Chapter 19. Meeting 19, Approaches: Grammars and L-Systems

19.1. Announcements

- Sonic system draft due: 27 April
- No class Tuesday, 20 April
- Be sure to do reading for next class:

  Riskin, J. 2003. "The Defecating Duck, or, the Ambiguous Origins of Artificial Life." *Critical Inquiry* 29(4): 599-633.

19.2. Quiz

- 10 Minutes

19.3. String Rewriting Systems

- Given an alphabet and rewrite (production) rules, transform strings
- A wide variety of formalizations and approaches
- Axel Thue: first systematic treatment
- Noam Chomsky: applied concept of re-writing to syntax of natural languages

19.4. Formal Grammars

- A set of rules for a formal language
- Formal grammars can be generative or analytic
- Generative grammars defined by
  - A finite set of nonterminal symbols (variables that can be replaced)
  - A finite set of terminal symbols (constants)
  - An axiom, or initial state
  - A finite set of production rules, replacing variables with variables or constants
- Generative grammars are iterative

19.5. Lindenmayer Systems

- Based on 1968 work of Aristid Lindenmayer
- Origins in model of a natural systems: "a theoretical framework for studying the development of simple multicellular organisms"
- 1984: began use of using computer graphics for visualization of plan structures
- L-systems: formal grammars where re-writing is parallel, not sequential: all symbols are simultaneously replaced

Image: Public domain (Wikipedia)

YouTube (http://www.youtube.com/watch?v=L54SE9KTMSQ)

YouTube (http://www.youtube.com/watch?v=t-FZhw9G-RQ)

YouTube (http://www.youtube.com/watch?v=t-FZhw9G-RQ)

- Motivation from natural systems: idea of cell divisions of occurring at the same time
- L-systems can take many different forms depending on rule systems and alphabet components

19.6. Context-Free

- Rules match one source to one or more destination
- Example:

Alphabet:

V: A B

Production rules :

P1: $A \rightarrow AB$

P2: $B \rightarrow A$

axiom:

ω : B

which produces for derivation step *n*:

n=0 : B

n=1 : A

n=2 : AB

n=3 : ABA

n=4 : ABAAB

n=5 : ABAABABA

Courtesy of Stelios Manousakis. Used with permission. From "Musical L-Systems." Master's Thesis, Royal Conservatory, The Hague, 2006.

- Originally proposed by Lindenmayer to model growth of algae
- Graphic representation Prusinkiewicz and Lindenmayer (1990)

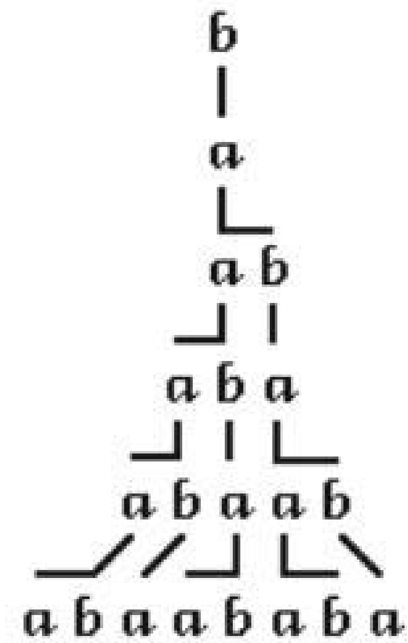


Figure 1.3: Example of a derivation in a DOL-system.

19.7. Context-Sensitive

- Rules match two or more sources to one or more destination
- 1L systems: match left or right of target source
- 2L systems: match left and right of target source
- 1L systems can be considered 2L systems with an empty (open matching) context
- Example:

Alphabet:

$V: ab$

Production rules:

$P1: b < a > \varnothing \rightarrow b$

$P2: b \rightarrow a$

axiom:

ω : baaaaaaa

which produces for derivation step *n*:

n=0 : baaaaaaa
n=1 : abaaaaaa
n=2 : aabaaaaa
n=3 : aaabaaaa
n=4 : aaaabaaa

Courtesy of Stelios Manousakis. Used with permission. From "Musical L-Systems." Master's Thesis, Royal Conservatory, The Hague, 2006.

19.8. Non-Deterministic and Table L-systems

- A context-sensitive or context free grammar can be deterministic
- If the application of rules if probabilistic, non-deterministic grammar is created
- Common approach: map one source to two or more destinations, with weighted probabilities for each destination
- Example:

Alphabet :

V: A B

Production rules :

P1: $A \xrightarrow{70\%} AB$

P2: $A \xrightarrow{30\%} BA$

P3: $B \longrightarrow A$

axiom :

ω : A

which can produce for derivation step n:

n=0 : A
n=1 : AB
n=2 : ABA
n=3 : BAAAB
n=4 : ABAABBAA

or:

n=0 : A
n=1 : BA
n=2 : AAB
n=3 : ABABA
n=4 : BAABAAAB

- Alternatively, rules can be changed during production, producing a Table L-system (Manousakis 2006, p. 29)

- Example:

Alphabet:

V: A B

axiom:

ω: **B**

Table 1:

Production rules :

$P1: A \rightarrow AB$

$P2: B \rightarrow A$

Table 2:

Production rules :

$P1: A \rightarrow B$

$P2: B \rightarrow BA$

If the set changes on derivation step n=3, this would produce:

T1 n=0 : B
n=1 : A
n=2 : AB
T2 n=3 : BBA
n=4 : BABAB
n=5 : BABBABBA

Courtesy of Stelios Manousakis. Used with permission. From "Musical L-Systems." Master's Thesis, Royal Conservatory, The Hague, 2006.

19.9. Non-Propagative L-systems

- Where rules replace source with more than one successor, the system grows and is propagative
- If rules only encode one-value destinations, the rule system is non-propagative

- Context-sensitive non-propagative L-systems are identical to a standard 1D CA
- Example:

Alphabet :

V : A B

Production rules :

$P1: A < A > A \rightarrow B$

$P2: A < A > B \rightarrow A$

$P3: A < B > A \rightarrow B$

$P4: A < B > B \rightarrow A$

$P5: B < A > A \rightarrow A$

$P6: B < A > B \rightarrow B$

$P7: B < B > A \rightarrow A$

$P8: B < B > B \rightarrow B$

axiom :

ω: BBBBBBBBBABBBBBBBBBB

Using the classic CA visualization for this grammar, and interpreting A = 1 (black pixel) and B = 0 (grey pixel), the first 30 generations look like this:

Figure 2.3. Cellular automata with L-systems.

Courtesy of Stelios Manousakis. Used with permission. From "Musical L-Systems." Master's Thesis, Royal Conservatory, The Hague, 2006.

19.10. Musical and Artistic Application of L-systems

- First published implementation: Prusinkiewicz

  Prusinkiewicz, P. 1986. "Score Generation with L-Systems." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 455-457.

- A spatial mapping of 2D graphical output of L-system curves to pitch (vertical) and duration (horizontal)

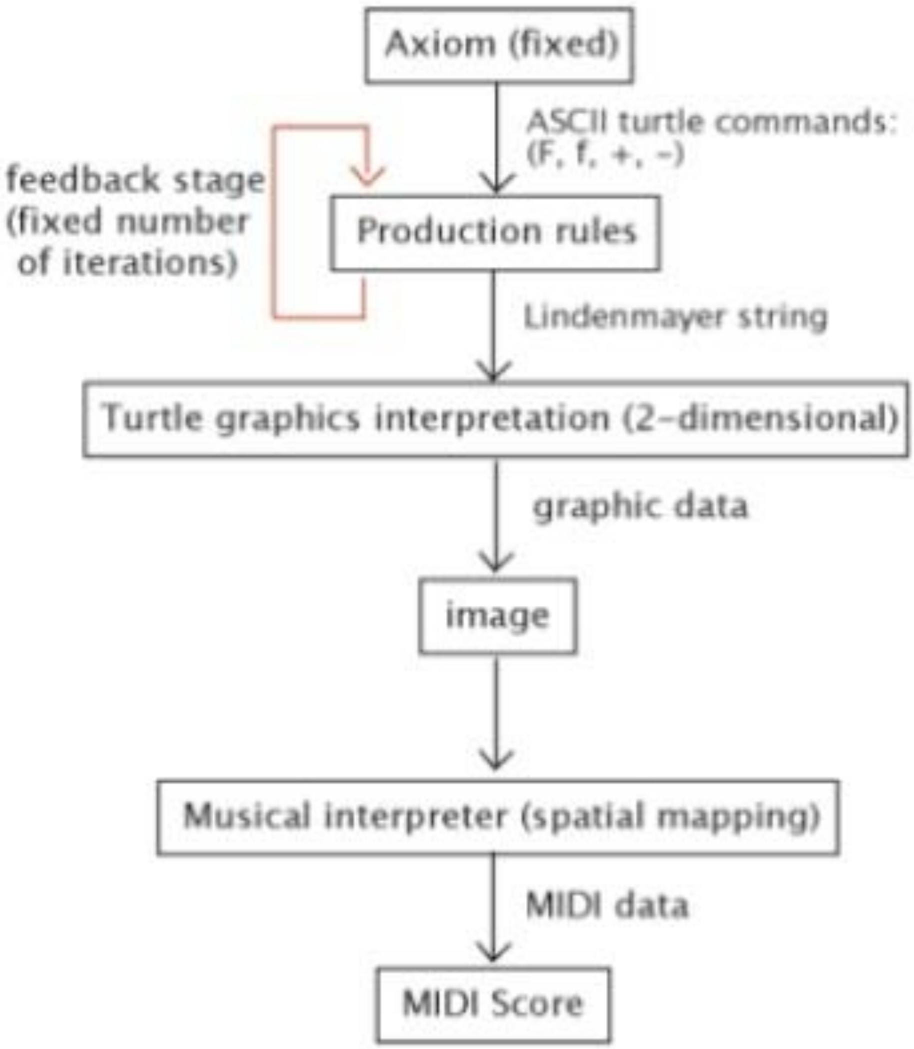
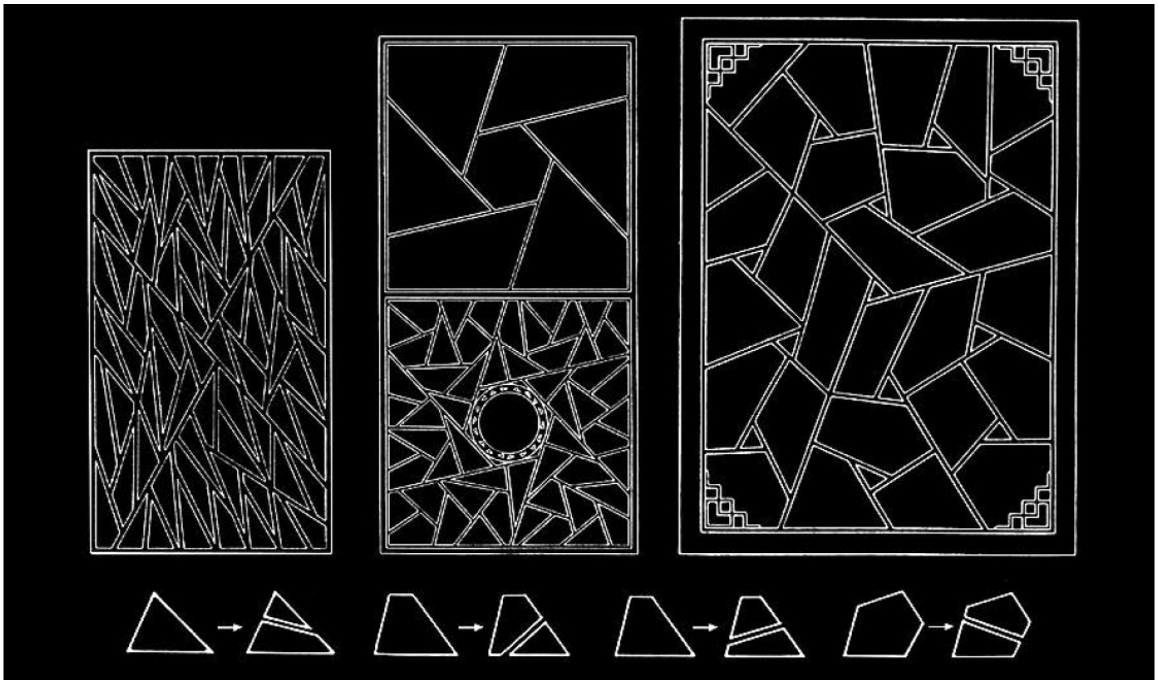


Figure 3.1 Prusinkiewicz' s L-system interpreter concept.

Courtesy of Stelios Manousakis. Used with permission. From "Musical L-Systems." Master's Thesis, Royal Conservatory, The Hague, 2006.

- States determine intervals, not absolute values
- Suggest application to other parameters: tempo, amplitude, and position of sound in space
- Creative applications in the visual arts and architecture

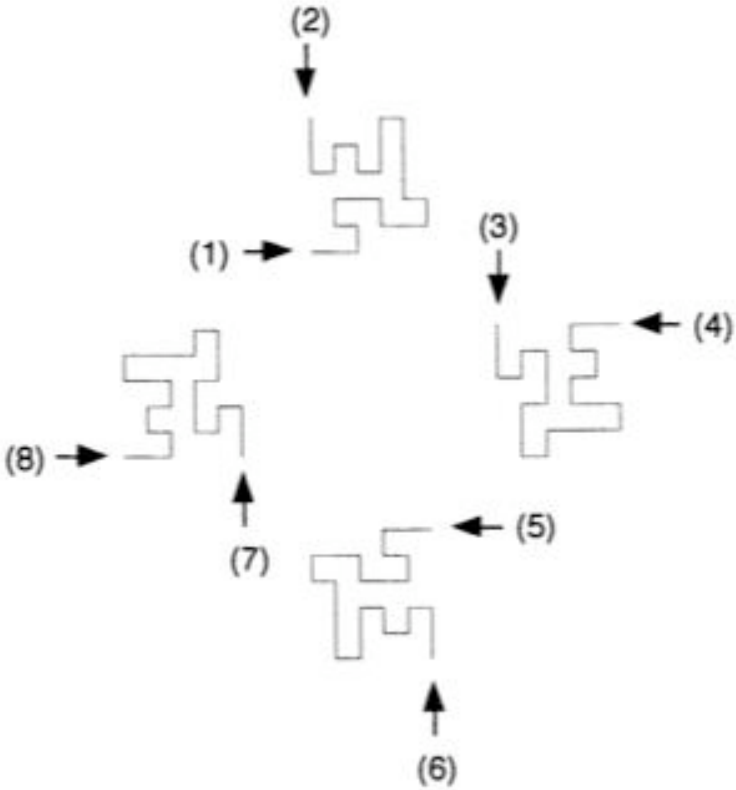
Stiny, G. and J. Gips. 1972. "Shape Grammars and the Generative Specification of Painting and Sculpture." In *Information Processing 71*. C. V. Freiman, ed. Amsterdam: North Holland. 1460-1465. Internet: http://www.shapegrammar.org/ifip/.



Courtesy of George Stiny. Used with permission.

19.11. Reading: Mason and Saffle

- Mason, S. and M. Saffle. 1994. "L-Systems, Melodies and Musical Structure." *Leonardo Music Journal* 4: 31-38.
- Are deterministic CA always fractal?
- The basic mapping (after Prusinkiewicz)



(a)

(b)

Source: Mason, S. and M. Saffle. "L-Systems, Melodies and Musical Structure." *Leonardo Music Journal* 4 (1994): 31-38.

- What are some alternative ways the 2D turtle graphics can be mapped and as musical values?
- Is it significant that “any melody can be modeled with an L-system, including the songs of aboriginal hunters, the plainchants of the medieval christian liturgy, the themes of beethoven’s symphonies and popular song tunes,” as the authors claim?
- What is the implied connection between fractals and beauty. Is this connection sufficiently supported?

19.12. A Grammar Specification String and Python Implementation

- Define a grammar in two required parts: alphabet and rules

  Both are specified in key{value} pairs

  Rules are specified as source{destination} pairs

  ```
  a{3}b{-2} @ a{b} b{a}
  ```

- Optionally include the axiom (one chosen at random otherwise)

  ```
  a{3}b{-2} @ a{b} b{a} @ baba
  ```

- Permit one to many rules of any size

  ```
  a{3}b{-2} @ a{ba} b{abb} @ baba
  ```

- Permit context sensitivity as many to one or many to many rules (not yet implemented)

  ```
  a{3}b{-2} @ aa{ba} bab{abb} @ baba
  ```

- Match any source as pattern specified with quasi regular expressions (not yet implemented)

  ```
  a{3}b{-2} @ *aa{ba} b*b{abb} bb*{abb} @ baba
  ```

- Configure non-deterministic destinations as two or more weighted options

  Weights can be specified with a floating point or integer value following the destination

  ```
  a{3}b{-2} @ a{ba|ab} b{abb=3|aa=2} @ baba
  ```

- Can create a grammar instance and view step-wise output

  ```
  >>> from athenaCL.libATH import grammar
  >>> g = grammar.Grammar()

  >>> g.load('a{3}b{-2} @ a{b} b{a} @ baba')
  >>> g.next(); g.getState()
  'baba'
  >>> g.next(); g.getState()
  'abab'
  >>> g.next(); g.getState()
  'baba'
  ```

```
>>> g.load('a{3}b{-2} @ a{ba|ab} b{abb=3|aa=2} @ baba')
>>> g.next(); g.getState()
'abbababbba'
>>> g.next(); g.getState()
'baaaabbababbbaabbabbaaba'
>>> g.next(); g.getState()
'aaabbaabbaabbaabaabbabaaaaabbbaababbabbbaabbabbbabaabbba'
```

- Can translate grammar string back into a list of source values

```
>>> from athenaCL.libATH import grammar
>>> g = grammar.Grammar()
>>> g.load('a{3}b{-2} @ a{ba|ab} b{abb=3|aa=2} @ baba')
>>> g.next(); g.getState()
'abbababbba'
>>> g.getState(values=True)
[3.0, 3.0, 3.0, -2.0, 3.0, -2.0, -2.0, -2.0, 3.0]
```

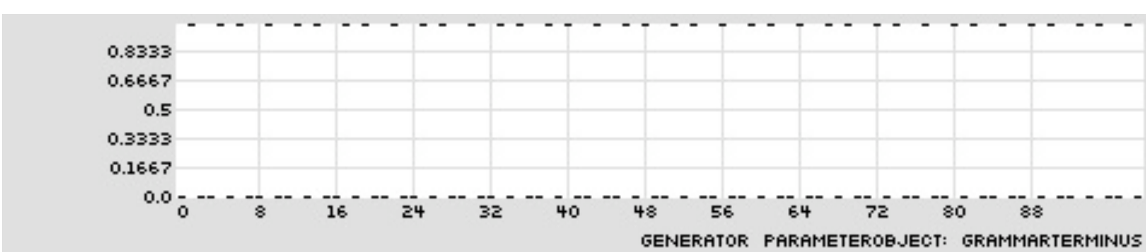
19.13. Grammar as ParameterObject

- The grammarTerminus ParameterObject

```
:: tpv grammar
Generator ParameterObject
{name,documentation}
grammarTerminus     grammarTerminus, grammarString, stepCount, selectionString
                    Description: Produces values from a one-dimensional string
                    rewrite rule, or generative grammar. The terminus, or final
                    result of the number of generations of values specifed by
                    the stepCount parameter, is used to produce a list of
                    defined values. Values are chosen from this list using the
                    selector specified by the selectionString argument.
                    Arguments: (1) name, (2) grammarString, (3) stepCount, (4)
                    selectionString {“randomChoice”, “randomWalk”,
                    “randomPermutate”, “orderedCyclic”,
                    “orderedCyclicRetrograde”, “orderedOscillate”}
```

- The Lindenmayer algae model after 10 generations

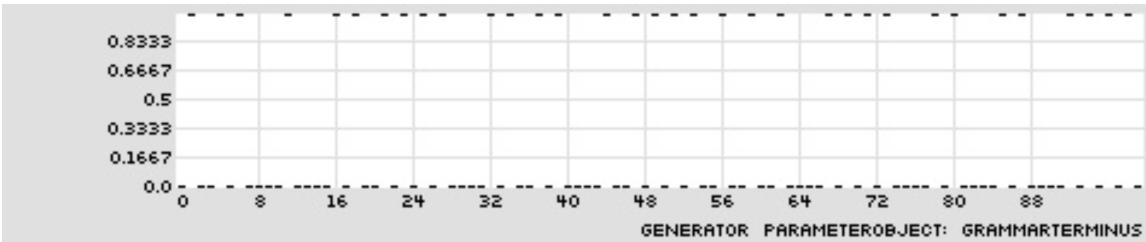
```
:: tpmap 100 gt,a{0}b{1}@a{ab}b{a}@b,10,oc
grammarTerminus, a{0}b{1}@a{ab}b{a}@b, 10, orderedCyclic
TPmap display complete.
```



- Modified Lindenmayer algae model after 10 generations with non-deterministic rule variation

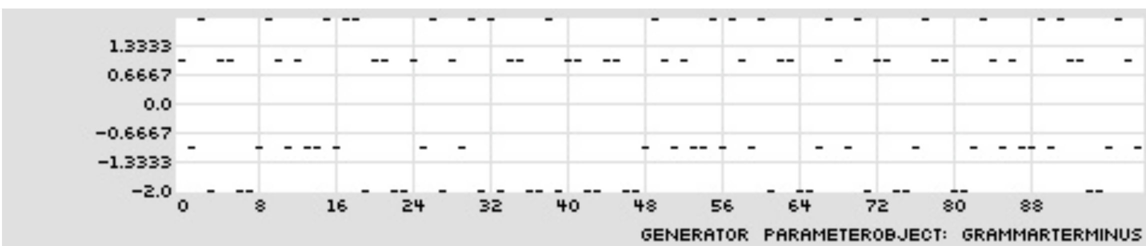
```
:: tpmap 100 gt,a{0}b{1}@a{ab}b{a|aaa}@b,10,oc
```

```
grammarTerminus, a{0}b{1}@a{ab}b{a=1|aaa=1}@b, 10, orderedCyclic
TPmap display complete.
```



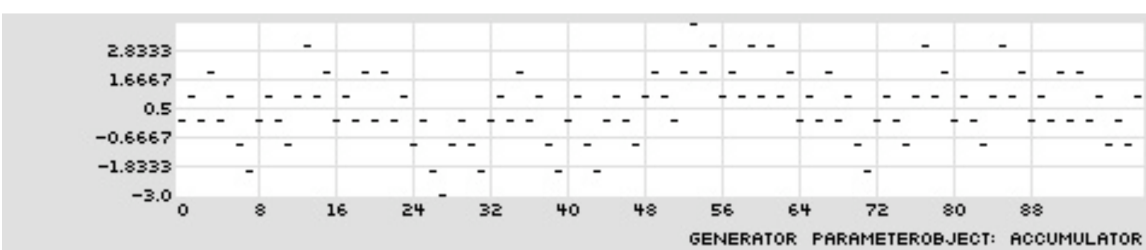
- Four state deterministic grammar

```
:: tpmap 100 gt,a{1}b{-1}c{2}d{-2}@a{ab}b{cd}c{aadd}d{bc}@ac,10,oc
grammarTerminus, a{1}b{-1}c{2}d{-2}@a{ab}c{aadd}b{cd}d{bc}@ac, 10,
orderedCyclic
TPmap display complete.
```



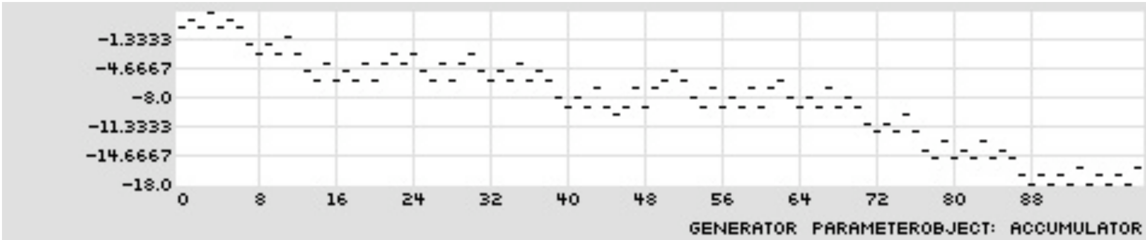
- Four state deterministic grammar placed in an Accumulator PO

```
:: tpmap 100 a,0,(gt,a{1}b{-1}c{2}d{-2}@a{ab}b{cd}c{ad}d{bc}@ac,10,oc)
accumulator, 0, (grammarTerminus, a{1}b{-1}c{2}d{-2}@a{ab}c{ad}b{cd}d{bc}@ac,
10, orderedCyclic)
TPmap display complete.
```



- Four state non-deterministic grammar placed in an Accumulator PO

```
:: tpmap 100 a,0,(gt,a{1}b{-1}c{2}d{-2}@a{ab}b{cd}c{ab|ca}d{ba|db}@ac,10,oc)
accumulator, 0, (grammarTerminus,
a{1}b{-1}c{2}d{-2}@a{ab}c{ab=1|ca=1}b{cd}d{ba=1|db=1}@ac, 10, orderedCyclic)
TPmap display complete.
```



- Alternative approaches to PO interface?
- Mappings and applications in athenaCL?

19.14. Grammar States as Accent Patterns

- Can treat the grammar alphabet as parameter values: integers, floating point values
- Command sequence:
  - emo mp
  - tmo lg
  - tin a 60
  - *non deterministic binary algae generator applied to accent*

    tie r pt,(c,8),(c,1),(gt,a{0}b{1}@a{ab}b{a|aaa}@b,10,oc)
  - tie a c,1
  - *four state deterministic applied to pulse multiplier*

    tie r pt,(c,8), (gt,a{1}b{2}c{4}d{8}@a{ab}b{cd}c{aadd}d{bc}@ac,10,oc),(c,1)
  - *four state deterministic applied to amplitude with different start string*

    tie a gt,a{.25}b{.5}c{.75}d{1}@a{ab}b{cd}c{aadd}d{bc}@bbc,6,oc
  - *four state deterministic applied to transposition with different start string*

    tie f gt,a{0}b{1}c{2}d{3}@a{ab}b{cd}c{aadd}d{bc}@dc,6,oc
  - *four state non-deterministic applied to transposition with different start string*

    tie f gt,a{0}b{1}c{2}d{3}@a{ab}b{cd|aa}c{aadd|cb}d{bc|a}@dc,6,oc
  - eln; elh

19.15. Grammar States as Pitch Values

- Can treat the grammar alphabet as specific pitch values
- Command sequence:
  - emo m
  - tmo lg
  - tin a 32
  - *four state deterministic applied to pulse multiplier*

    tie r pt,(c,8), (gt,a{1}b{2}c{4}d{8}@a{ab}b{cd}c{aadd}d{bc}@ac,8,oc),(c,1)
  - tie o c,-2
  - *four state deterministic applied to transposition with different start string*

    tie f gt,a{0}b{7}c{8}d{2}@a{ab}b{cd}c{aadd}d{bc}@ad,6,oc
  - *four state deterministic applied to amplitude with different start string*

    tie a gt,a{.25}b{.5}c{.75}d{1}@a{ab}b{cd}c{aadd}d{bc}@bbc,6,oc
  - eln; elh

19.16. Grammar States as Pitch Transpositions

- Can treat the grammar alphabet as transpositions iteratively processed through an Accumulator
- Command sequence:
  - emo m
  - tmo lg
  - tin a 15
  - *four state deterministic applied to pulse multiplier*

    tie r pt,(c,8), (gt,a{1}b{2}c{4}d{8}@a{ab}b{cd}c{aadd}d{bc}@ac,8,oc),(c,1)
  - *four state deterministic applied to accumulated transposition with different start string*

    tie f a,0,(gt,a{1}b{-1}c{7}d{-7}@a{ab}b{cd}c{ad}d{bc}@ac,10,oc)
  - *four state deterministic applied to amplitude with different start string*

tie a gt,a{.25}b{.5}c{.75}d{1}@a{ab}b{cd}c{aadd}d{bc}@bbc,6,oc

- eln; elh

19.17. Grammar States as Path Index Values

- Can treat the grammar alphabet as index values from the Path iteratively processed through an Accumulator
- Command sequence:
  - emo m
  - *create a single, large Multiset using a sieve*

    pin a 5@0|7@2,c2,c7
  - tmo ha
  - tin a 6
  - *constant rhythm*

    tie r pt,(c,4),(c,1),(c,1)
  - *select only Multiset 0*

    tie d0 c,0
  - *select pitches from Multiset using accumulated deterministic grammar starting at 12*

    tie d1 a,12,(gt,a{1}b{-1}c{2}d{-2}@a{ab}b{cd}c{ad}d{bc}@ac,10,oc)
  - *create only 1 simultaneity from each multiset; create only 1-element simultaneities*

    tie d2 c,1; tie d3 c,1
  - *four state deterministic applied to amplitude with different start string*

    tie a gt,a{.25}b{.5}c{.75}d{1}@a{ab}b{cd}c{aadd}d{bc}@bbc,6,oc
  - eln; elh