carrying information about the tree, the branching structure and each branch. This information is used to tag every branch (Fig. 2) with:

- The branch index, showing how many branches have been generated in the string before that branch.
- The depth-level (order), showing how many branches away from the trunk that branch is located.

Inheritance. A branch can inherit some characteristics (states) from its generating branch. This means it will be seeded by the values inherited in the respective dimensions. Inheritance is a property transmitted by the branching symbol starting the new data stream ("push" command). Different sets of branching symbols can be used within one grammar, permitting simultaneous use of various inheritance types. This allows for numerous possible types of dependencies and attractions between branches, with stronger or weaker connections and the inheritance fields being programmable through the rules.

Mapping branches to automata in Musical L-systems. Branching allows for contrapuntal polyphonic data structures to emerge in space-time. Branches can be mapped to automata in several ways, which either preserve the original branching structure or group branches into sets of output automata. These options are:

1. Treat every branch as an output automaton.
2. Group all branches of the same depth-level into one output automaton.
3. Define a number of output automata and map the branches to them in groups. This happens:
   (a) With overlaps, by assigning each new branch to a currently inactive automaton, or
   (b) Stochastically, using a probability table to calculate the density of data streaming for each output automaton.

The third option is by far the most versatile. With overlapping, minor data distortion occurs—depending on the ratio of branches to output automata. Stochastic mapping can be used as a pattern distribution technique with a defined relative density. The total number of real branches is stochastically grouped into output automata according to the respective probability weights assigned to each output. Several branches can thus be interwoven into a single automaton, forming complex combined patterns.

Non-Propagative L-systems

This is another family of L-systems that can be used to generate multiple automata acting simultaneously. In non-propagative L-systems, there is no data amplification (hence the name), meaning that the length of the output string remains fixed. Non-propagative L-systems can produce the same results with standard cellular automata algorithms in *n*-dimensions [18]; therefore cellular automata terminology is used to describe them. The string represents

a collection of interacting cells (cell grid), where each cell is a data stream, with symbols describing their states. As such, the parsing method for these L-systems requires all symbols in a derivation to be parsed, interpreted and output simultaneously.

Cells can be mapped to automata in the following ways:

1. Every cell is an output automaton.
2. Every cell is an output automaton, its control value being the average of the cell's *n* last states.
3. The cell grid is split into regions corresponding to automata; the control value is the average of the states of the cells in a region.
4. The cell grid is split into regions corresponding to automata; the control value is the average of the states of the cells in a region for the *n* last derivations.

Averaging can be used as a technique for investigating local instead of individual behaviors, the locus being an area in the string (cell neighborhood), a period in time (cell history) or both (neighborhood history).

Extensions

The L-system model can be extended to include higher-level control mechanisms for altering rules at any level. Some useful mechanisms implemented in Musical L-systems for sample-level synthesis are:

Table L-systems (TL-systems)

TL-systems may be used to change the production and/or interpretation rules during a composition according to a predefined plan, thus making it possible to program developmental switches of the system in time. Different sets of rules sharing the same alphabet are saved as presets. A clock mechanism is then used to choose when to replace the current ruleset with another. This mechanism can be external, following a score, another process or with manual intervention—or it can be controlled internally from an L-system with metadata symbols.

Evolving Grammars with Genetic Algorithms

A non-deterministic extension of TL-systems is implemented using genetic algorithms. Genetic algorithms simulate natural evolution and are effective in solving the "inference" or inverse problem, that is in finding rules that produce desired structures. They can be used as a tool for designing L-system grammars prior to composing or to generate stochastic variations that direct the output structure toward a certain goal within a piece.

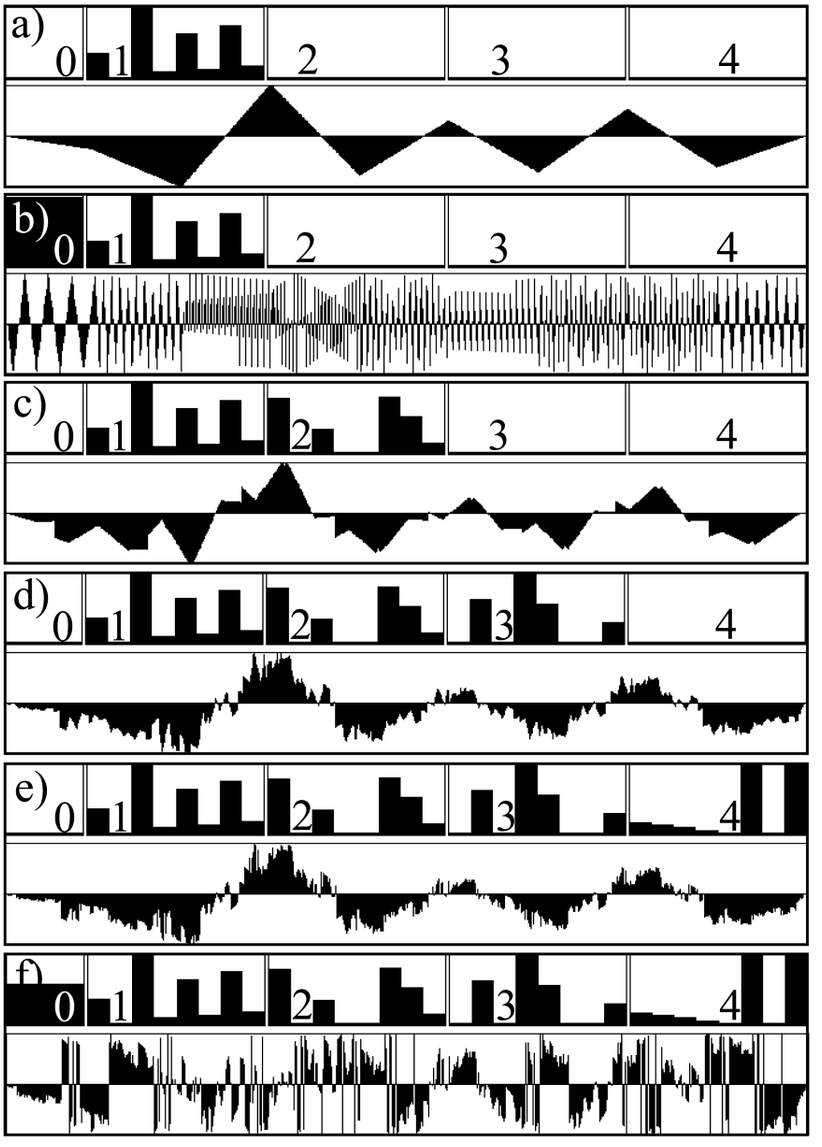
Hierarchical Systems and L-system Networks

The concept of decomposition rules can expand to include embedded grammars. An automaton may command a number of subordinate L-systems with their own grammar, particularly fitted for modeling a certain process (Fig. 3). A metadata action or trajectory can control any aspect of an L-system—axiom, rules, variables, parsing speed, branching structure, automata states or mapping. A coherent and unified control of musical structures in all time-levels can be achieved by relating different sets of L-systems, either hierarchically or in a network, using varying degrees of interactions and interdependencies.

Sound Synthesis Implementation in Musical L-Systems: Generating Wavetables

The synthesis method is twofold: segmentation and modulation. A seed waveform is divided into a number of wave fragments, whose time scale is near the auditory threshold—depending on the segment size and the sampling rate. A bracketed or non-propagative L-system generates a number of automata, each

Fig. 7. "Layer" segmentation and movement mapping: Six waveforms (1,024 samples) generated by a bracketed L-system with five depth-levels. (b) elastic barriers; (f) circular space. The other waveforms do not exceed the edges. (© Stelios Manousakis)



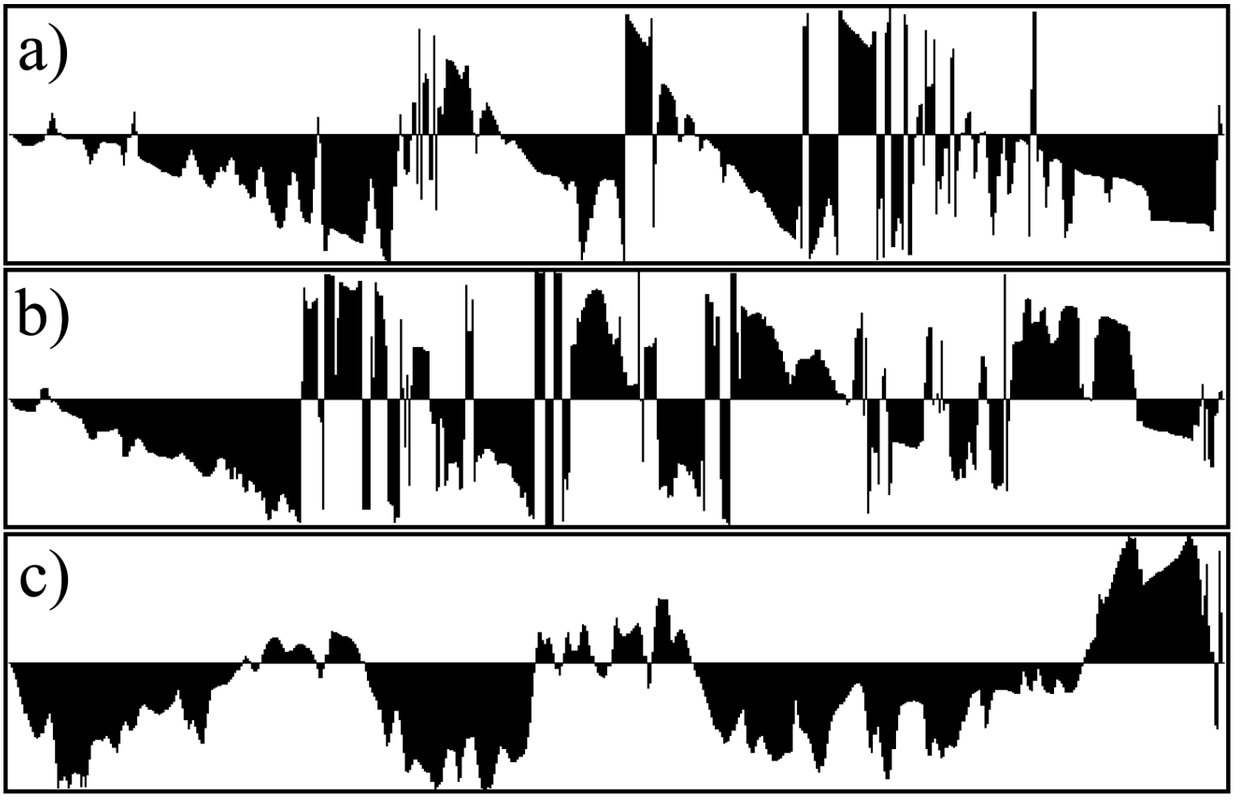


Fig. 8. Waveforms generated after one derivation of a deterministic L-system using "tree" segmentation and movement mapping over an empty seed (512 samples): (a) linear interpolation, circular space; (b) random interpolation, circular space; (c) linear interpolation, elastic barriers, branch movement quadrupled. (© Stelios Manousakis)

mapped to a waveform segment that it modulates according to the interpretation rules. With this technique a waveform can be synthesized on the sample level without having to specify each sample value separately.

Seeding

A table is seeded with a list or waveform produced by a function, by sampled audio or manually; this table can be linked to a parameter outside the L-system (e.g. belonging to the compositional environment, live input, etc.). Thereby the L-system can grab, modulate and transform a given structure. If no seed is provided, modulation starts from zero (i.e. a void state) and the L-system generates new data from an empty initial structure.

Interpolation

The technique for using a given number of input values (automata states) to modulate a larger number of table values (samples) is interpolation. There are five interpolation options (Fig. 4):

1. Defined interpolation: interpolate using a table (default is linear interpolation).
2. "Fractal" interpolation: the overall shape of the table is used as the interpolation shape between two input values.
3. Random interpolation; interpolate randomly.
4. Loop: no interpolation. The overall shape is sequentially repeated until the entire table is filled.
5. Bypass: no interpolation. Each input value is repeated a number of times in order to stretch the shape to the table size.

Segmentation

Segmentation can be static or dynamic, meaning that the wavetable size and the size of its segments may change in time, either due to a change in the L-system structure or through parameterization of the amount of segments and/or their lengths. The methods for segmenting a table are:

Non-Hierarchical Segmentation: "Cell" Method. Segmentation is parallel and non-overlapping. The waveform is divided into a number of segments, each modulated by an automaton. This can be:

1. A cell or cell group for non-propagative L-systems.
2. A branch or branch-group for bracketed L-systems.
3. An extension of this technique exists for non-bracketed propagative L-systems: In a similar manner to non-propagative L-systems, the entire production is parsed simultaneously, with each symbol in the string linked to a wave segment. The main difference is data amplification, with every derivation having more symbols, and hence segments, than the previous. There are two ways to deal with this growth: The wavetable maintains its size but the length of each segment is reduced with each generation; or the segment lengths remain unaltered, but the wavetable size grows.

Hierarchical Segmentation. It is possible to use the branching structure of a bracketed L-system to segment a wavetable hierarchically. In this case a sample value may be accessed by more than one automaton. There are two such methods:

"Layer" method. Layer-specific hierarchical aspects of bracketed L-systems are used as an extension of the concept of defined interpolation. The order of a branch, meaning how many branches it is away from the trunk, represents its hierarchical importance in a generation. Branches closer to the trunk are considered more important than others further away. Branches of the same

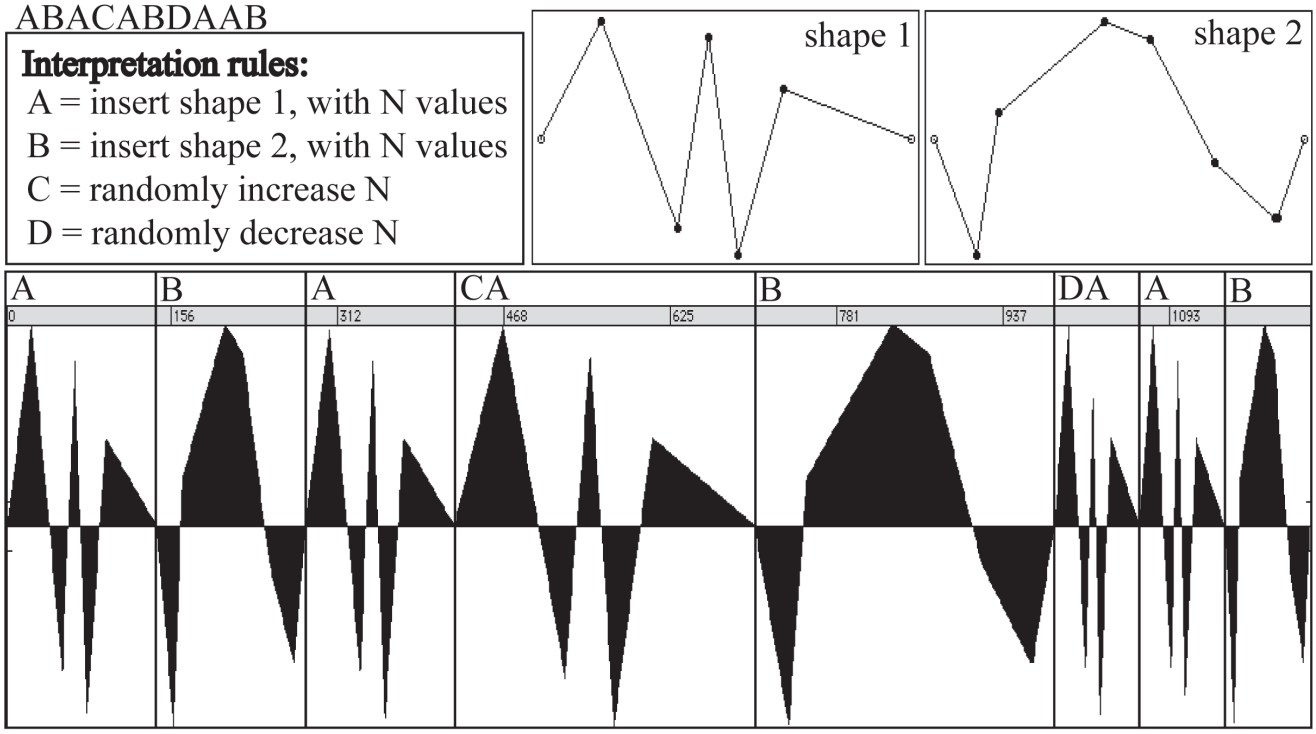


Fig. 9. "Object" mapping with a simple L-system using "cell" segmentation. (© Stelios Manousakis)

depth-level are grouped together, with 1st-order branches defining waveform segments and higher-order branches defining the interpolation layer that determines the shape between values of their superordinate depth-level. The trunk is used as a multiplier for determining the overall effect of branch movement in the table (Fig. 5a). Embedded L-systems can also be used to define interpolation layers.

"Tree" Method. A wavetable can be segmented using the hierarchical structure of the bracketed L-system that modulates it. Branch-specific hierarchies define the size and position of a segment in the table. Again, the depth-level of a branch determines its hierarchical importance, translated into the amount of sample values it controls. The higher in hierarchy, the bigger the segment (Fig. 5b). For the trunk, this is given by the equation:

$$TrunkLength[n] = \frac{TableLength[n]}{\left(\frac{TrunkParsingSteps}{GenerationParsingSteps}\right)}$$

That is, the effect of the trunk is scaled according to its relative sequential length—with the variable *TrunkParsingSteps* representing the amount of symbols treated as "active time-cells" in the trunk—as compared to the total sequential length of the generation (*GenerationParsingSteps*).

In every derivation, the wavetable is divided without overlaps by the amount of 1st-order branches. Subordinate branches subdivide the space of their parent branch, and so on, making a hierarchical division of the table and each segment. The same value in a table can, therefore, be modulated by a number of different depth-level branches in the same generation. The number of samples that a 1st-order branch modulates is considered to be the standard length-unit. The equation for determining this is:

$$LengthUnit[n] = \frac{TableLength[n]}{AmountOf1stOrderBranches}$$

For 2nd-order and higher sub-branches, the length equation takes into consideration the total amount of sub-branches of the same order that the parent of a branch contains (*ParentOffspring*):

$$BranchLength[n] = \frac{\left(\frac{TableLength[n]}{BranchOrder}\right)}{ParentOffspring}$$

The position of the area of activity of a branch (the offset in the table) is defined as:

$$BranchPosition[n] = (LengthUnit[n] * ParentLocalIndex) + (LocalIndex * BranchLength[n])$$

where *LocalIndex* represents how many branches of the same order the parent has generated before that branch, and *ParentLocalIndex* represents how many branches of the same order the grandparent generated before generating the parent.

Segment Length Parameterization

In all the above methods, segments of the same depth-level may have equal lengths, or their relative sizes may be adjusted individually via a scaling parameter. This parameter can be controlled from an external process, such as another L-system or a random function, or internally by each automaton as follows:

- For non-propagative L-systems, through the history, neighborhood average or neighborhood history of a cell. This way, segment lengths change once per derivation (which, for non-propagative L-systems, equals once per parsing step).
- For bracketed L-systems, through an assigned dimension in each automaton—and for grouped automata, through the average state in that dimension—or with metadata actions. In both cases, segment lengths may change continuously within a derivation. Another option is to calculate the amount of branches grouped per output automaton—in which case sizes change once per derivation. It is also possible to account for the amount both of branches and of states.

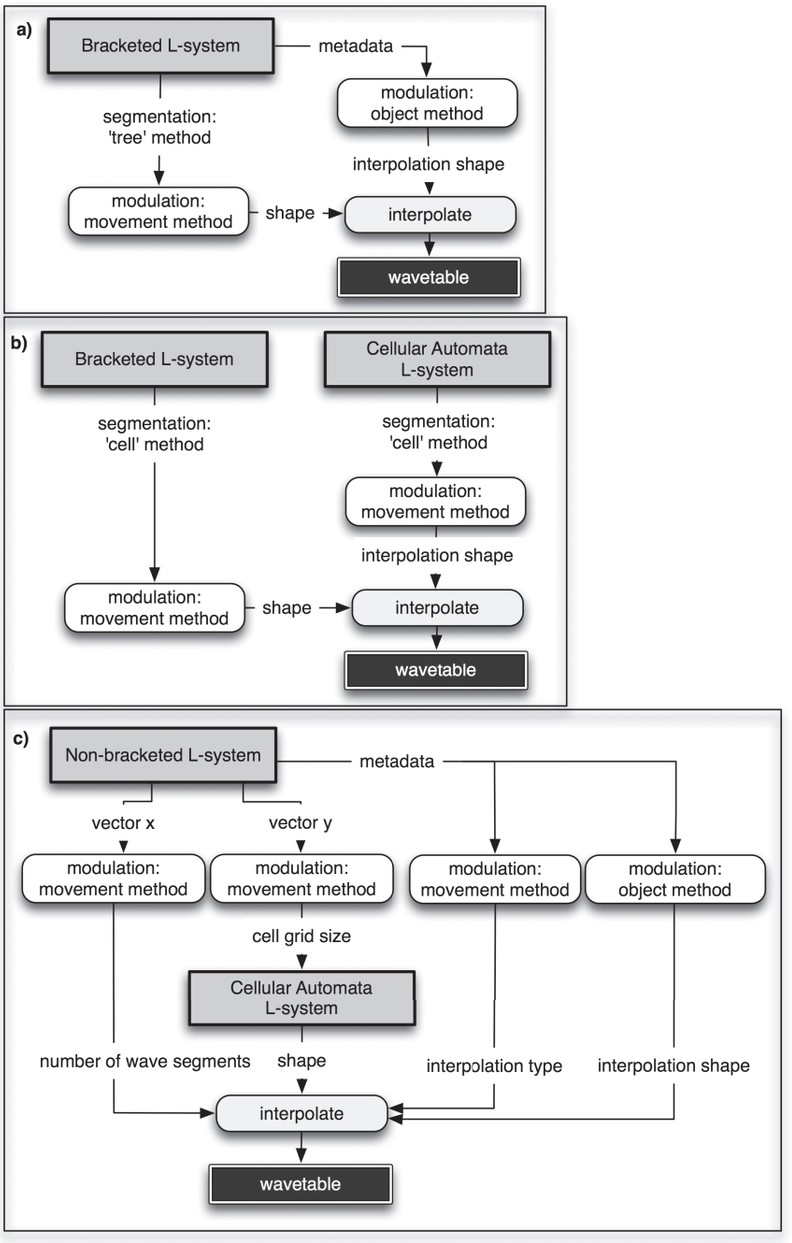


Fig. 10. Sample-level sound synthesis with L-systems: (a) Hierarchical segmentation with movement and object mapping. (b) Using two parallel L-systems as separate layers for modulating a waveform. (c) Hierarchical sample-level synthesis with a turtle automaton controlling a cellular automata world. (© Stelios Manousakis)

Modulation

A segment can be modulated using one of two methods.

Movement Method. An L-system may generate multiple automata active in *n*-dimensions. Each dimension can be mapped to a different waveform, with each automaton controlling a particular segment of each waveform as defined in the segmentation process (Figs 6–8). Typically, the sample values of a segment are modulated iteratively, deriving control from the automaton trajectory instead of its absolute position. The latter can be used, but iterative mappings are noticeably more interesting and flexible.

The amount of overall modulation is controlled by a scaling parameter. A clipping function defines the range of the amplitude space and the character of its edges. There are three options:

1. Wall—exceeding values are clipped to the range,
2. Elastic barrier—exceeding values are reflected, and
3. Circular space—exceeding values are wrapped around the other edge.

Object Method. A symbol in the alphabet can be interpreted as a metadata command to insert an "object" in the table segment—a function, shape or distribution consisting of *n* values. These objects can be compound data structures further defined through a set of additional parameters, which in turn can be set by other metadata symbols in the string or by an automaton state. The "objects" can therefore be reshaped and modified in discrete or continuous manners. When such a symbol is parsed, the object is inserted in the table in the area of activity of the respective automaton (Fig. 9).

Object mapping can also be used during the interpretation rules; in that case an "object" corresponds to a curve of *n* values inserted in a dimension.

Some Sound Synthesis Strategies

An L-system can generate multiple wavetables depending on the amount of dimensions and automata involved. It is possible to use a combination or network of L-systems to generate data even on the sample level (Fig. 10). Very interesting results can also be obtained by combining and layering L-systems in various levels (Fig. 11).

L-system–generated wavetables can be used in the time domain as audio or control data, or in the frequency domain (Fig. 12).

Results

The sound synthesis method presented here is very versatile, capable of generating complex, organically evolving sounds with a large timbral palette, covering almost the full periodicity-to-aperiodicity spectrum. Satisfactory results can generally be obtained with relatively little programming on the production side.

Aside from the evident influence of the production and interpretation rules used, different mappings affect the output distinctively: In short, movement mappings tend to generate novel sounds specific to this method, whereas object mappings can approximate more familiar sound typologies, because they can function as organizational principles for sonic quanta. Hierarchical segmentations fluctuate characteristically between large and small timbral variations, as different depth-level branches are sequenced. With defined interpolation, sonorities depend greatly on the interpolation table; fractal interpolation gives rich harmonic spectra whereas random interpolation is noisier. Circular amplitude spaces give more distorted, raw and "crunchy" results, especially with high

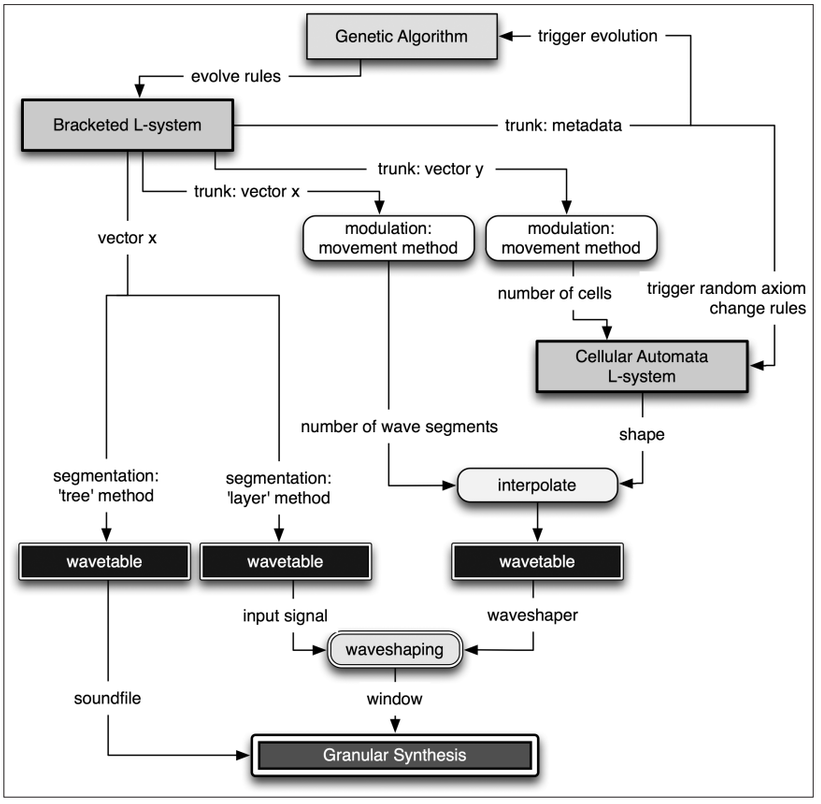


Fig. 11. Hierarchical evolving sound synthesis: An L-system triggers its own evolution and produces the waveform and window for granulation, while controlling an embedded L-system that generates a dynamic wavetable for waveshaping the window. (© Stelios Manousakis)

scaling factors, whereas elastic barriers produce rounder sounds.

Compared to other non-standard techniques, the sonic range of Musical L-systems is broader, as a large pool of mappings is coupled with a seemingly infinite number of possible production and interpretation rules. Timbre-wise, "fractal interpolation" techniques are the most similar to this technique and rule-based the least; stochastic methods fall in between but have the most larger-scale development similarities due to their automation capabilities. However, behaviors tend to feel more organic in Musical L-systems, with complex patterns emerging in time—occasionally revealing a grammar's fractal nature. A particular characteristic, especially with hierarchical mappings, is that morphing between sounds and smooth, continuous fluctuations of partials are often followed by abrupt timbral shifts that are musically dramatic.

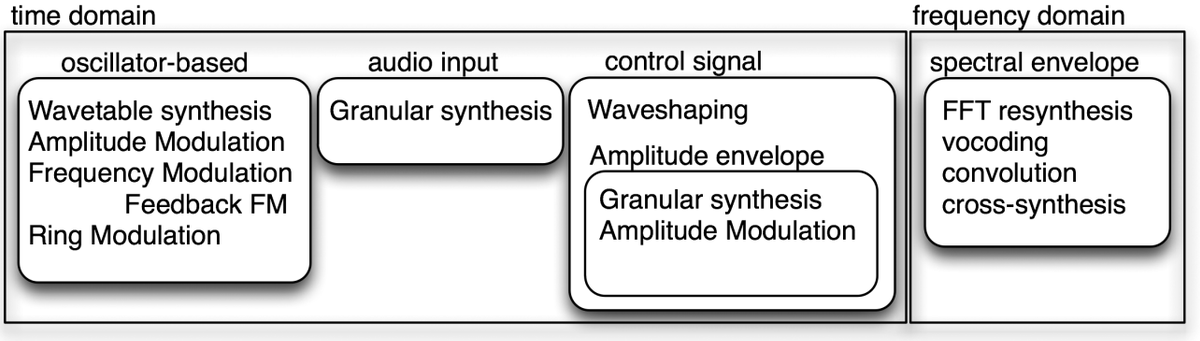
Apart from its diversity, the true power of the Musical L-system sound synthesis is that it belongs to a larger system that unifies the compositional procedure by using a single method for generating both structure and content in all time-levels, thus simultaneously addressing the problems of synthesis and control. The underlying philosophy is that a musical piece can be designed as a complex dynamical system, which the composer shapes and directs without having to explicitly specify every detail. Using this principle, I have composed two pieces [19]: In "Do Digital Monkeys Inhabit Virtual Trees?," L-systems generate all control structures and the sound material for a controlled granular synthesis instrument with ring modulators and control-rate waveshapers. "Undercover Harpsichord Agents Terrorize the Court" features a similar setup, except that the original material is sampled (harpsichord); this is a hybridization, using the same techniques for dynamically transforming an input in non-standard manners, instead of synthesizing source material from scratch.

Future Work

This multifaceted model still has potential for further development. Future work includes, firstly, porting the current implementation from Max/MSP/Jitter to SuperCollider, allowing for smoother implementation and incorporating more elements from the original model [20], such as stochastic and parametric rules. Concerning sound synthesis, development paths include:

- Designing an L-system wavetable oscillator that links parsing speed to playback frequency, thus changing the wavetable contents after each cycle.
- Implementing individual and dynamic edges for each segment, controllable by the assigned automaton.
- Using audio input buffers as objects to be inserted and transformed and further experimenting with using L-systems to process live audio in real-time on the sample level.
- Developing a static interpretation for non-bracketed L-systems not based

Fig. 12. Possible uses for L-system generated wavetables. (© Stelios Manousakis)



on waveform segmentation, in which the curve formed by the trajectory of an automaton in a dimension during one generation is interpreted as a waveform. This is implemented in the current system only for higher-level parameters.

References

1. S.R. Holtzman, "An Automated Digital Sound Synthesis Instrument," *Computer Music Journal* **3**, 53–61 (1979).

2. P. Berg, R. Rowe, D. Theriault, "SSP and Sound Description," *Computer Music Journal* **4**, No. 1, 25–35 (1980).

3. P. Berg, "PILE: a language for sound synthesis," *Computer Music Journa*l **3**, No. 1, 30–37 (1979).

4. H. Brün, A. Chandra. "A Manual for SAWDUST," <grace.evergreen.edu/~arunc/brun/sawdust/sawdust.html> (2001).

5. Holtzman [1].

6. I. Xenakis. *Formalized Music* (Stuyvesant, NY: Pendragon Press, 1992) p. 387.

7. M.H. Serra, "Stochastic Composition and Stochastic Timbre: GenDy 3 by Iannis Xenakis," *Perspectives of New Music* **31**, No. 1, 236–257 (1993).

8. N. Valsamakis and E.R. Miranda, "Extended Waveform Segment Synthesis, a non-standard synthesis model for composition," *Proceedings of the Sound and Music Computing Conference* (2005).

9. A. Chandra, "The Linear Change of Waveform Segments Causing Non-Linear Changes of Timbral Presence," *Contemporary Music Review* **10**, No. 2, 157–169 (1994).

10. S.D. Yadegari, "Using Self-Similarity for Sound-Music Synthesis," *Proceedings of International Computer Music Conference,* Montreal (1991).

11. G. Monro, "Fractal Interpolation Waveforms," *Computer Music Journal* **19**, No. 1, 88–98 (1995).

12. J. Dashow, "Fractal Interpolation," *Computer Music Journal* **20**, No. 1, 8–10 (1996).

13. N. Collins, "Errant Sound Synthesis," *Proceedings of International Computer Music Conference,* Belfast (2008).

14. S. Manousakis, "Musical L-systems," MA thesis, Institute of Sonology, <modularbrains.net/support/SteliosManousakis-Musical_L-systems.pdf> (2006).

15. Manousakis [14].

16. P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants* (New York: Springer-Verlag, 1990) p. 228.

17. P. Prusinkiewicz, "Score Generation with L-systems," *Proceedings of the International Computer Music Conference,* Den Haag (1986).

18. M. Alfonseca, A. Ortega, "Representation of some cellular automata by means of equivalent L-Systems," *Complexity International* **7** (2000).

19. Listen at <modularbrains.net/DoDigitalMonkeysInhabitVirtualTrees.html> and <modularbrains.net/UndercoverHarpsichordAgentsTerrorizeTheCourt.html>.

20. R. Karwowski and P. Prusinkiewicz, "Design and implementation of the L+C modeling language," *Electronic Notes in Theoretical Computer Science* **86**, No. 2 (2003).

Manuscript received 2 January 2009.

Stelios Manousakis is a composer, performer and researcher with a background in music and linguistics, currently pursuing a Ph.D. at DXARTS at the University of Washington.