Two-State Graph-Rewriting Automata

Kohji Tomita*,
Haruhisa Kurokawa*,
Satoshi Murata**

* National Institute of Advanced Industrial Science and Technology (AIST)
** Tokyo Institute of Technology

NKS 2006 Conference, Jun 18, 2006
Lattice-based symbol dynamics

- Cellular Automata Model
  - Self-Reproducing automata by von Neumann in 1950s
  - State transition on lattice space
  - Life Game, Lattice Gas Automata
  - New Kind of Science by Wolfram
  - Cell space cannot be generated
    - Infinite space or torus is assumed
Graph-rewriting automata

- Variant of graph-rewriting system
- Graph development with state transition and structure rewriting
- Rules in a regular form (like CA)
- Not restricted on lattice space
- Define cellular automata on graphs
  - Standard state transition rules
  - Rewrite rules of graph structure
- Dynamic graph automata
Why dynamic graph?

- Rich expressibility:
  - Not restricted on lattice space
  - Changing topology & number of nodes
  - Connection between remote nodes
  - Arbitrary many division of space
  - Closed surface / boundary condition
Non-lattice model

- Disadvantage:
  - Information of connection relation
    - 3-link planar graphs
  - Less simple
    - 3-link planar graphs (& regular rules)
  - Visualization
    - Embed in 3D space
Related study

- Evolution of Networks
  (NKS Ch. 9, Wolfram 2002)
- Graph grammar based systems
  – Some systems in Artificial Chemistry
    (e.g., Benkö et al. 2003)
  – Programmable Parts (Klavins et al. 2005)
  – DynaGraph (Saidani et al. 2004)
Definition of Graph-Rewriting Automata
Basic structure

- 3-link planar graph
  - Minimum to generate nontrivial structures
  - Link order at each node
- Node state:
  from arbitrary finite set
- Examples:
Graph-rewriting automata

Structural rewriting (without states)
Graph-rewriting automata

Structural rewriting 1 (with states)

trans (x, a, b, c) → y

div (x, a, b, c) → (u, v, w)

fus (u, v, w, a, b, c) → x

com (a, b, c, d, x, y) → (u, v)
Graph-rewriting automata

Structural rewriting 2 (with states)

Node rules

Link rules

trans x, (a, b, c) ≅ y

div x, (a, b, c) ≅ y

com (x, y)

anh (x, y)

(separation of graphs)
Update procedure

- Given: initial graph, rule set (list of rules)

- Deterministic update
  - Synchronous rule application
    - node rules (trans, div) at even time
    - link rules (com, anh) at odd time
  - Lateral inhibition
    - suppress neighbor link rule activation

\[
\begin{align*}
\text{div } 1(0,0,2) &\rightarrow 1 \\
\text{com } (1,2) &\rightarrow \\
\text{com } (0,0) &\rightarrow \\
\end{align*}
\]
Example 1: regular division

\[
\begin{align*}
\text{div } 0 & \ (1, 1, 1) \rightarrow 0 \\
\text{div } 1 & \ (0, 0, 0) \rightarrow 1 \\
\text{com } & \ (0, 1)
\end{align*}
\]
Example 2: self-replication of 4-node structure

- Initial state: 4-node (different states)
- 19 rules (6 states):

- com (2, 3)
- div 0 (1, 3, 3) □ 0
- div 1 (0, 2, 2) □ 1
- div 3 (0, 0, 2) □ 4
- div 2 (1, 1, 3) □ 2
- trans 0 (0, 0, 1) □ 1
- trans 1 (1, 1, 0) □ 0
- trans 4 (4, 4, 2) □ 2
- trans 2 (2, 2, 4) □ 4
- trans 4 (4, 0, 2) □ 3
- trans 4 (4, 2, 0) □ 1
- trans 0 (0, 1, 4) □ 2
- trans 2 (4, 4, 4) □ 5
- trans 4 (2, 2, 2) □ 5
- trans 1 (0, 0, 0) □ 5
- trans 0 (1, 1, 1) □ 5
- trans 2 (2, 1, 4) □ 0
- trans 1 (1, 2, 0) □ 3
- anh (5, 5)
Two-State Graph-rewriting Automata
Exhaustive trial of two-state rules

- Internal state \( \mathbb{Z} \{0, 1\} \)
- Development processes from simple initial structures

- Execution until 80 steps or 1,000 nodes
Notation of rule-set

- \([0..3]^8 [0..2]^3 = 1,769,472\)
- 11 digits (8 for node rules + 3 for link rules)

```
0: s () □ 0
1: s () □ 1
2: d () □ 0
3: d () □ 1
0: nop
1: com
2: anh
```

example: 01223110 210

```
s 0,(0,0,0) □ 0
d 1,(0,0,0) □ 1
s 0,(0,0,1) □ 1
s 1,(0,0,1) □ 1
d 0,(0,1,1) □ 0
d 0,(1,1,1) □ 0
a (0,0)
c (0,1)
```
Possible local configurations

[Diagram showing various molecular structures, with some marked as applicable (3/34).]
Results
Results – separated cases
Nodes limit – simple case
Node limit – other cases
Halt
Steps limit
Self-replicating processes
Other self-replicating patterns

03220112 020 S2

03120021 201 S1

03010212 202 S1

03120021 201 S1

03010212 202 S1
01000232 010 S1
03210010 102 S1
Programmability

- Using Many States
- Rule set design
Turing machine

- Logical model of computation
- Modeled by ladder structure in GA
Self-replicating Turing machine

- Expression of self-replicating TM
  - 20 states, 257 rules (2-symbols)

- Universal Turing machine
  - Minsky’s “small” UTM (4-symbols 7-states)
  - 30 states, 955 rules (for reproducing) +
    23 states, 745 rules (for computation)
Simulation of synchronous graph-rewriting automata by asynchronous updating model

- Arbitrary rules are applied at arbitrary time
- By explicitly introducing local synchronization by different states (like simulating SCA by ACA)
- Execution of structural change can be detected by neighbors
Alternative Formulations

(1) Non-planar, many states
(2) Dual GA
(1) Non-planar graphs

- New link rule `swap`
- Node rules
  - rule $x, (a, b, c) \rightarrow y$
- Link rules
  - rule $(x,y)$

Diagram:
- Trans
- Div
- Com
- Anh
- Swp
von Neumann style self-replication

- Self-replication with translation/transcription of encoded program in structure
  Using construction arm  (requires many states)
(2) Dual graph automata

Assign states to cells

Rewrite rules:
- Cell generation
- Cell commutation
- Cell fusion
Ex. Self-reproduction

- Initial graph: 2-state 3-cell
- Rules (10)
  a 1,3,3,1,2,0 (a 1,3,3,1,0,2)
  a 3,2,3,4,4,0
  a 4,1,3,4,3,0 (a 4,1,3,3,4,0)
  g 3,2,0,4 (g 3,0,2,4)
  c 1,1,2,0
  c 2,3,4,4
  f 0,2,4,4,0

- Generation
- Commutation
- Fusion
Conclusions & Future Work
Conclusions

- **Graph-rewriting automata**
  - System that generates its boundary condition by itself
  - Not restricted to lattice space
    - Changing topology & number of nodes

- Programmability using many states

- Various development processes with 2 states
  - Self-replication

- Some variants (non-planar, dual)
Future work in two-state case

- In preliminary stage. More analysis
  - Class III, IV behavior?
  - Localized structure?
  - Minimal rules?
- Universality
  - computational, construction
- Large scale graphs
- Visualization
- Simpler 1-D case with 2-links?
General case

- Function other than self-replication

- Extensions:
  - Asynchronous, continuous, ...
  - External environment, interaction among groups