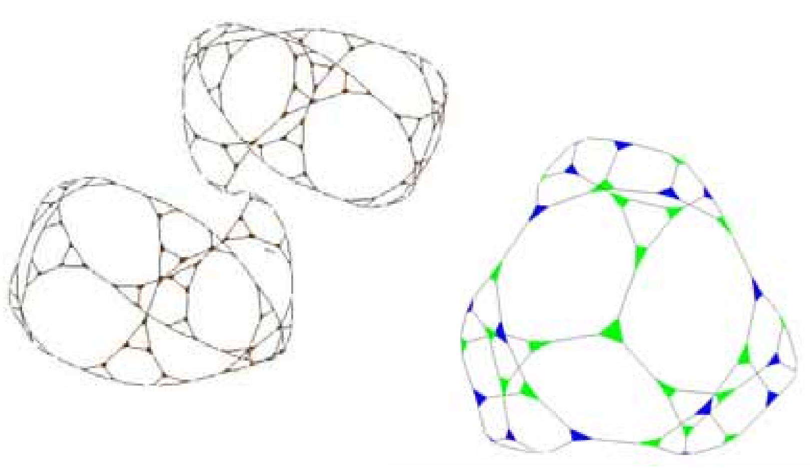
Two-State Graph-Rewriting Automata



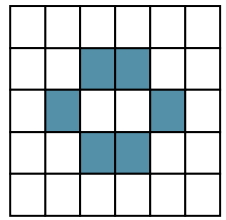
Kohji Tomita*,
Haruhisa Kurokawa*,
Satoshi Murata**

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

NKS 2006 Conference, Jun 18, 2006

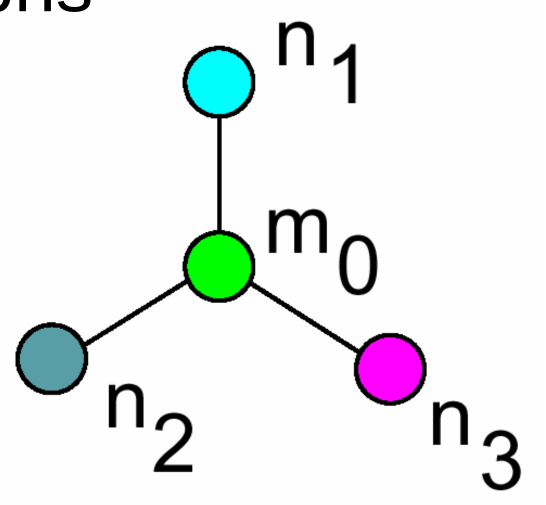
Lattice-based symbol dynamics

- 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

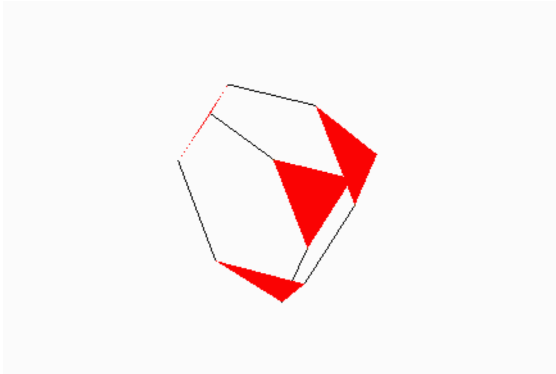


Graph-rewriting automata

- 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)

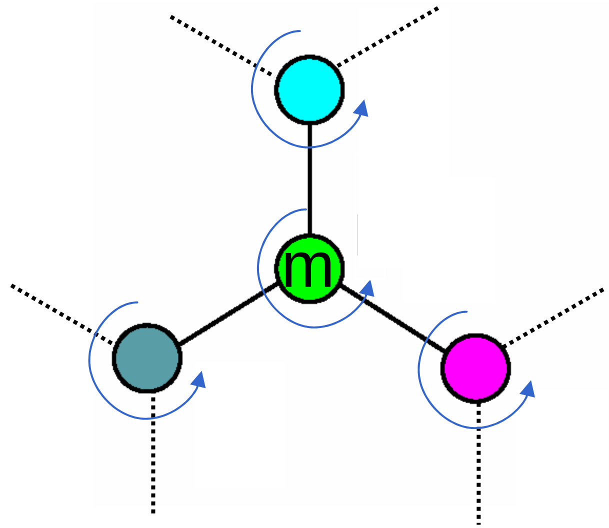
Contents

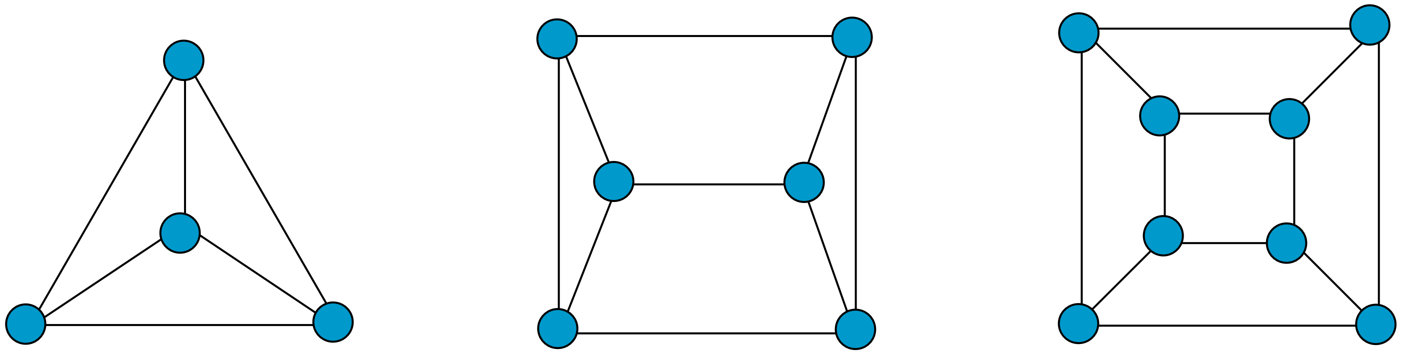
- Definition of graph-rewriting automata
- Examples
- Two-State graph-rewriting automata
- Alternative formulations
- Conclusions

Definition of Graph-Rewriting Automata

Basic structure

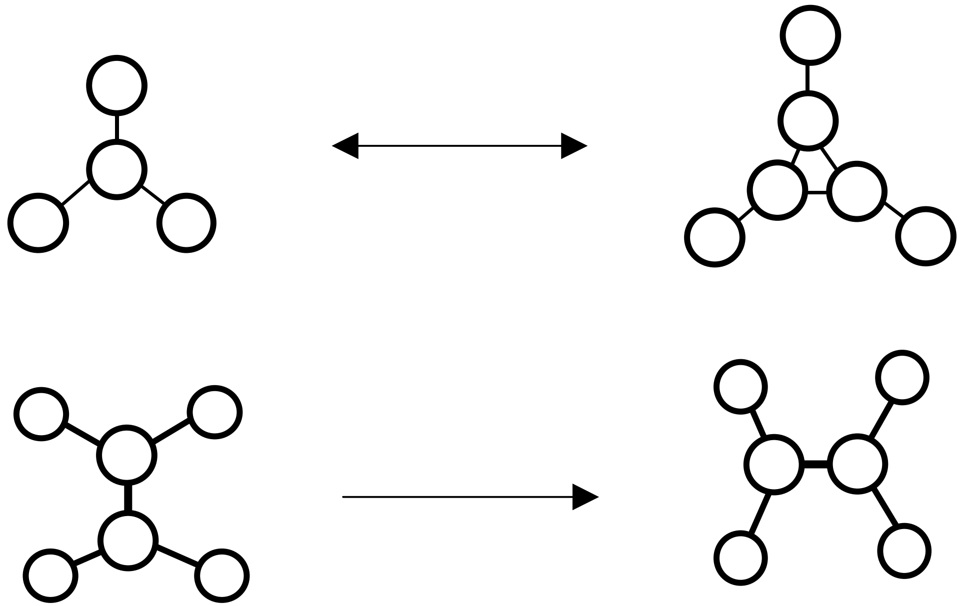
- 3-link planar graph
  - Minimum to generate nontrivial structures
  - Link order at each node
- Node state:
  from arbitrary finite set
- Examples:





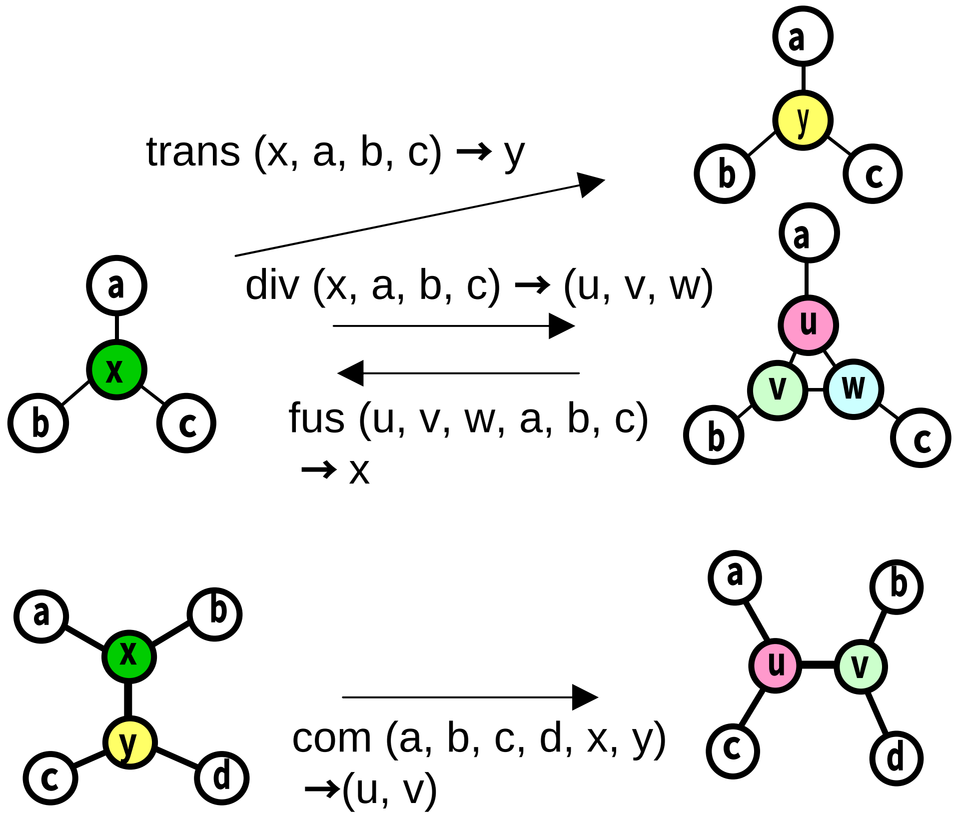
Graph-rewriting automata

Structural rewriting (without states)



Graph-rewriting automata

Structural rewriting 1 (with states)

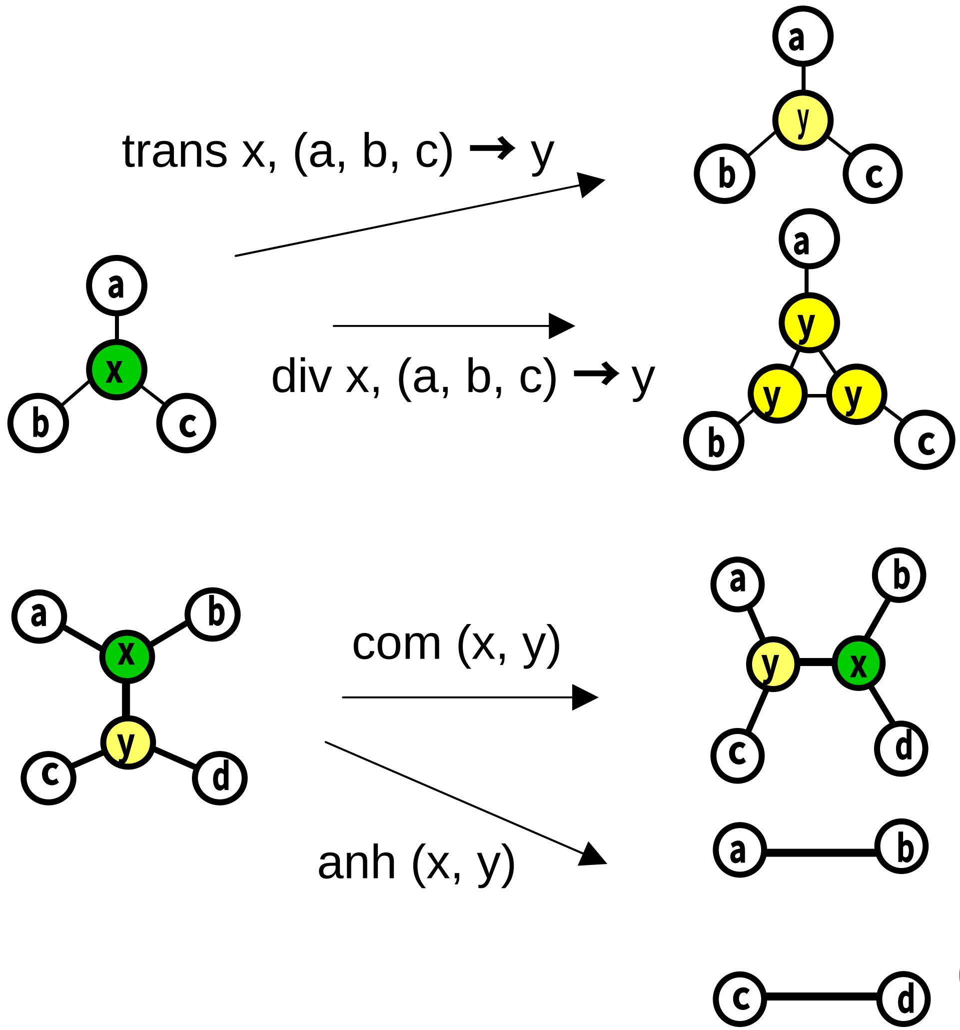


Graph-rewriting automata

Structural rewriting 2 (with states)

Node rules

Link rules

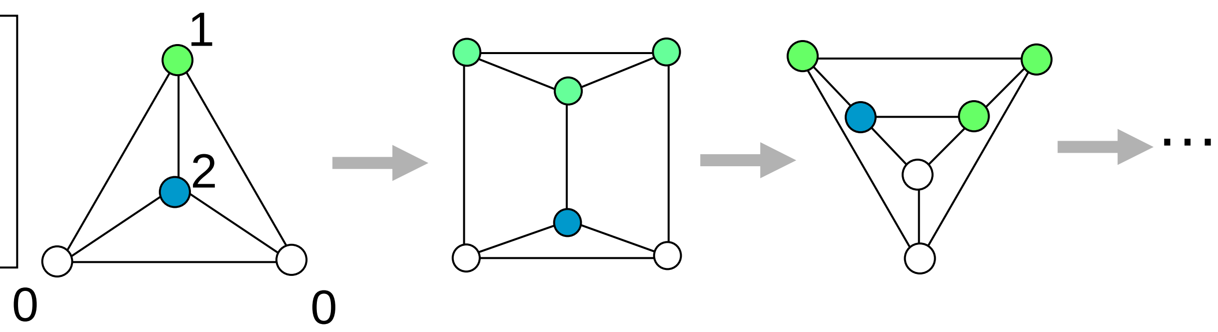


(separation of graphs)

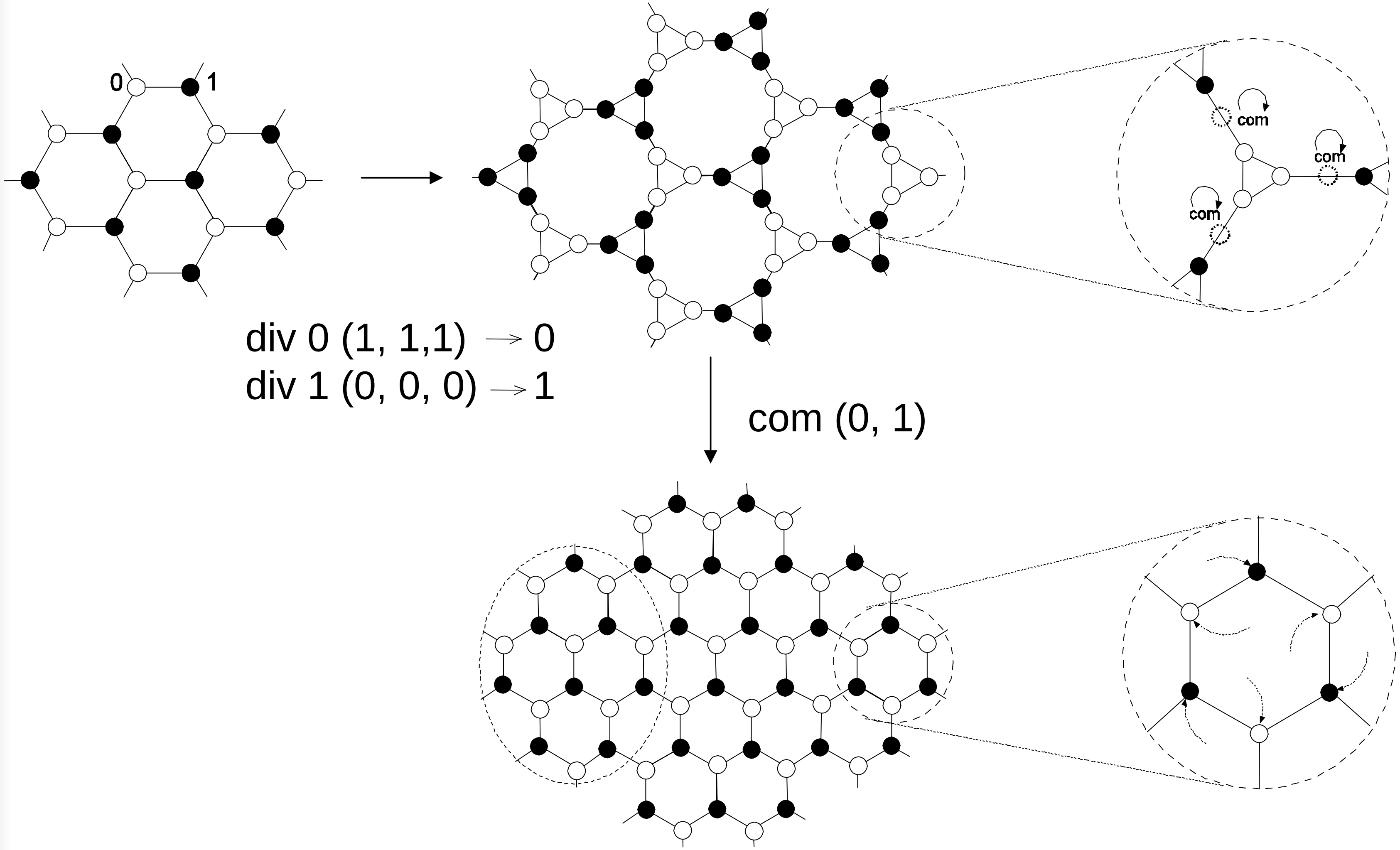
Update procedure

- 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

div 1(0,0,2)→1
com (1,2)
com (0,0)



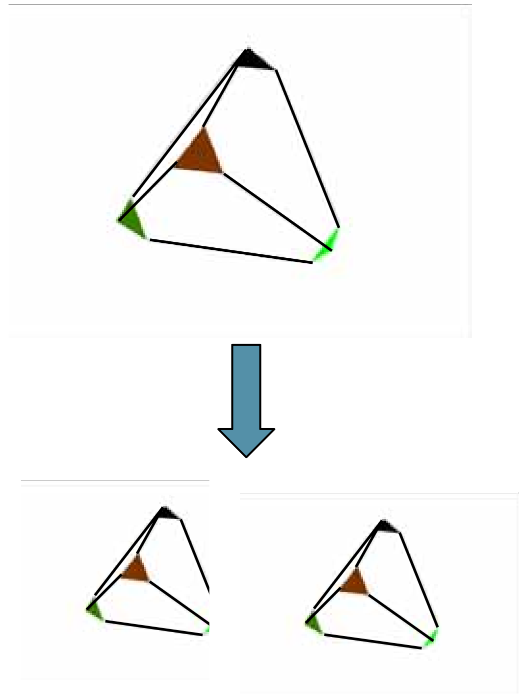
Example1: regular division



Example2: self-replication of 4-node structure

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

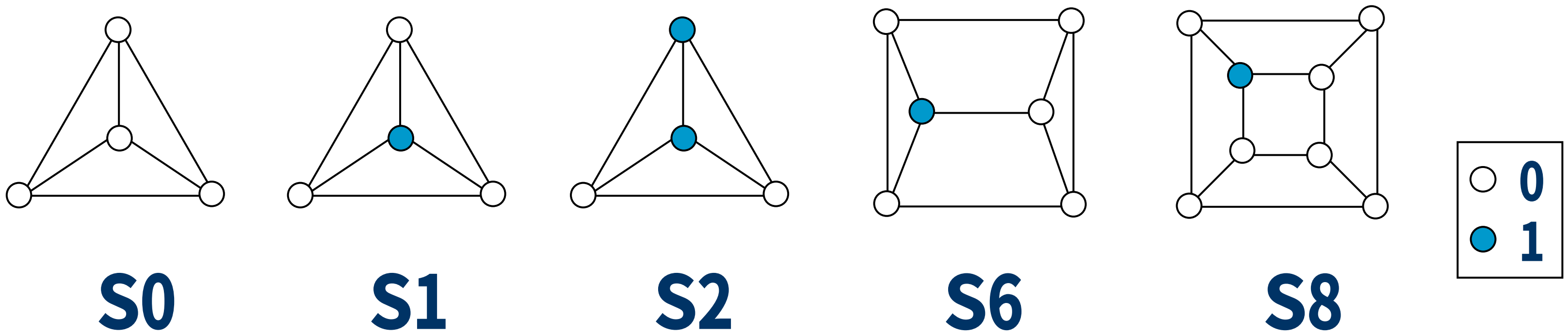
com (2, 3)
div 0 (1, 3, 3) →0
div 1 (0, 2, 2) →1
div 3 (0, 0, 2) →4
div 2 (1, 1, 3) →2
trans 0 (0, 0, 1) →1
trans 1 (1, 1, 0) →0
trans 4 (4, 4, 2) →2
trans 2 (2, 2, 4) →4
trans 4 (4, 0, 2) →3
trans 4 (4, 2, 0) →1
trans 0 (0, 1, 4) →2
trans 2 (4, 4, 4) →5
trans 4 (2, 2, 2) →5
trans 1 (0, 0, 0) →5
trans 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 ∈ {0, 1}
- Development processes from simple initial structures



- Execution until

  80 steps or 1,000 nodes

Notation of rule-set

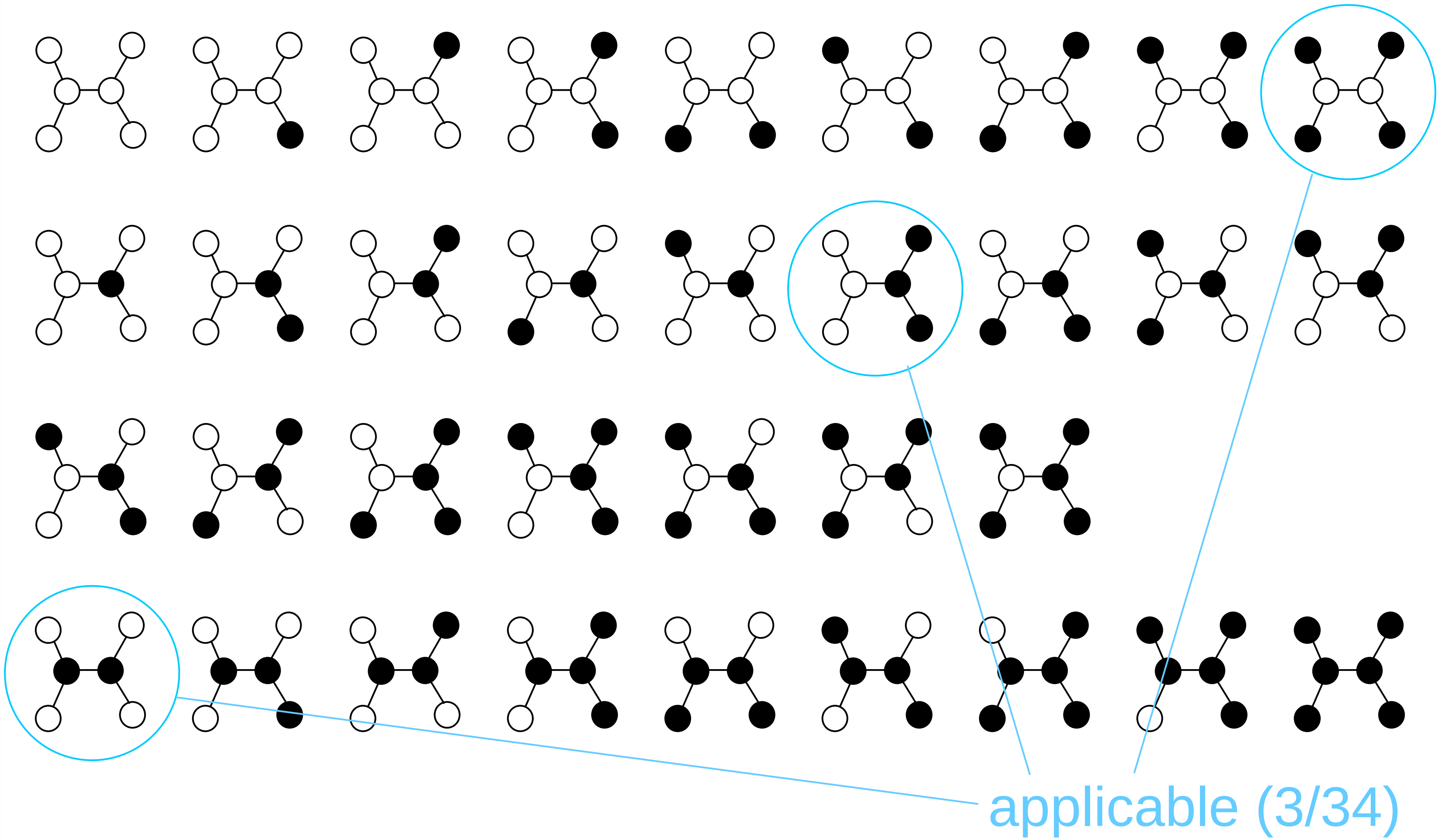
- $[0..3]^8 [0..2]^3 = 1{,}769{,}472$
- 11 digits (8 for node rules + 3 for link rules)

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

- example: 01223110 210

s 0,(0,0,0) → 0	d 1,(0,0,0) → 1	a (0,0)
s 0,(0,0,1) → 1	s 1,(0,0,1) → 1	c (0,1)
d 0,(0,1,1) → 0	s 1,(0,1,1) → 1	
d 0,(1,1,1) → 0	s 1,(1,1,1) → 0	

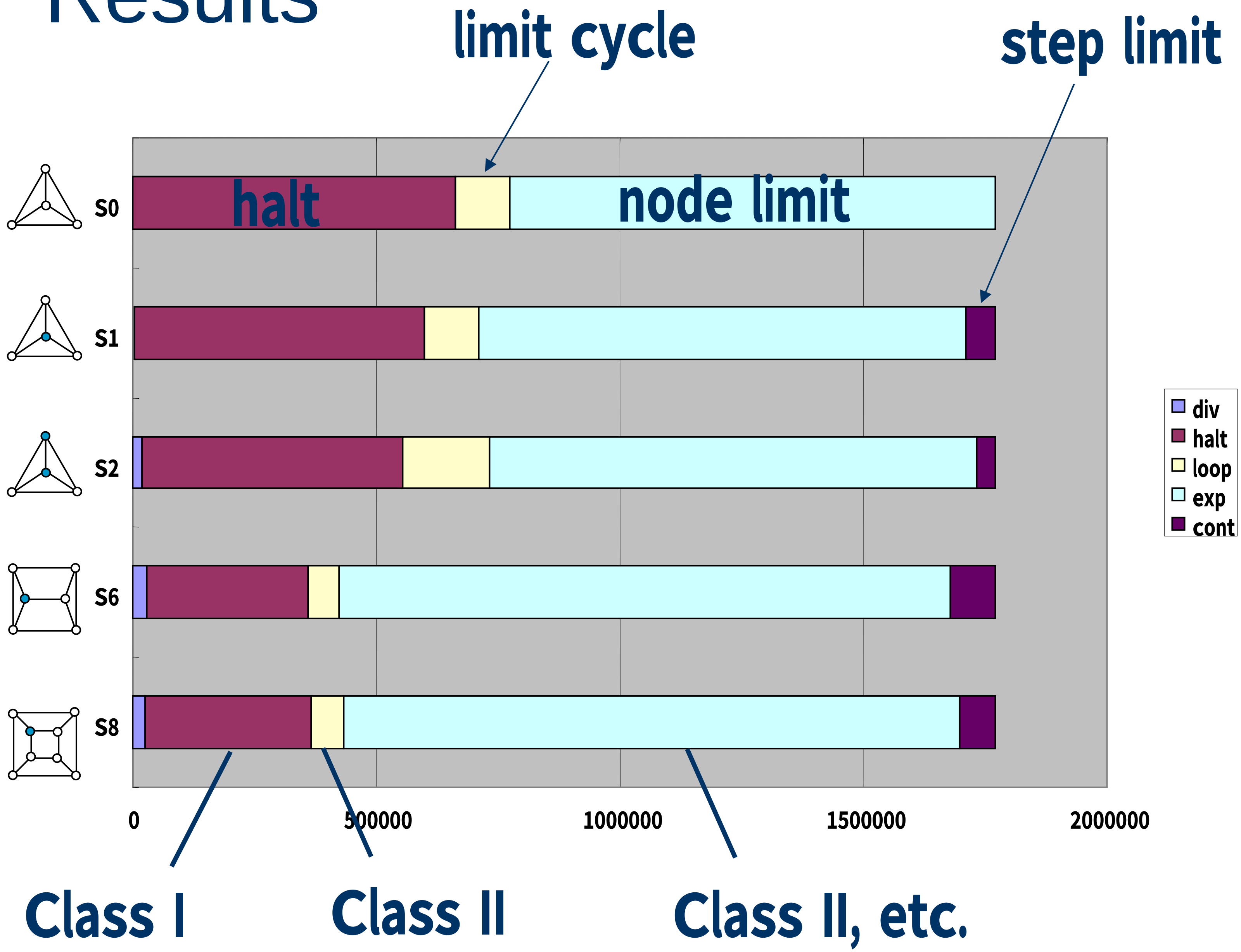
Possible local configurations



No link rule for 17/27

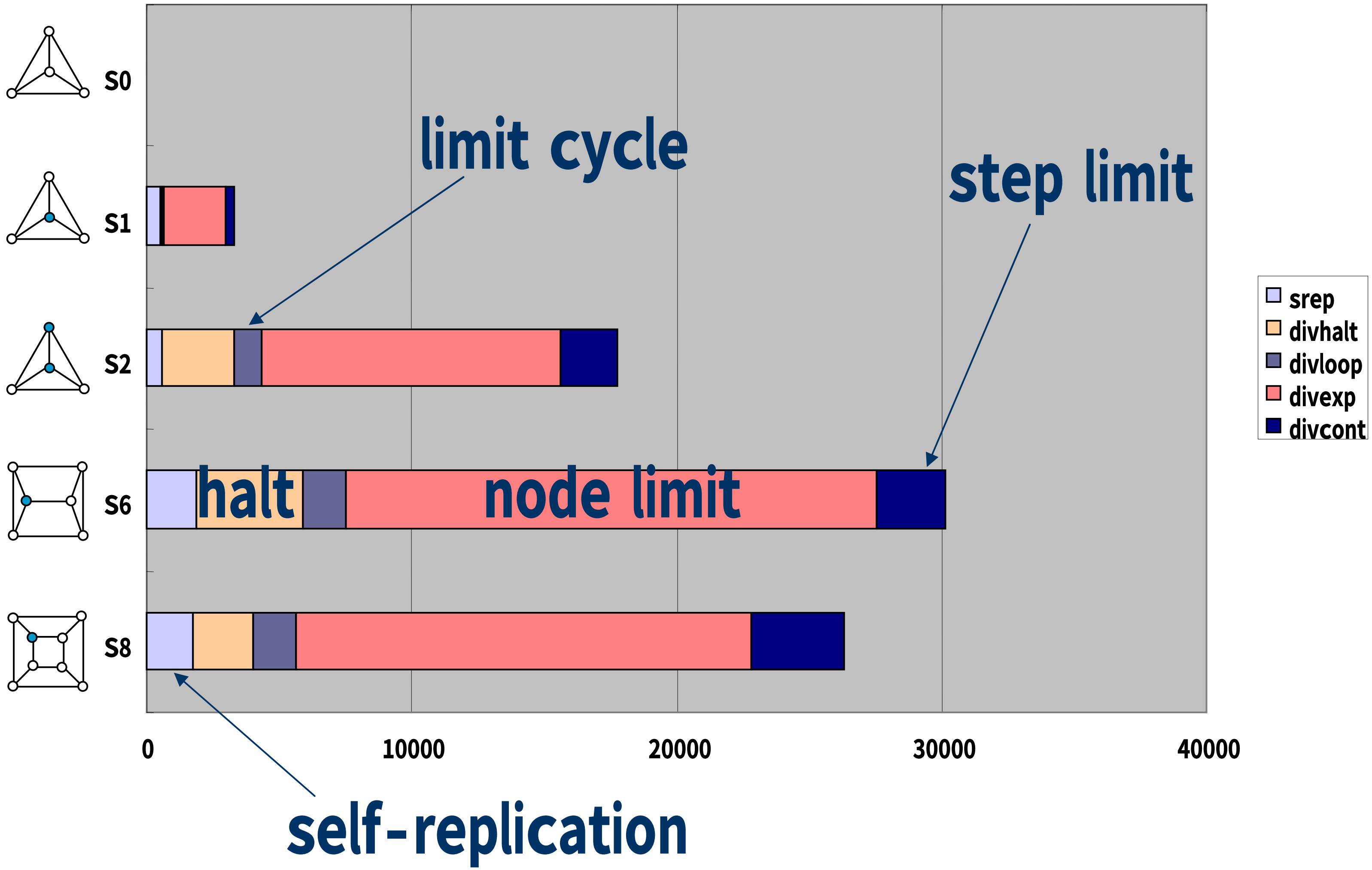
000 = 011 = 012 = 021 = 022 = 110 = 120 = 210 = 220 = 111 = 112 = 121 = 122 = 211 = 212 = 221 = 222

Results

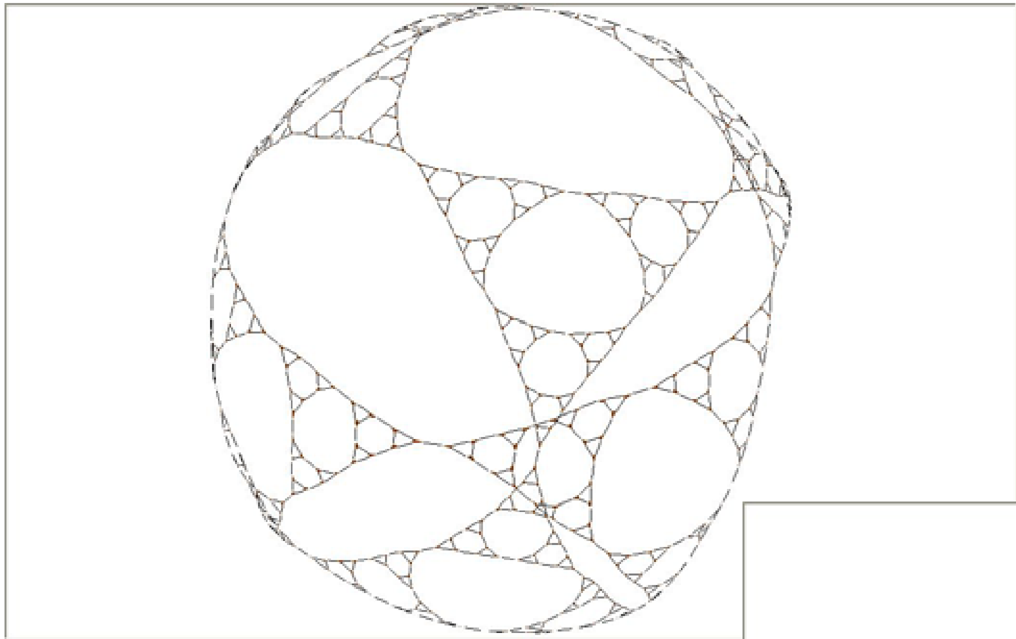


limit cycle
step limit
halt
node limit
S0
S1
S2
S6
S8
div
halt
loop
exp
cont
0
500000
1000000
1500000
2000000
Class I
Class II
Class II, etc.

Results – separated cases



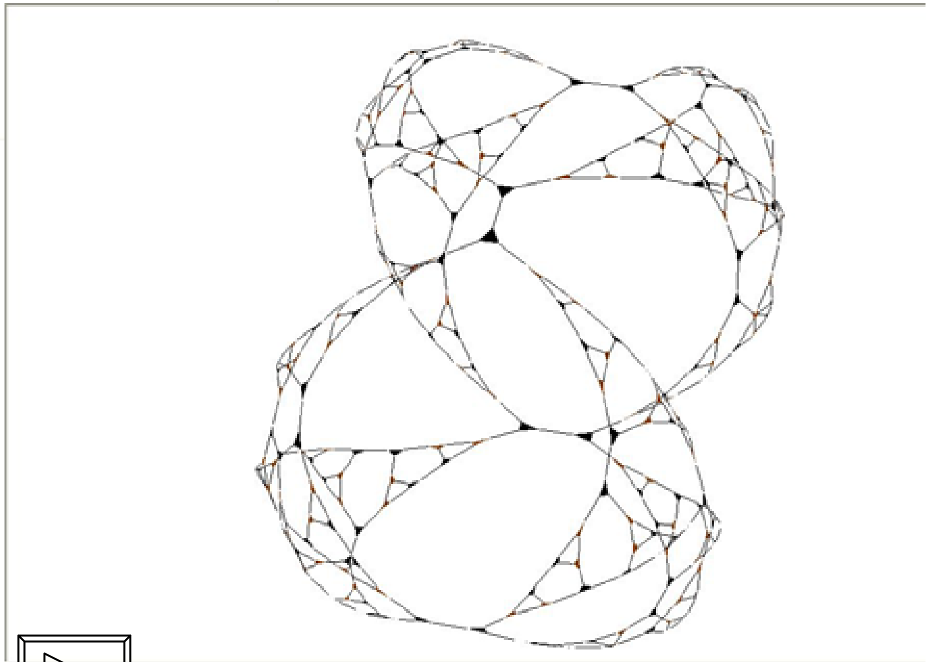
Nodes limit – simple case



Same state

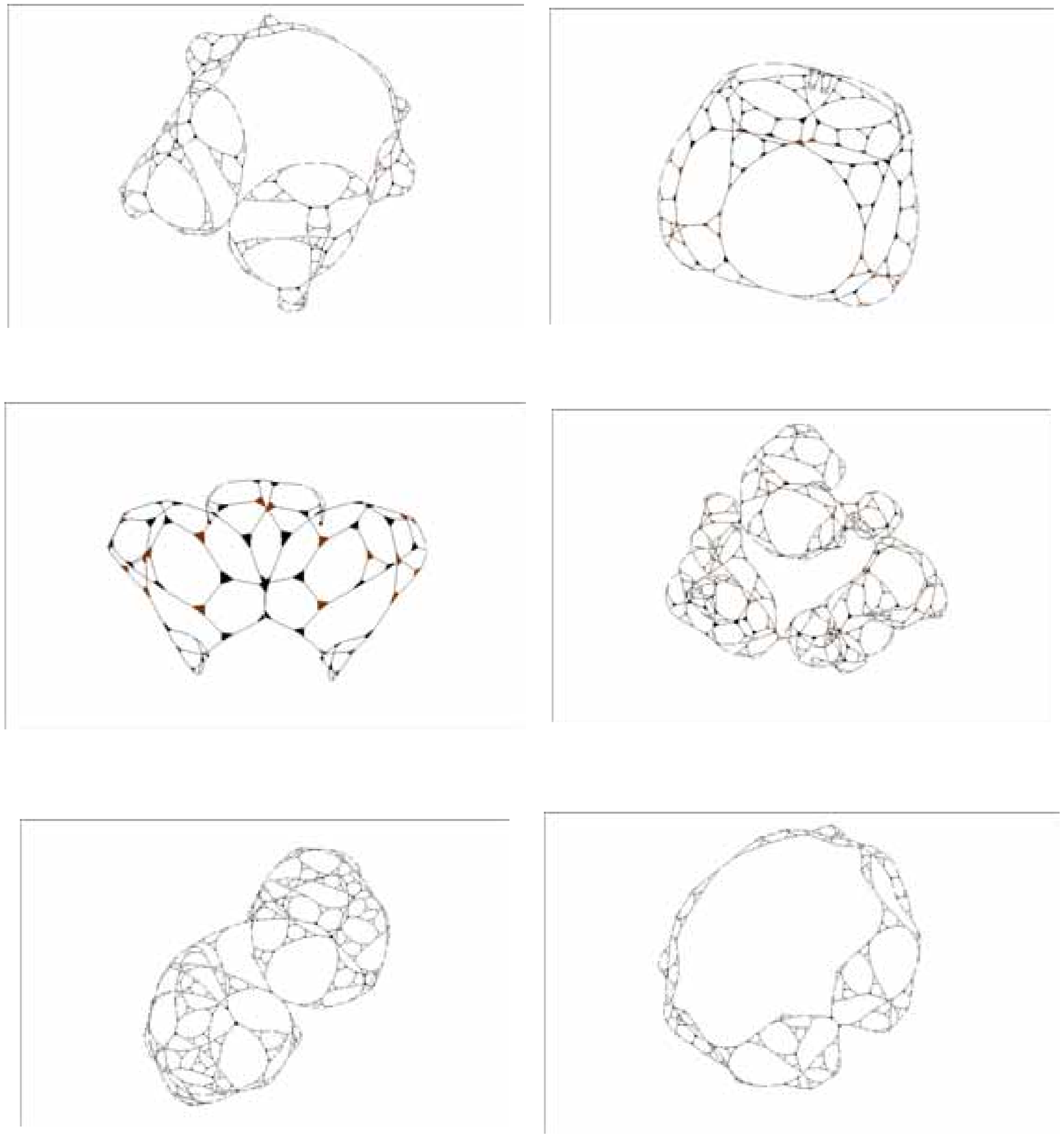
00001103 200 S2

State change
Less than twice

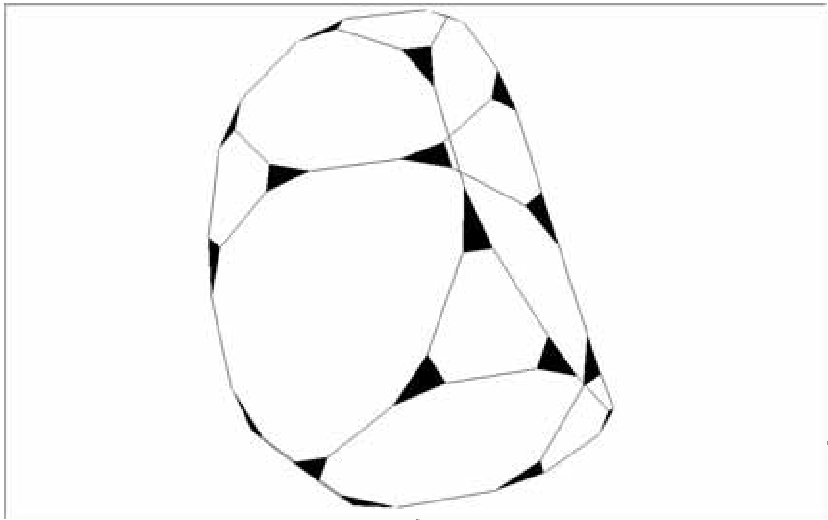


00001310 000 S2

Node limit – other cases



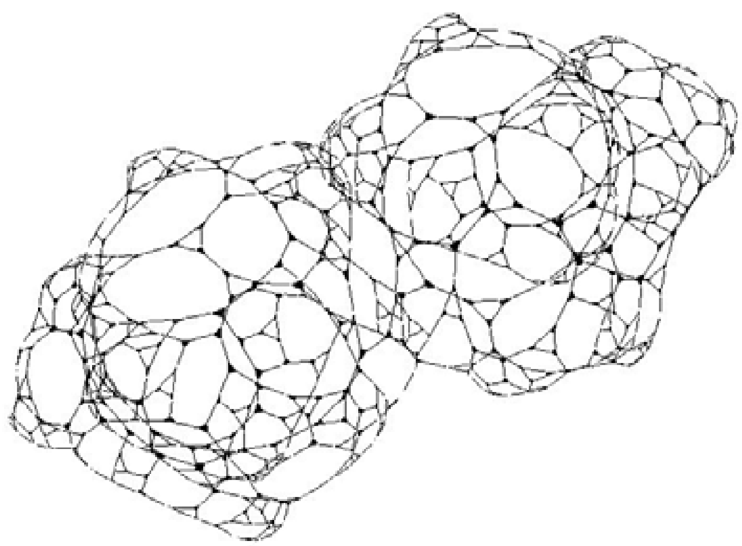
Halt



18 nodes

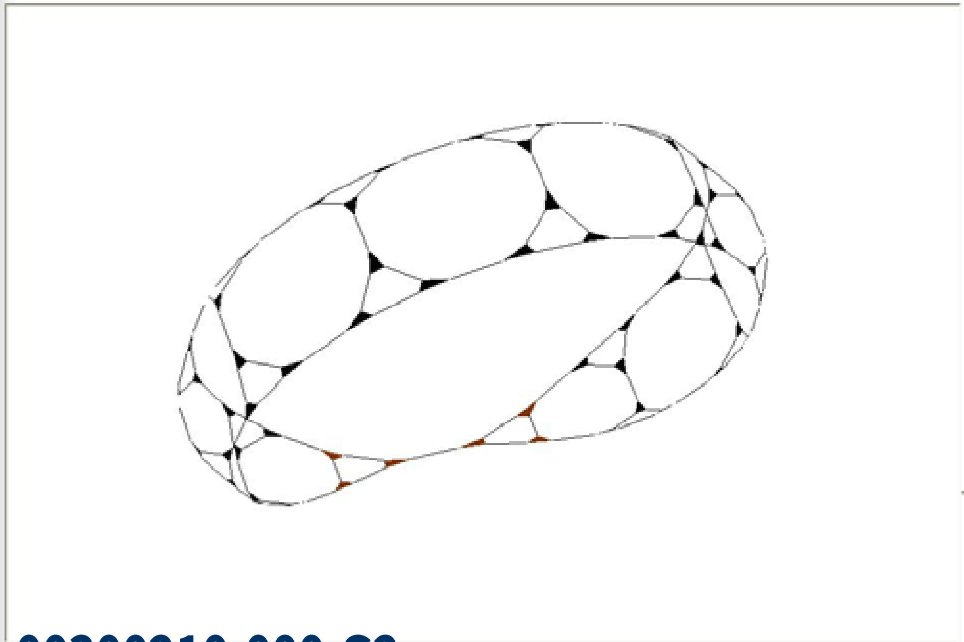
00000302 200 S2

612 nodes

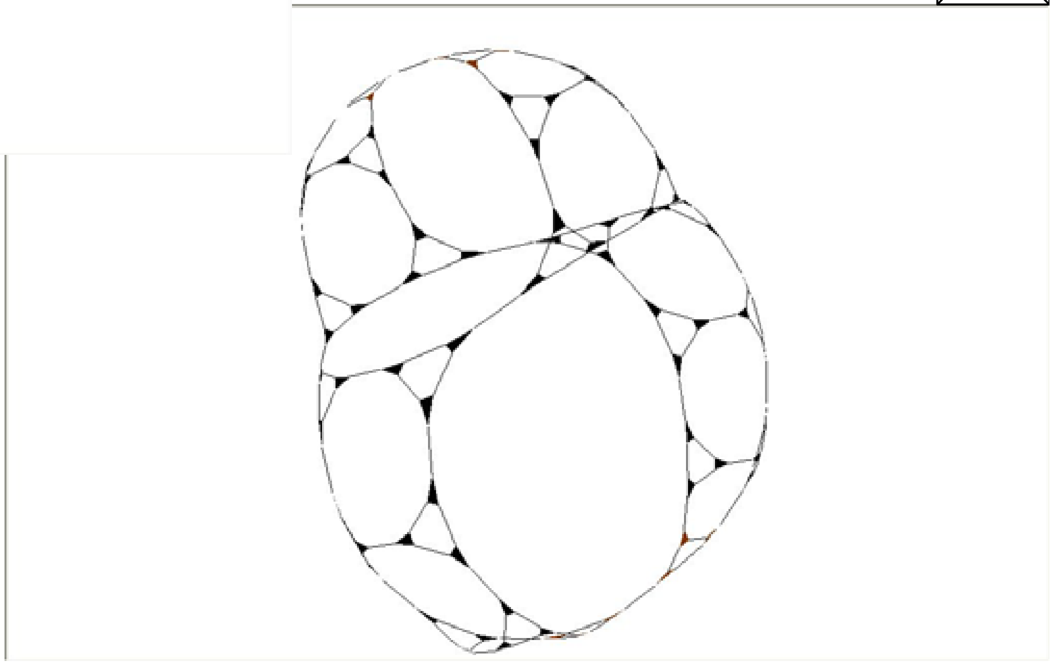


00230302 010 S2

Steps limit

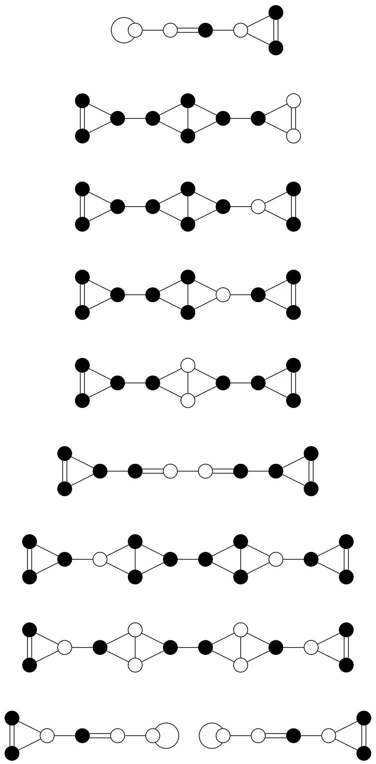


00300210 000 S2

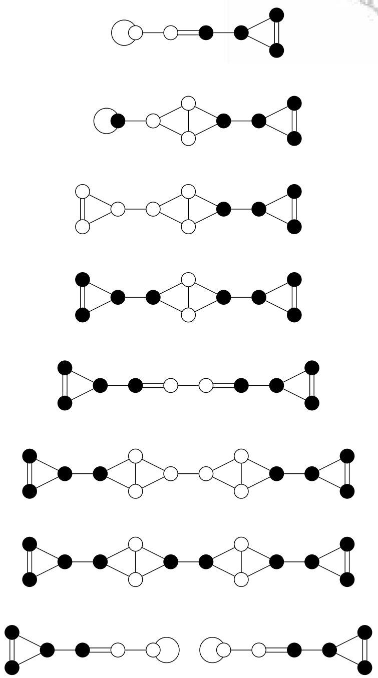


00300023 000 S2

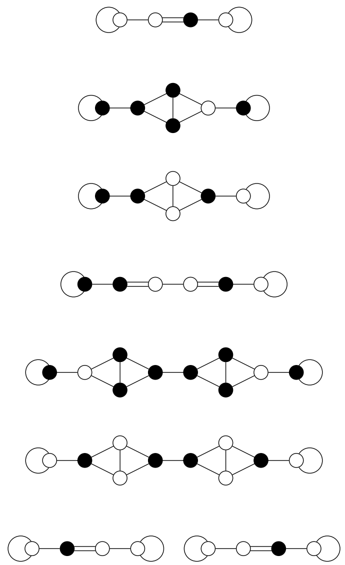
Self-replicating processes



01100202 201 S2

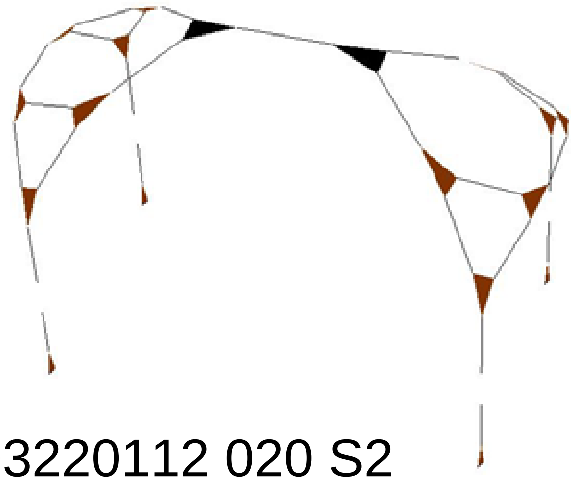


03002310 201 S1

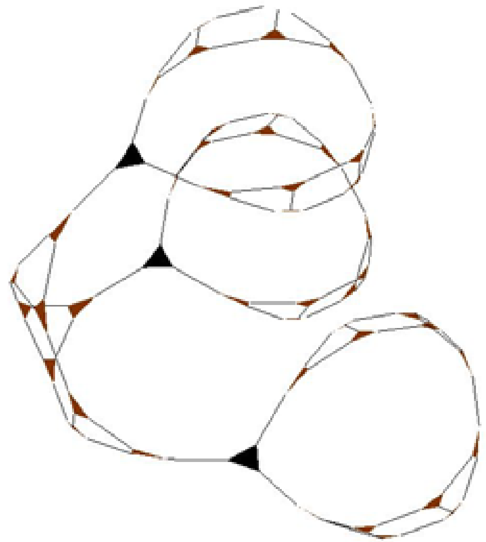


01110200 201 S2

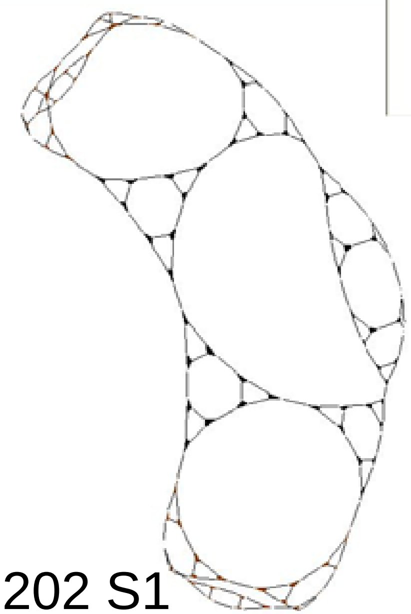
Other self-replicating patterns



03220112 020 S2



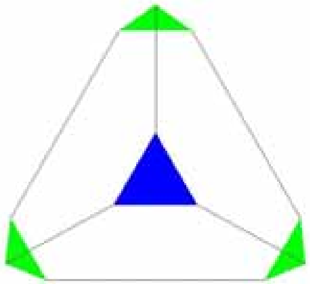
03120021 201 S1



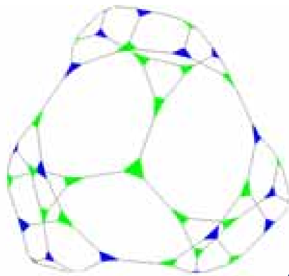
03010212 202 S1

01000232 010 S1

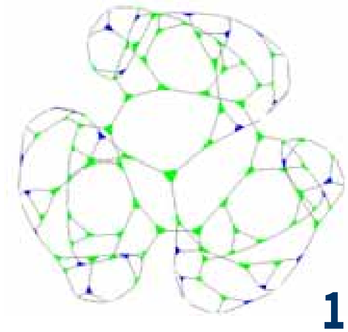
node size (step)



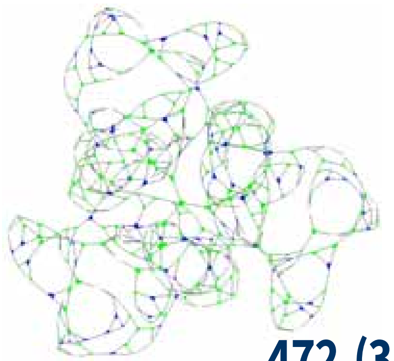
4 (0)



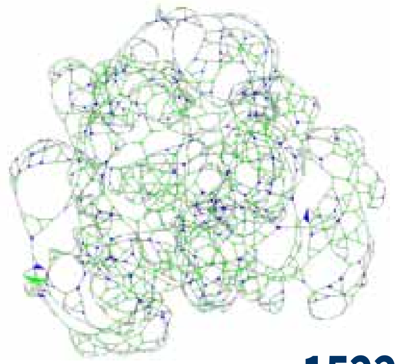
40 (10)



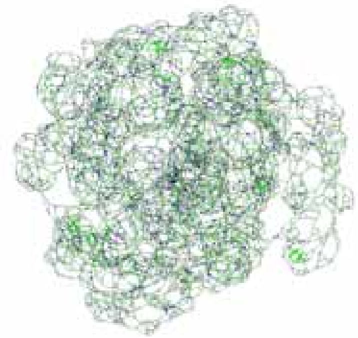
118 (24)



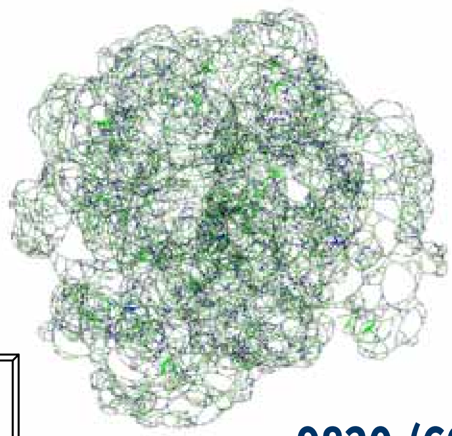
472 (38)



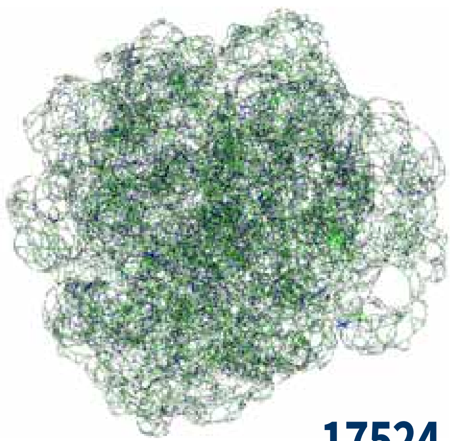
1522 (46)



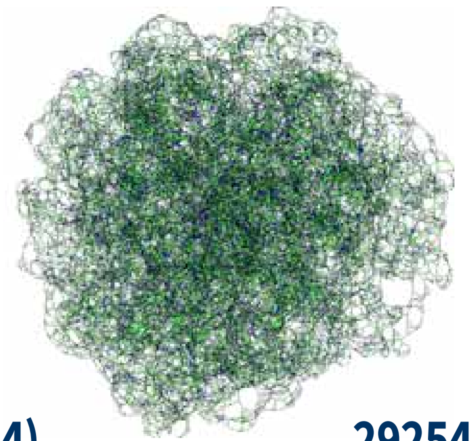
5770 (56)



9820 (60)



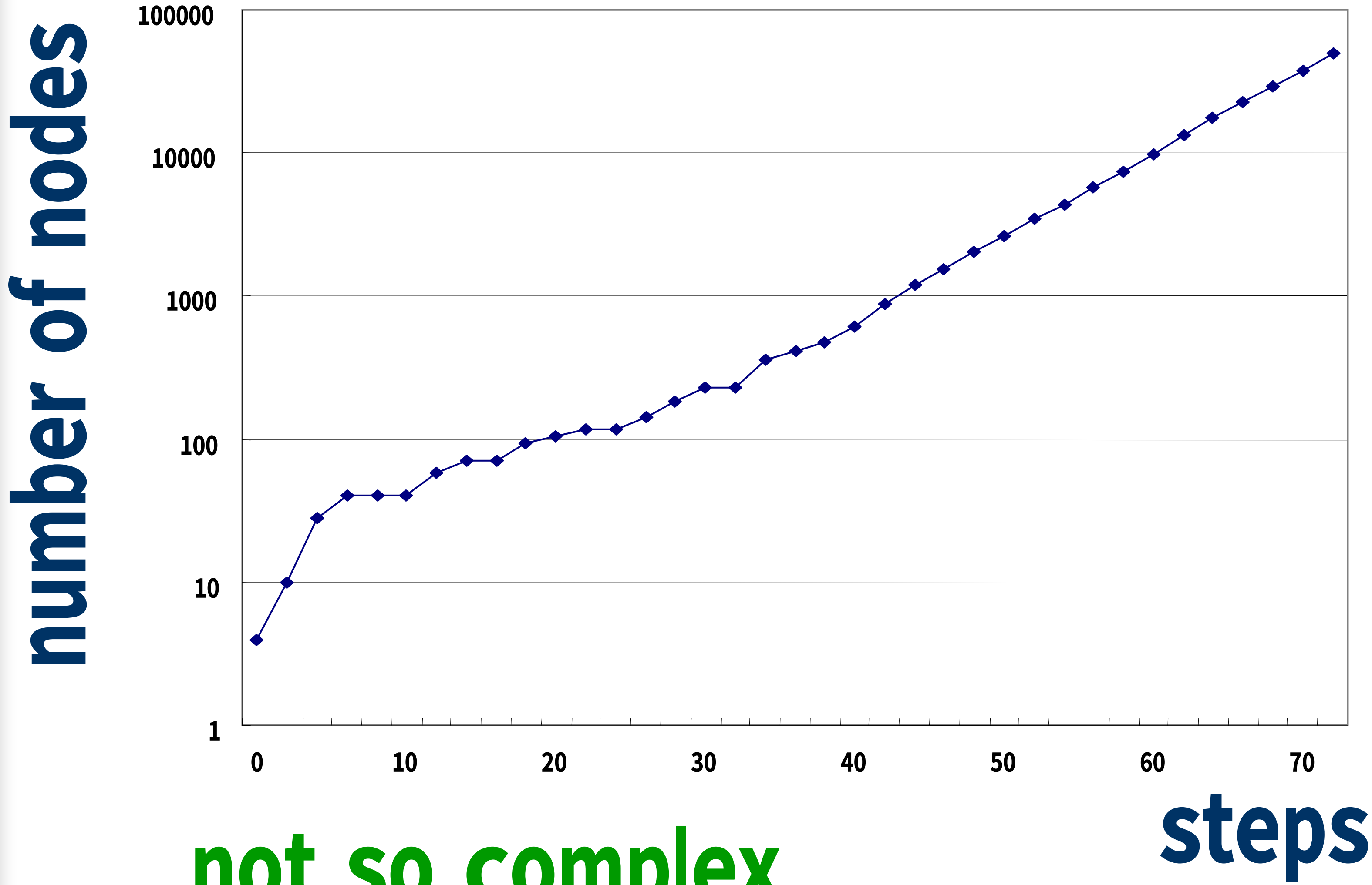
17524 (64)



29254 (68)

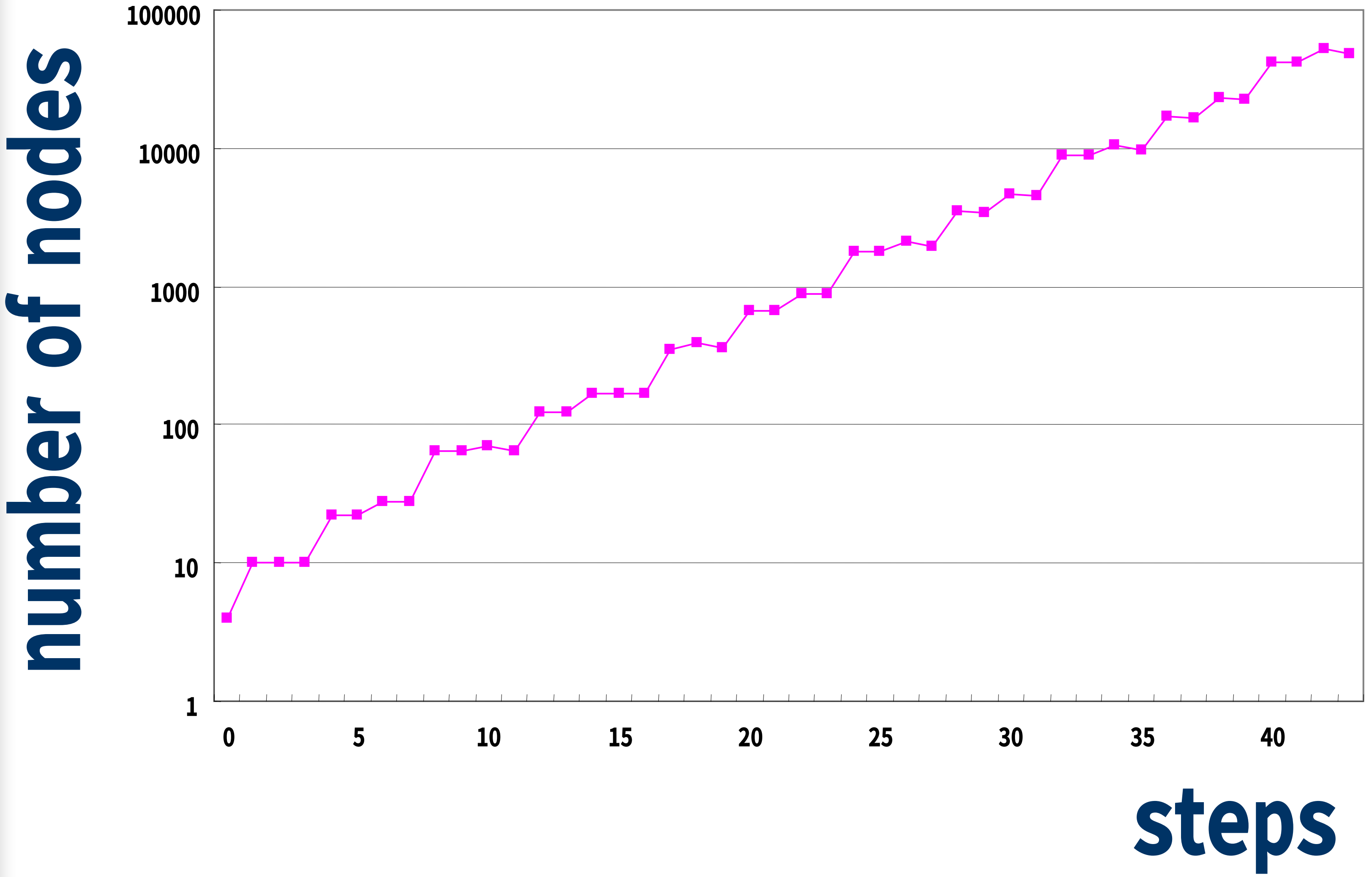
01000232 010 S1

(without annihilation)



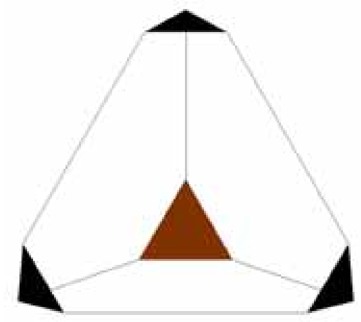
not so complex

03210010 102 S1

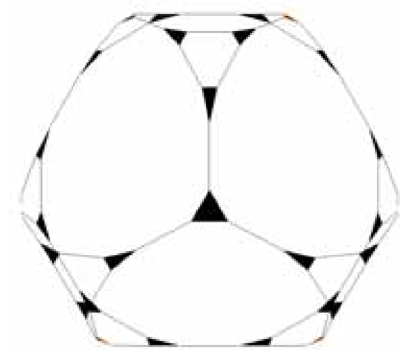


03210010 102 S1

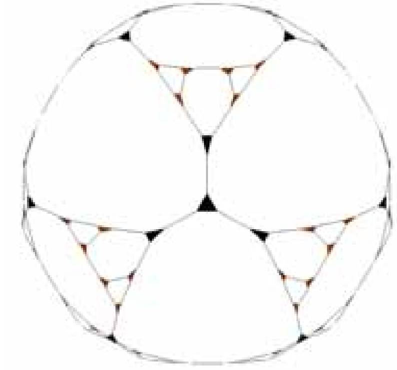
node size (step)



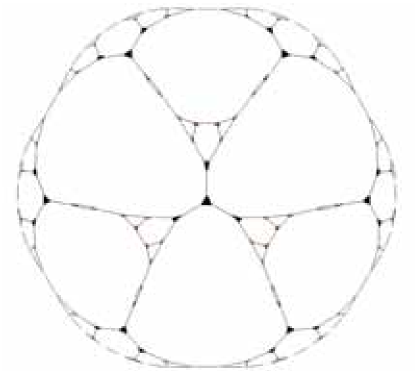
4 (0)



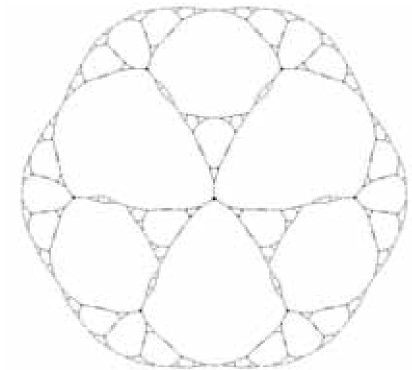
28 (6)



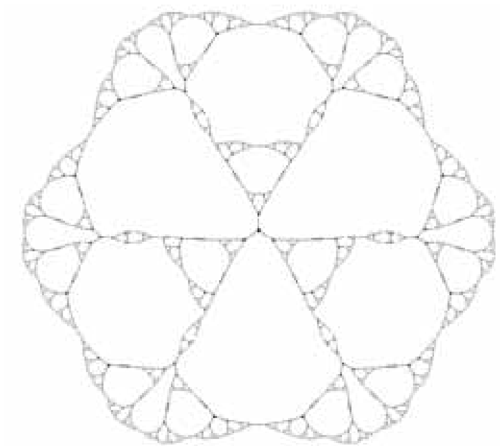
64 (11)



124 (13)



358 (19)



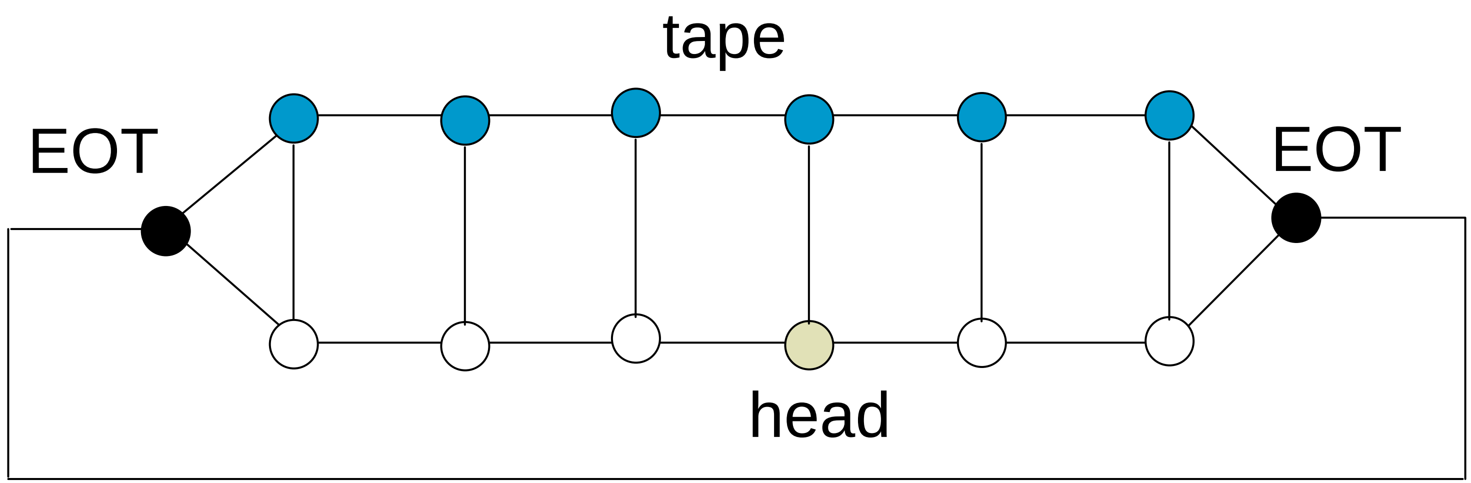
658 (21)

Programmability

- Using Many States
- Rule set design

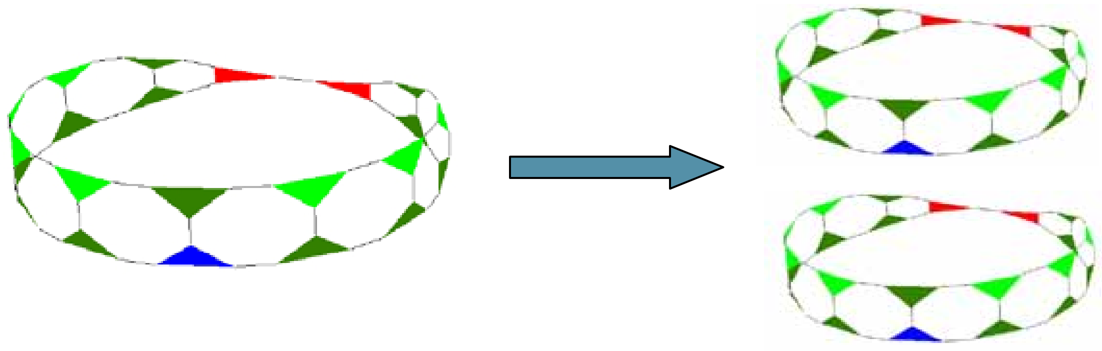
Turing machine

- 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

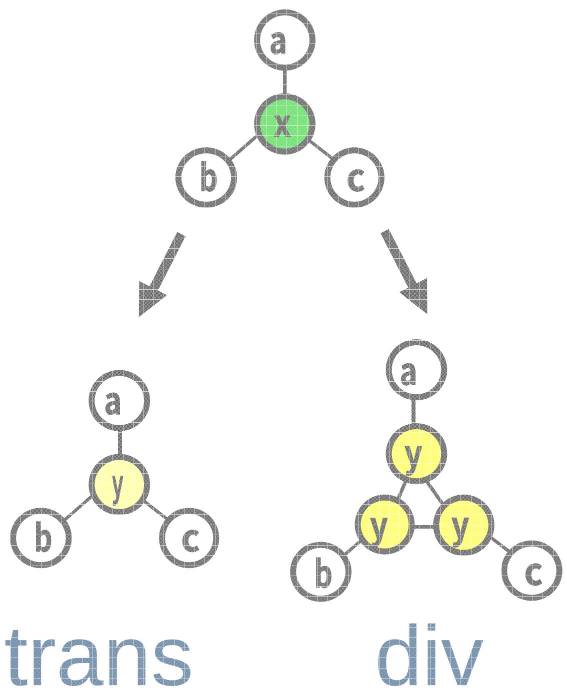
(1) Non-planar, many states
(2) Dual GA

(1) Non-planar graphs

- New link rule `swap'

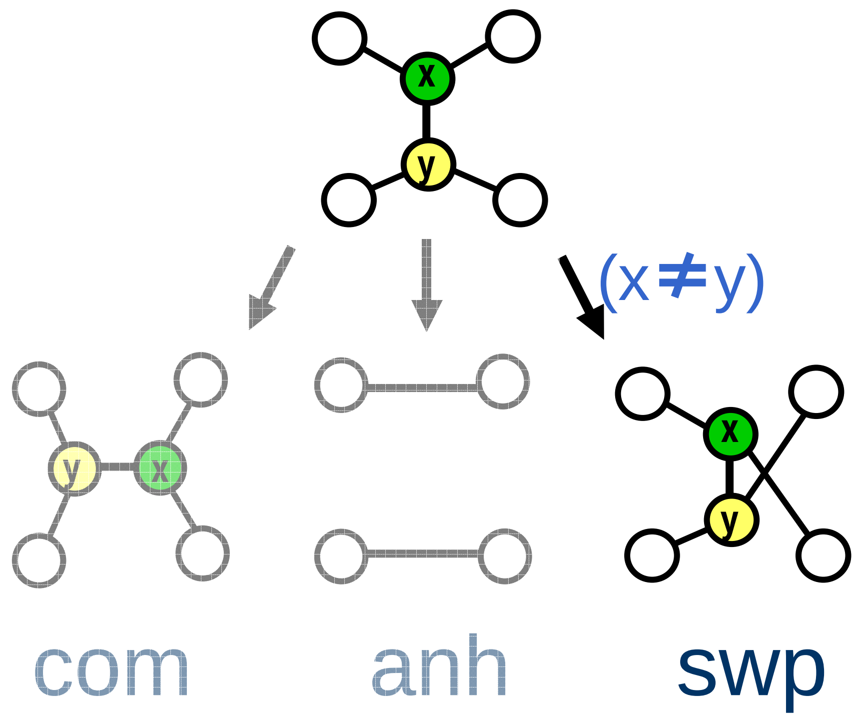
- Node rules

  rule x, (a, b, c) → y



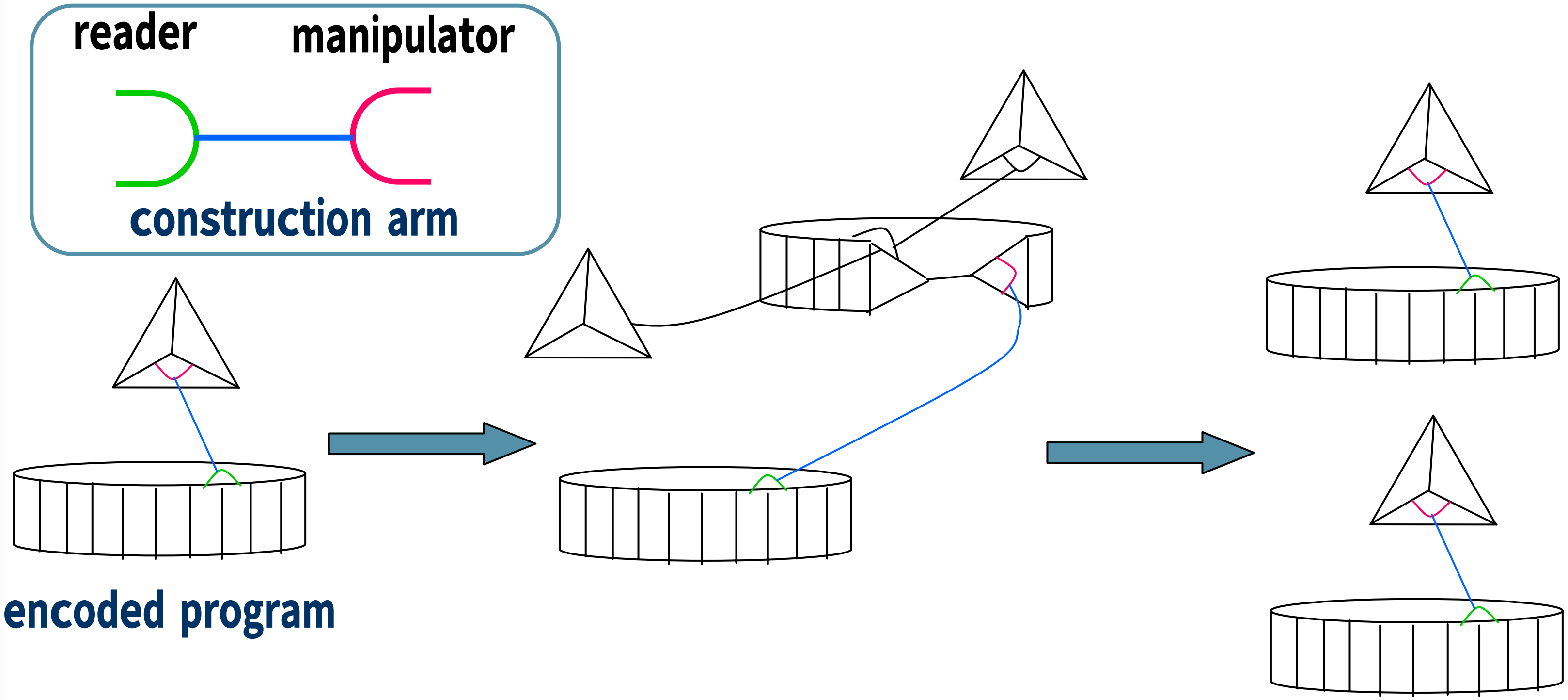
- Link rules

  rule (x,y)



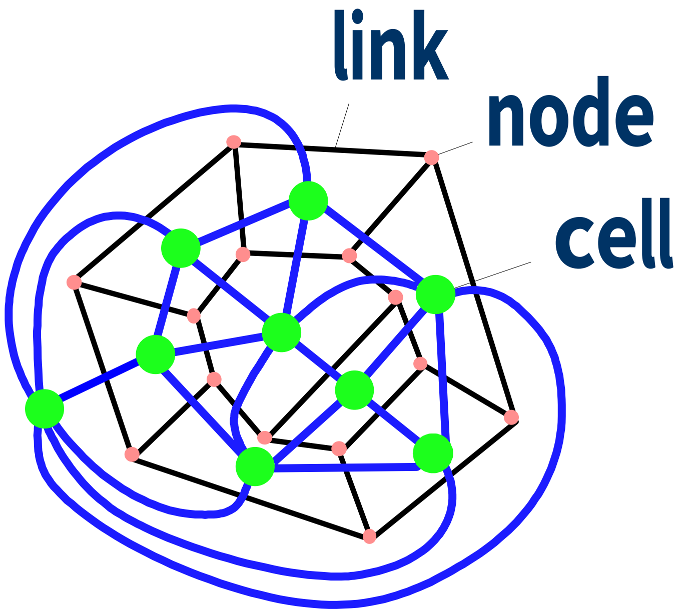
von Neumann style self-replication

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

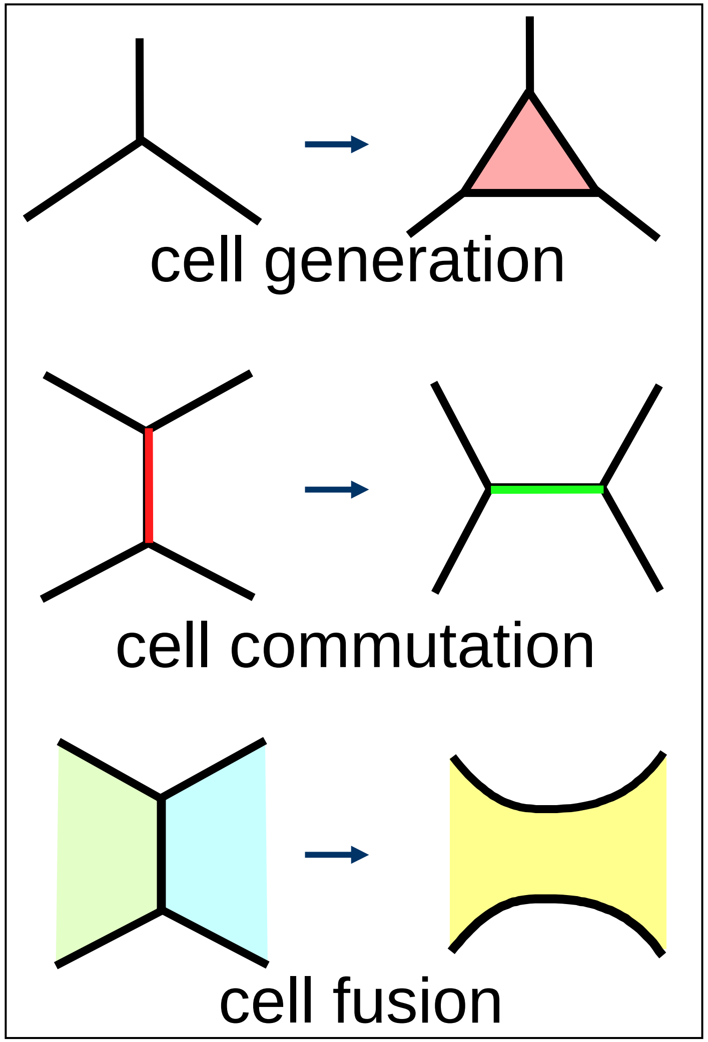


(2) Dual graph automata

Assign states to cells



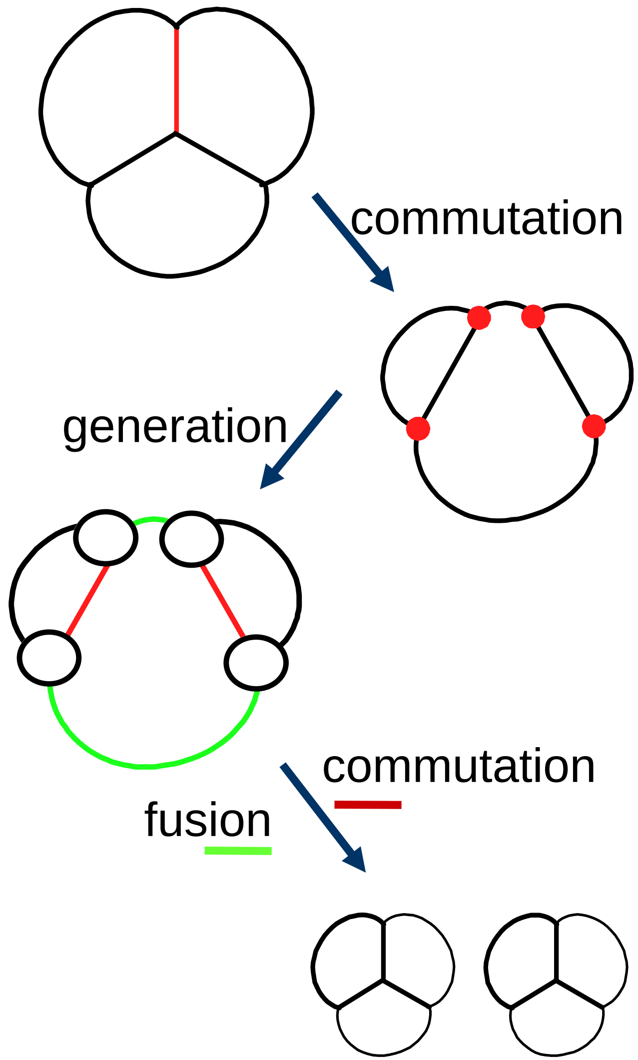
rewrite rules



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



Conclusions & Future Work

Conclusions

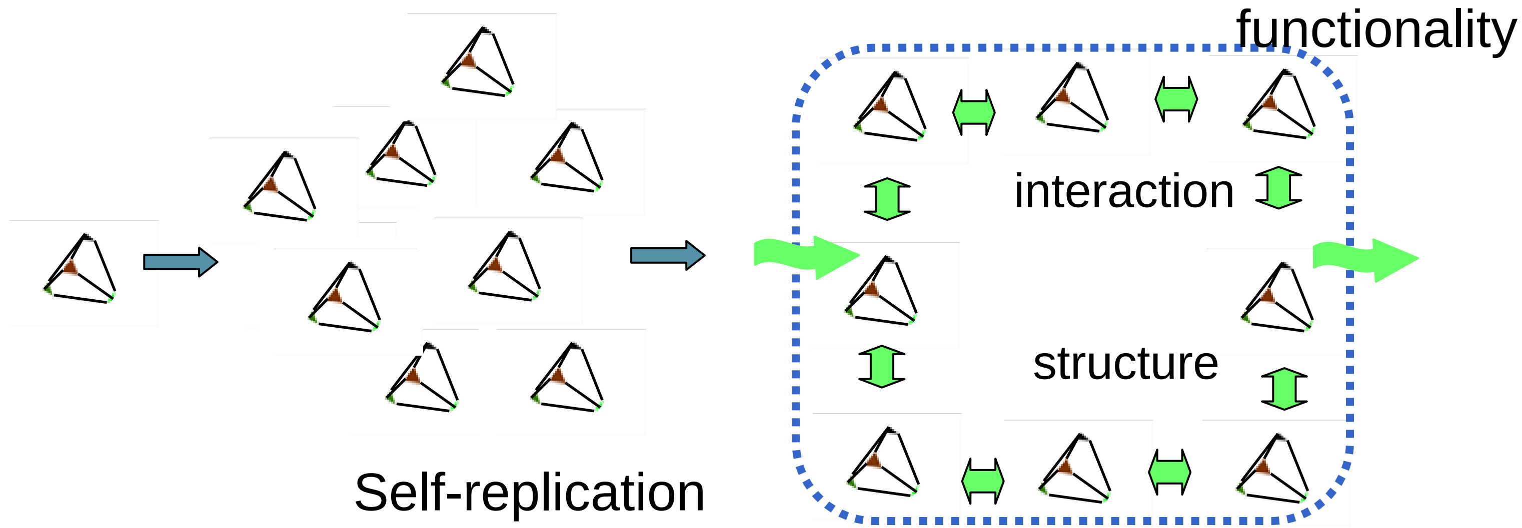
- Graph-rewriting automata
  - System that generates its boundary condition by itself
  - Not restricted to lattice space
    - Changing topology & number of nodes
- Programmability using many states
- Various development processes with 2 states
  - Self-replication
- Some variants (non-planar, dual)

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