Artificial Life
Summer 2025

Self Replication
Langton's Loop
Lindenmayer Systems

Master Computer Science [MA-INF 4201]
Mon 14c.t. – 15:45, HS-2

Dr. Nils Goerke, Autonomous Intelligent Systems,
Department of Computer Science, University of Bonn

Overview:

- Self-Replication
- Langton's Self-Replicating Loop
- Lindenmayer Systems

Overview:

- **Self-Replication**
- Langton's Self-Replicating Loop
- Lindenmayer Systems

Self Replication

One of the fundamental properties of biological living systems is the capability of **reproduction**.
For artificial (life) systems the equivalent capability is called **replication.**

Self Replication

One of the fundamental properties of biological living systems is the capability of **reproduction**.
For artificial (life) systems the equivalent capability is called **replication.**

John von Neumann was fascinated from the idea to build a machinery that would be capable of reproducing itself, (or replicating itself).
His work on 2 dimensional Cellular Automata was, in part, inspired by this idea.

Self Replication

One of the fundamental properties of biological living systems is the capability of **reproduction**.
For artificial (life) systems the equivalent capability is called **replication.**

John von Neumann was fascinated from the idea to build a machinery that would be capable of reproducing itself, (or replicating itself).
His work on 2 dimensional Cellular Automata was, in part, inspired by this idea.

As a direct result he developed (1940) the famous **von Neumann's Universal Constructor**.

John von Neumanns Self-Replicating Automaton

John von Neumann's universal constructor is a machinery, based on a rectangular grid CA.

The idea was to define a system that is
universal with respect to computation capabilities and
universal with respect to construction.

Thus, a system that could construct anything, should be capable of constructing a copy of itself: replication.

John von Neumanns Self-Replicating Automaton

John von Neumann's universal constructor is a machinery, based on a rectangular grid CA.

The idea was to define a system that is
universal with respect to computation capabilities and
universal with respect to construction.

Thus, a system that could construct anything, should be capable of constructing a copy of itself: replication.

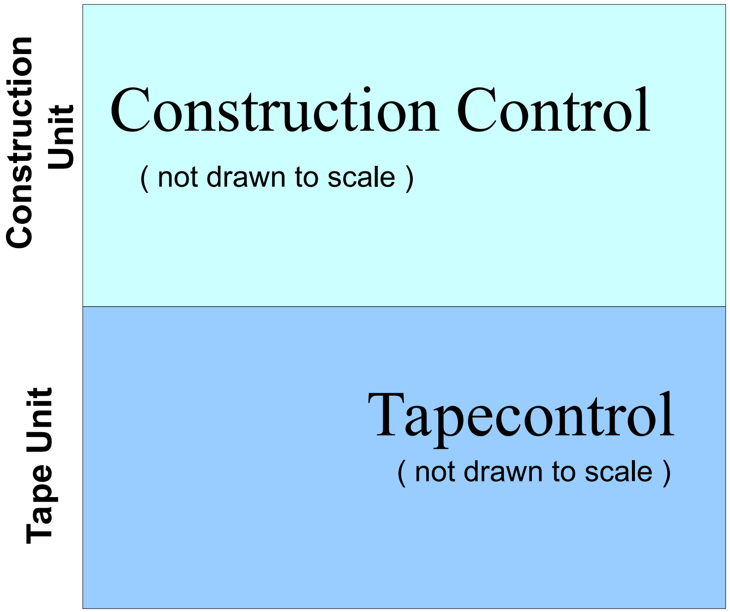
Extra homework, (a Quine):
Write a program, that prints out it's own source code!

John von Neumanns Self-Replicating Automaton

John von Neumann designed an artificial creature with the capability to re-produce itself. Therefore he proposed some basic requirements for such a machinery:

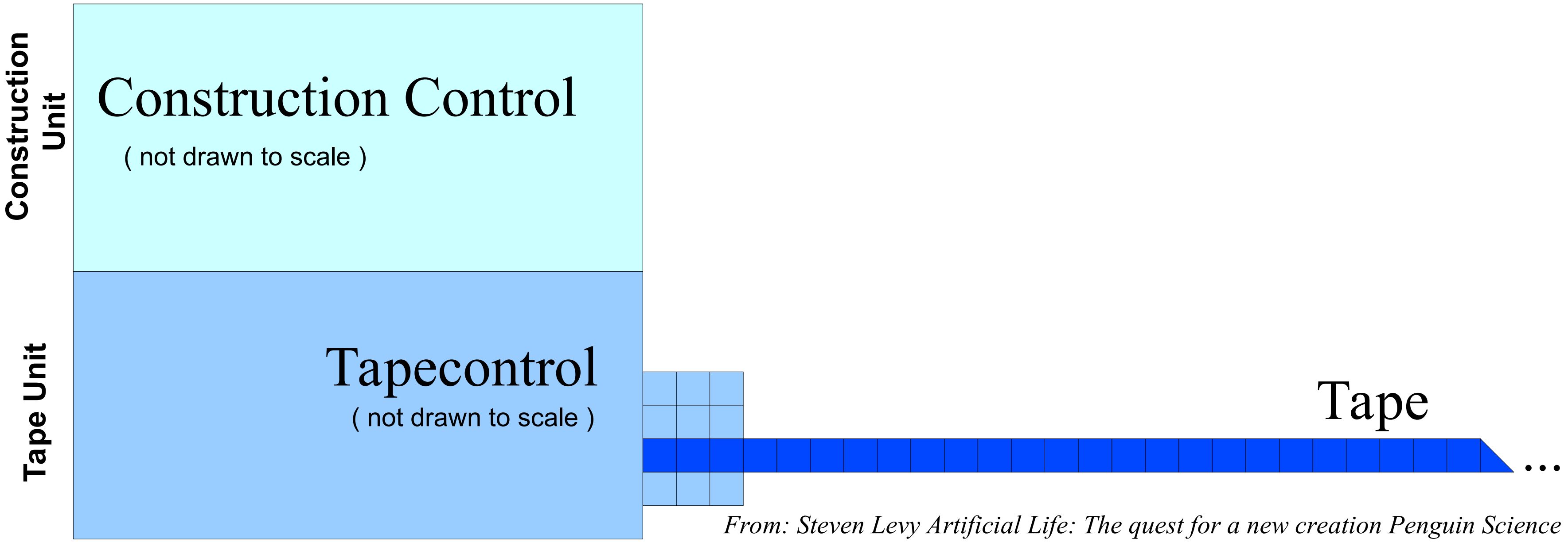
- Several computational elements.
- A manipulating element (like a hand).
- A cutting element, capable of disconnecting elements.
- A fusing element to connect two parts.
- A sensing element, which could recognize parts.
- “*Girders*”, rigid structural elements (building blocks) that build the chassis and the information carrier.

John von Neumanns Self-Replicating Automaton



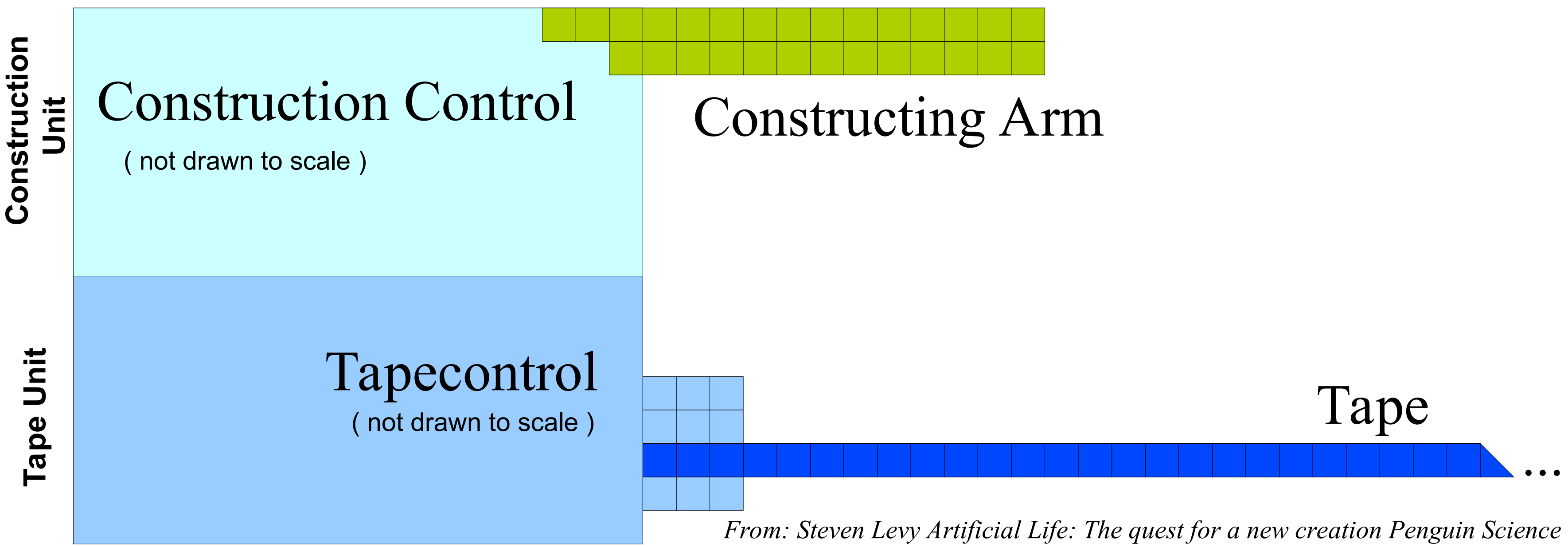
From: Steven Levy Artificial Life: The quest for a new creation Penguin Science 1993
from: Arthur Burks, Essays on Cellular Automata, University of Illinois Press, 1970

John von Neumanns Self-Replicating Automaton



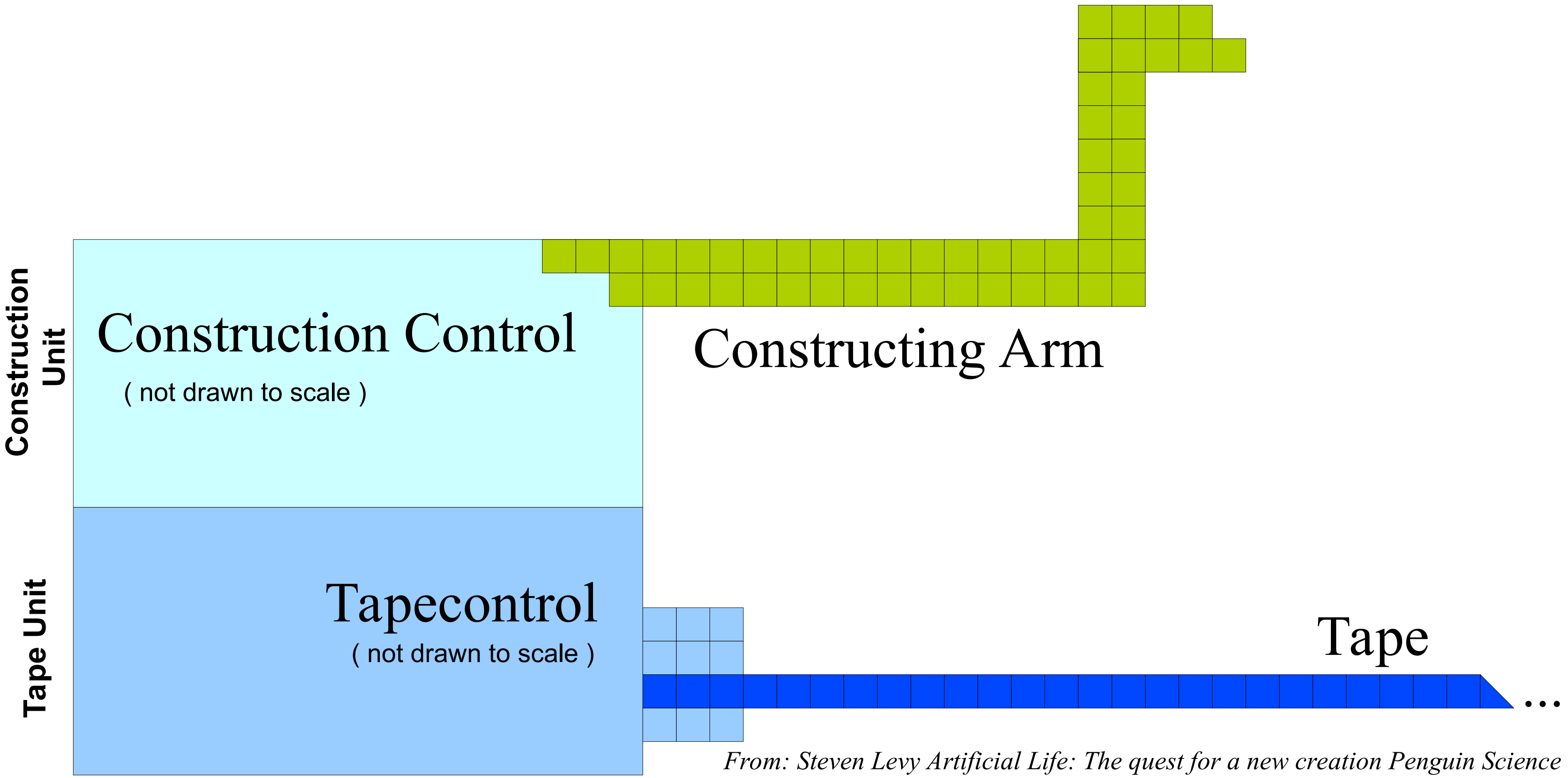
From: Steven Levy Artificial Life: The quest for a new creation Penguin Science 1993
from: Arthur Burks, Essays on Cellular Automata, University of Illinois Press, 1970

John von Neumanns Self-Replicating Automaton



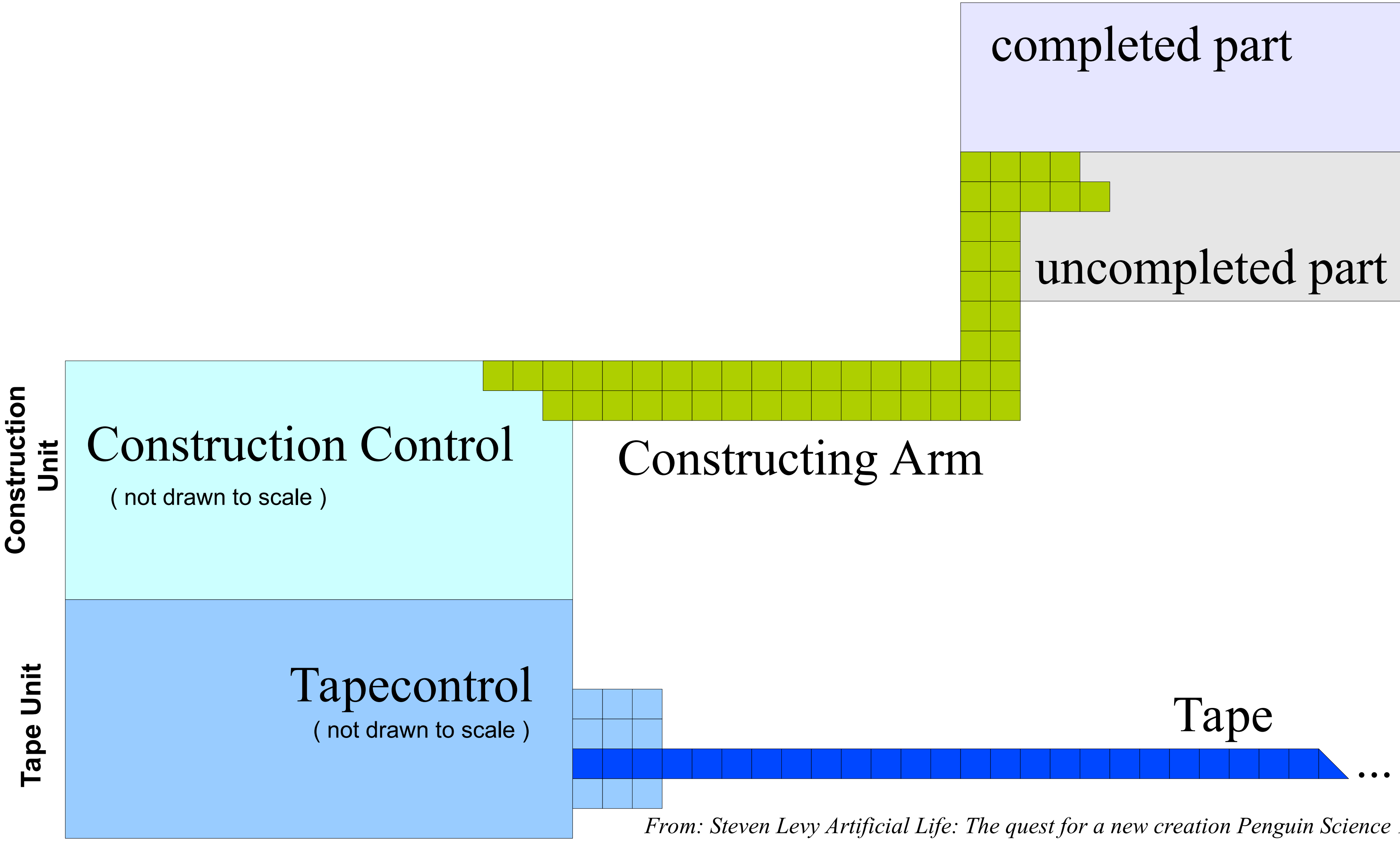
From: Steven Levy Artificial Life: The quest for a new creation Penguin Science 1993
from: Arthur Burks, Essays on Cellular Automata, University of Illinois Press, 1970

John von Neumanns Self-Replicating Automaton



From: Steven Levy Artificial Life: The quest for a new creation Penguin Science 1993
from: Arthur Burks, Essays on Cellular Automata, University of Illinois Press, 1970

John von Neumanns Self-Replicating Automaton



From: Steven Levy Artificial Life: The quest for a new creation Penguin Science 1993
from: Arthur Burks, Essays on Cellular Automata, University of Illinois Press, 1970

John von Neumanns Self-Replicating Automaton

John von Neumann's self-replicating automaton:

- “lives “ on an (virtually) infinite, rectangular grid
- with unlimited supply of elements,
- has 29 different states for the elements (cells),
- has a construction unit,
- has a construction arm (hand, cutting, fusing, sensing),
- has a tape unit,
- has an (infinite) tape,
- and would consist of approx 150000 elements.

John von Neumanns Self-Replicating Automaton

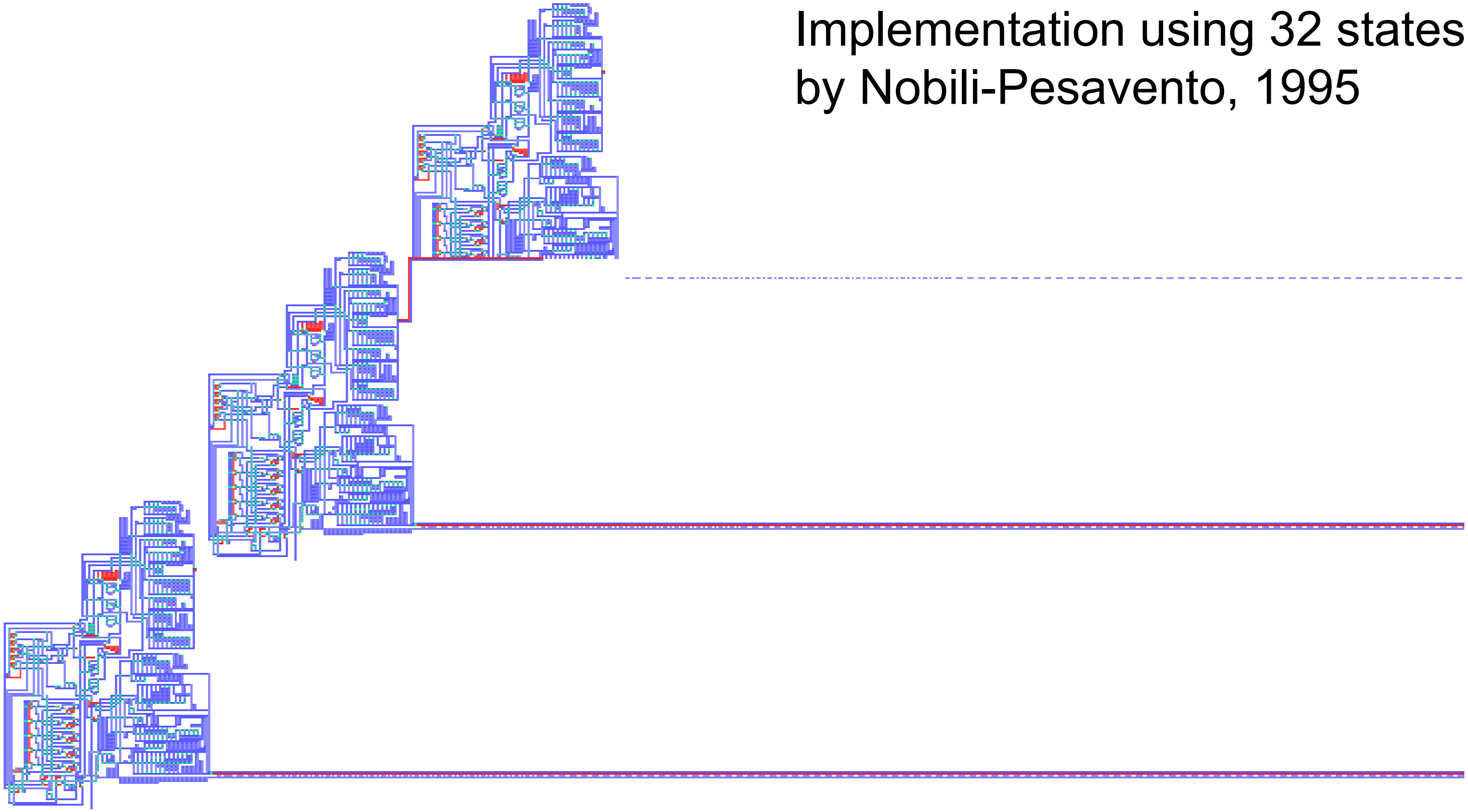
„Von Neumann's specification defined the machine as using 29 states, these states constituting means of signal carriage and logical operation, and acting upon signals represented as bit streams.

A 'tape' of cells encodes the sequence of actions to be performed by the machine.
Using a writing head (termed a construction arm) the machine can print out (construct) a new pattern of cells, allowing it to make a complete copy of itself, and the tape."

From: *http://en.wikipedia.org/wiki/Von_Neumann_Universal_Constructor*

John von Neumanns Self-Replicating Automaton

Implementation using 32 states by Nobili-Pesavento, 1995



From: *http://en.wikipedia.org/wiki/Von_Neumann_Universal_Constructor*

Overview:

- Self-Replication
- **Langton's Self-Replicating Loop**
- Lindenmayer Systems

Langton's Self-Replicating Loop

Chris Langton is one of the scientists directly connected with the beginning of Artificial Life as a scientific subject.

He became famous:

- for organizing the First Conference on Artificial Life in 1987
- for his work on a CA based self-replicating structure that is called Langton's Loop.
- for his measure λ on complexity and
- for his simple Turing machine called Langton's Ant.

Langton's Self-Replicating Loop

Langton's Self-Replicating Loop:
is a 2-dim **Cellular Automaton,**
defined by a **Rule table** and
a special **Starting Configuration** of cells set:

The iteration of this CA is changing the initial configuration with respect to the rule defined.
This spatio-temporal development will change the cells of the CA grid in such a way, that after some iteration steps a second structure with exactly the same shape, and thus the same capabilities, will arise; the loop has replicated.

Langton's Self-Replicating Loop

Langton's Self-Replicating Loop
is a 2-dim **Cellular Automaton**,
defined by a **Rule table** and
a special **Starting Configuration** of cells set:

d=2, CA in 2-dim, rectangular grid

r=1, von Neumann Neighborhood, with $n = 4r+1 = 5$

k=8, 8 states 0-7, with silent state 0

only **219** entries out of possible $q = k^n = 8^5 = 32768$
in the rule table **don't yield** the silent state **0**

86 cells are set in the starting configuration,

The entire loop replicates after **151** time steps.

Langton's Self-Replicating Loop

Langton's Self-Replicating Loop consists of:

Langton's Self-Replicating Loop

Langton's Self-Replicating Loop consists of:

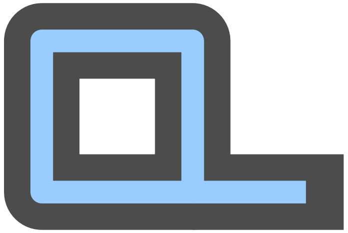
a square shaped body, the loop,
and a construction arm.

Langton's Self-Replicating Loop

Langton's Self-Replicating Loop consists of:

a square shaped body, the loop,
and a construction arm.

The loop and the arm comprise of
a channel that is covered by a sheath.

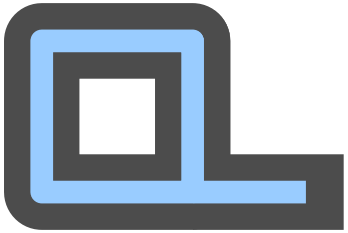


Langton's Self-Replicating Loop

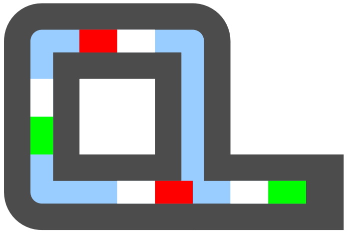
Langton's Self-Replicating Loop consists of:

a square shaped body, the loop,
and a construction arm.

The loop and the arm comprise of
a channel that is covered by a sheath.

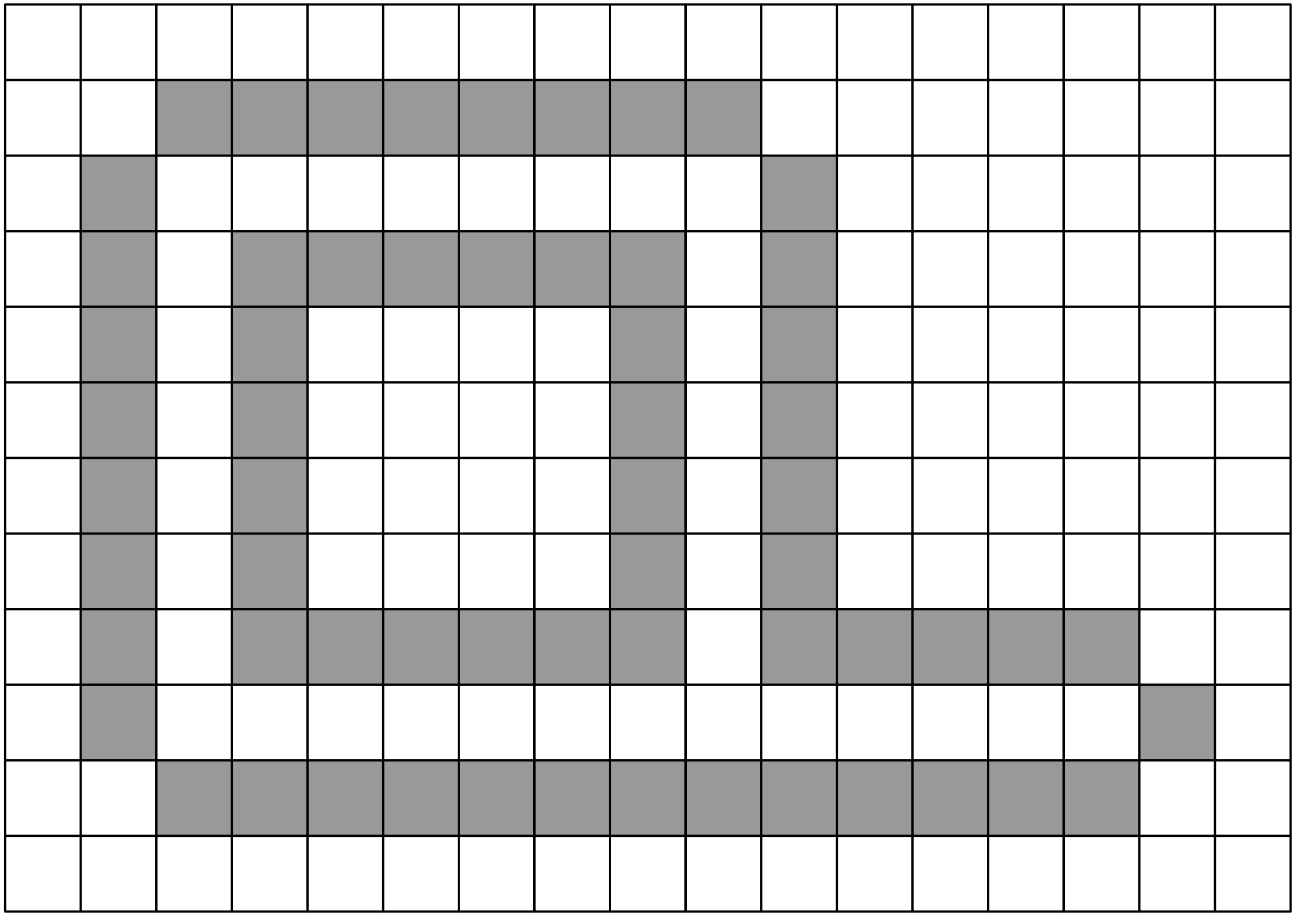


Inside the channel is a sequence,
or string of messages that control
the reproduction process
(together with the underlying CA rule).



Langton's Self-Replicating Loop

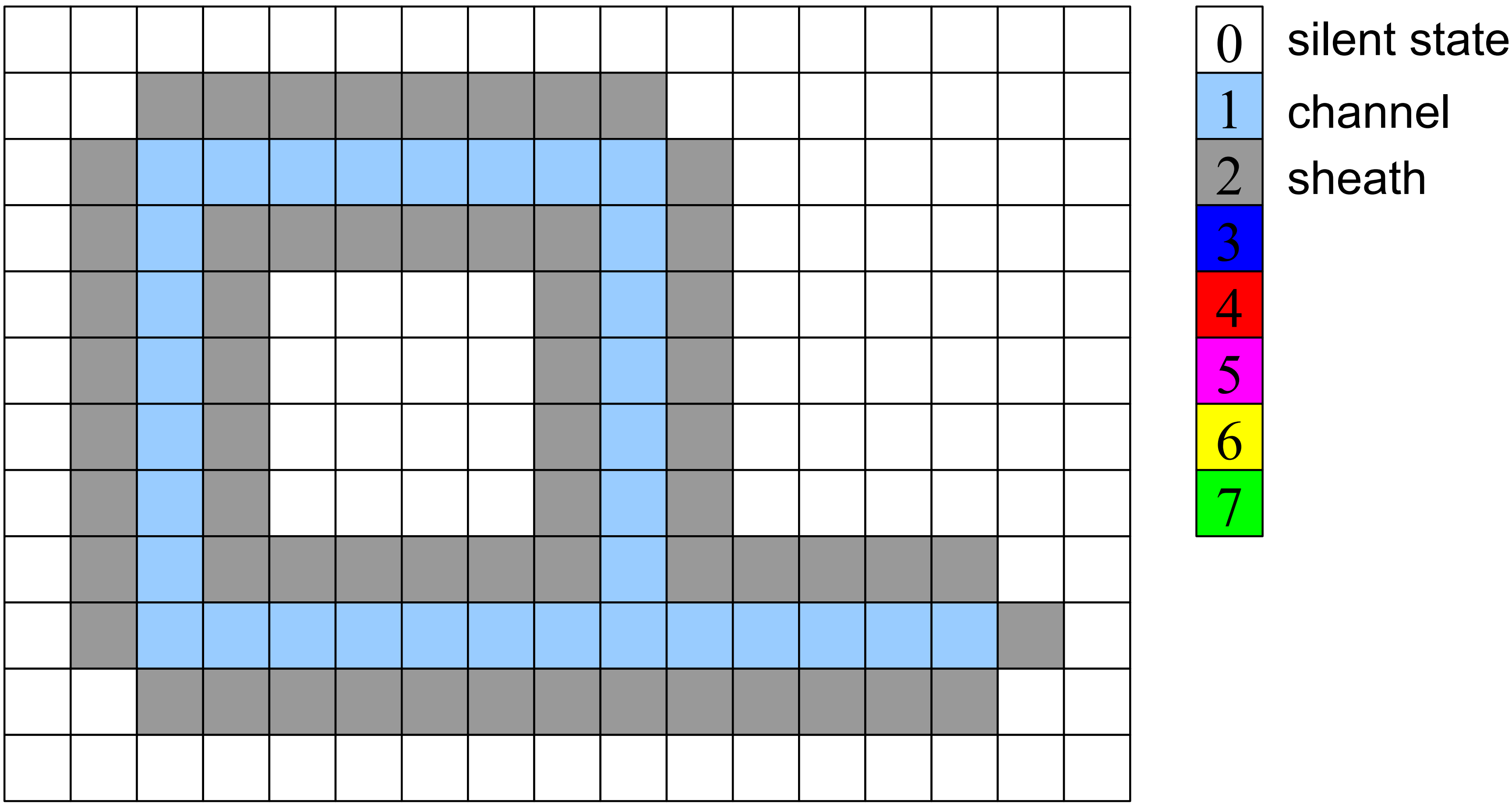
CA, d=2, von Neumann neighborhood, r=1, k=8, *Christopher Langton 1984*



0	silent state
1	
2	sheath
3	
4	
5	
6	
7	

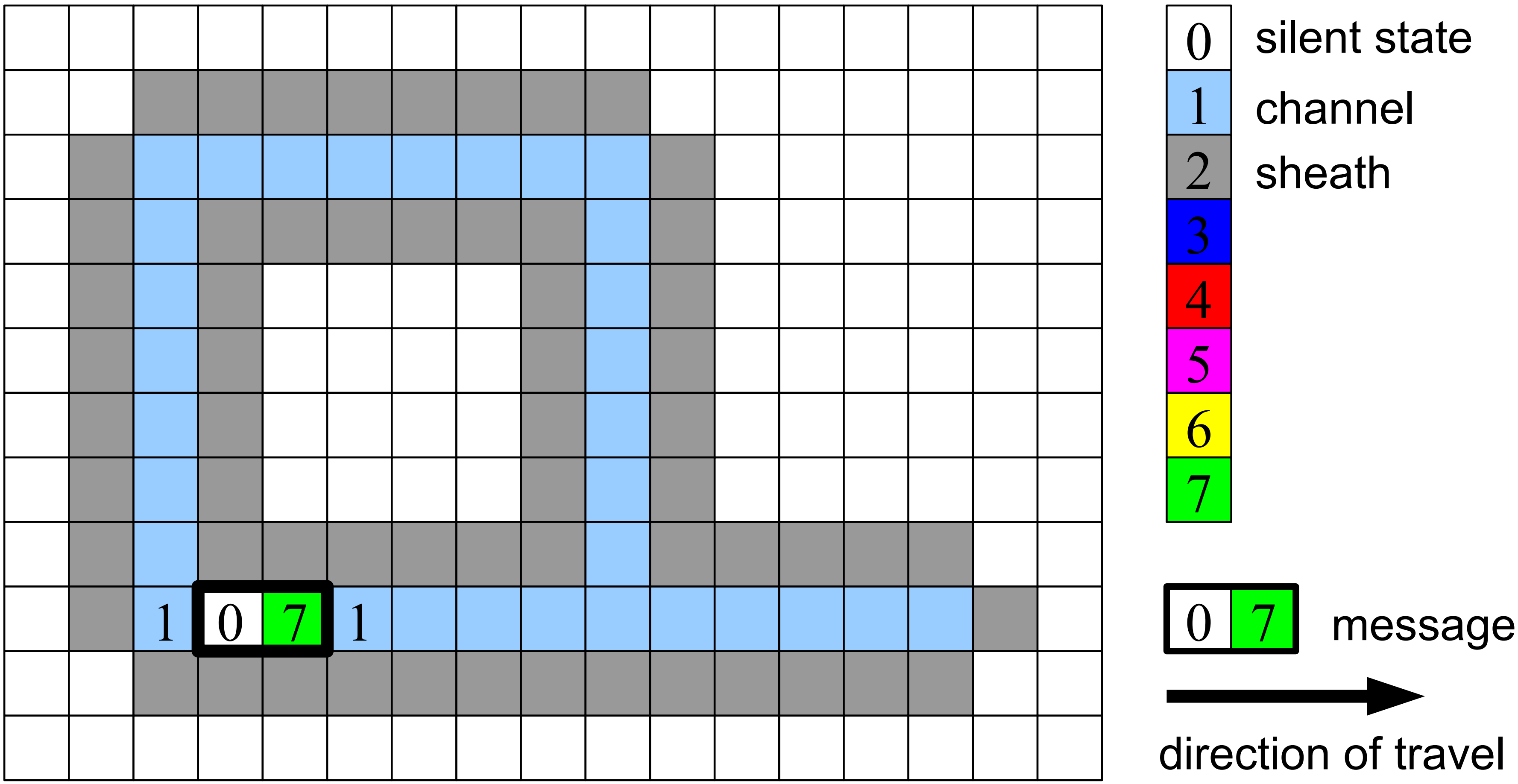
Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984



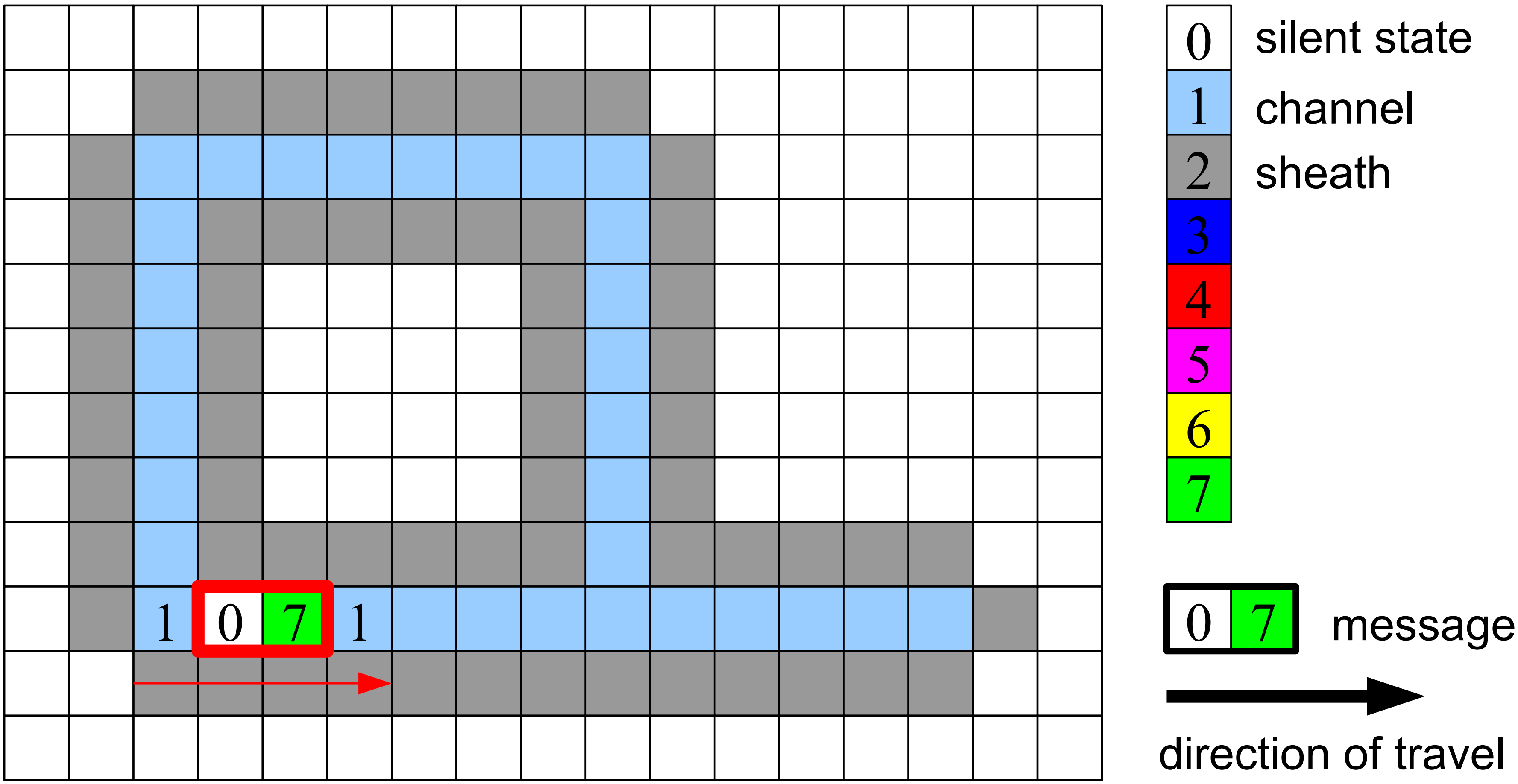
Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984



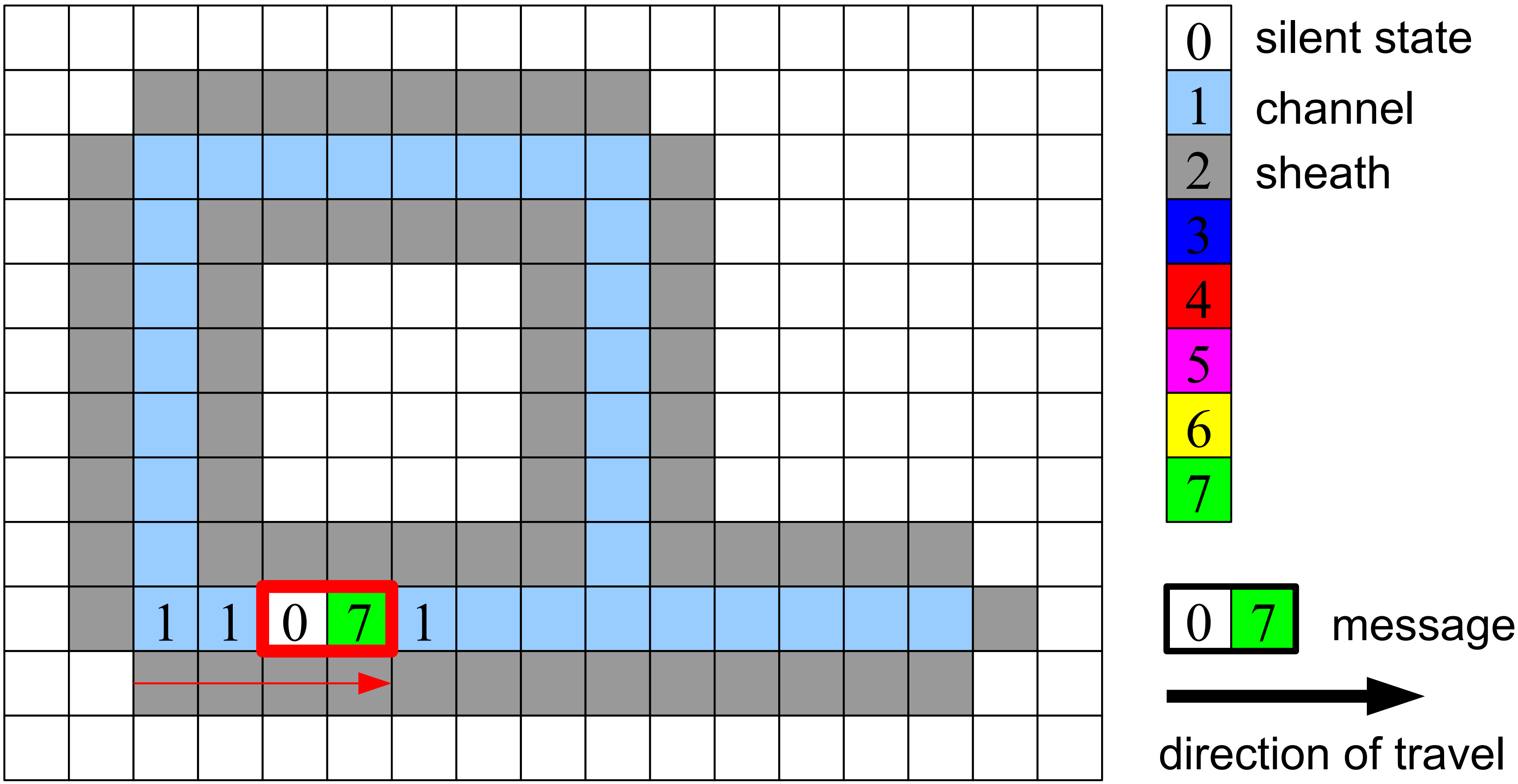
Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984



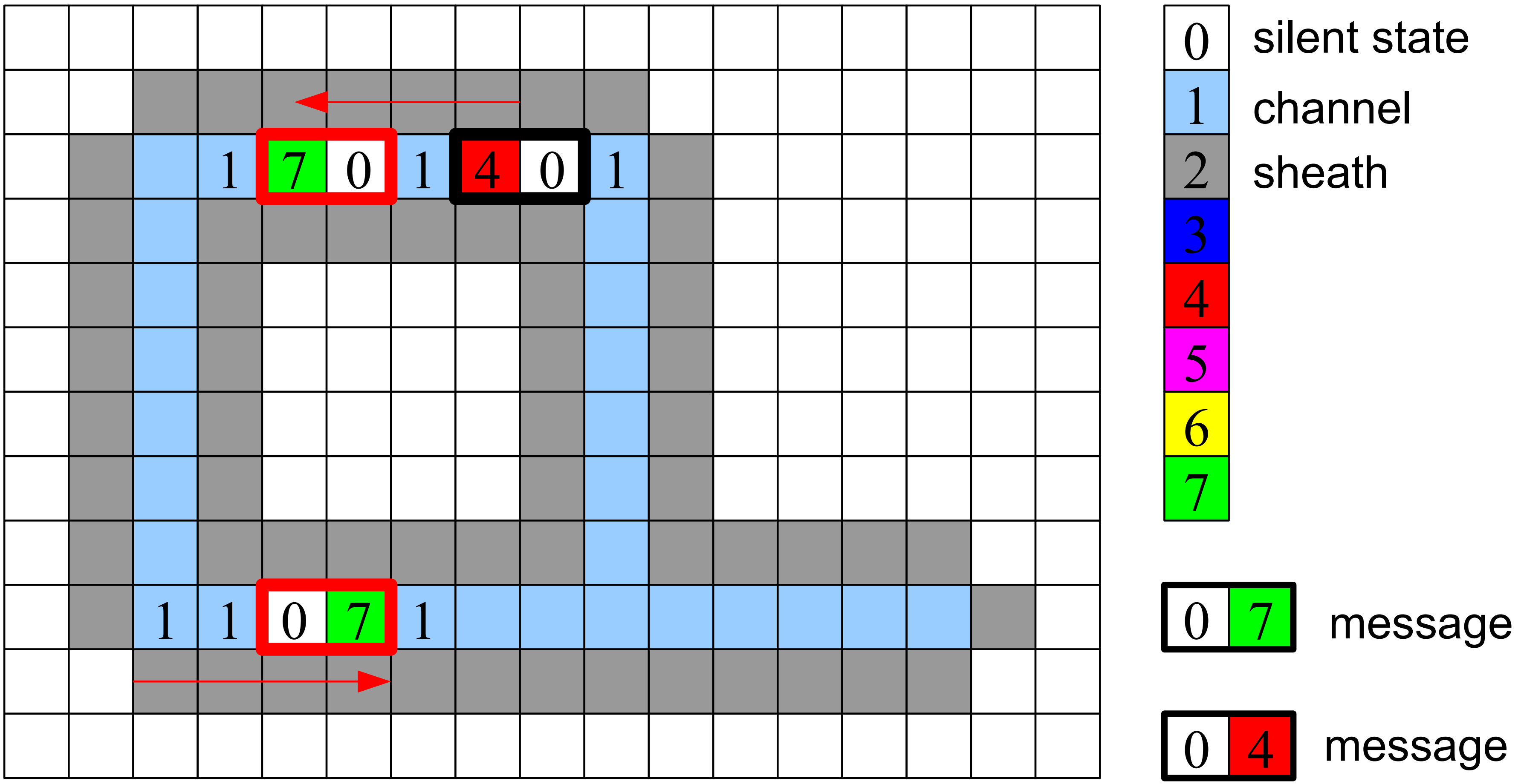
Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984



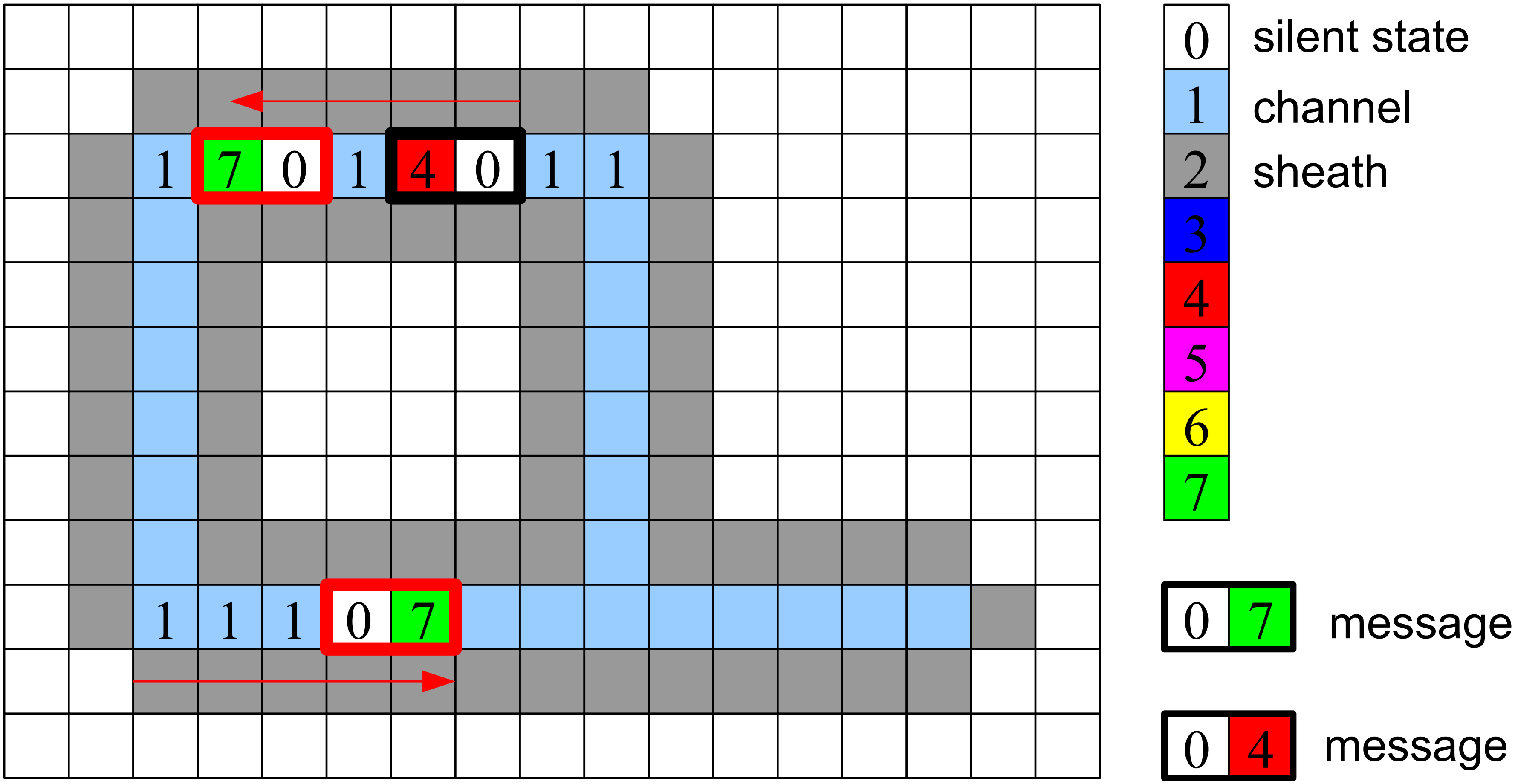
Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984



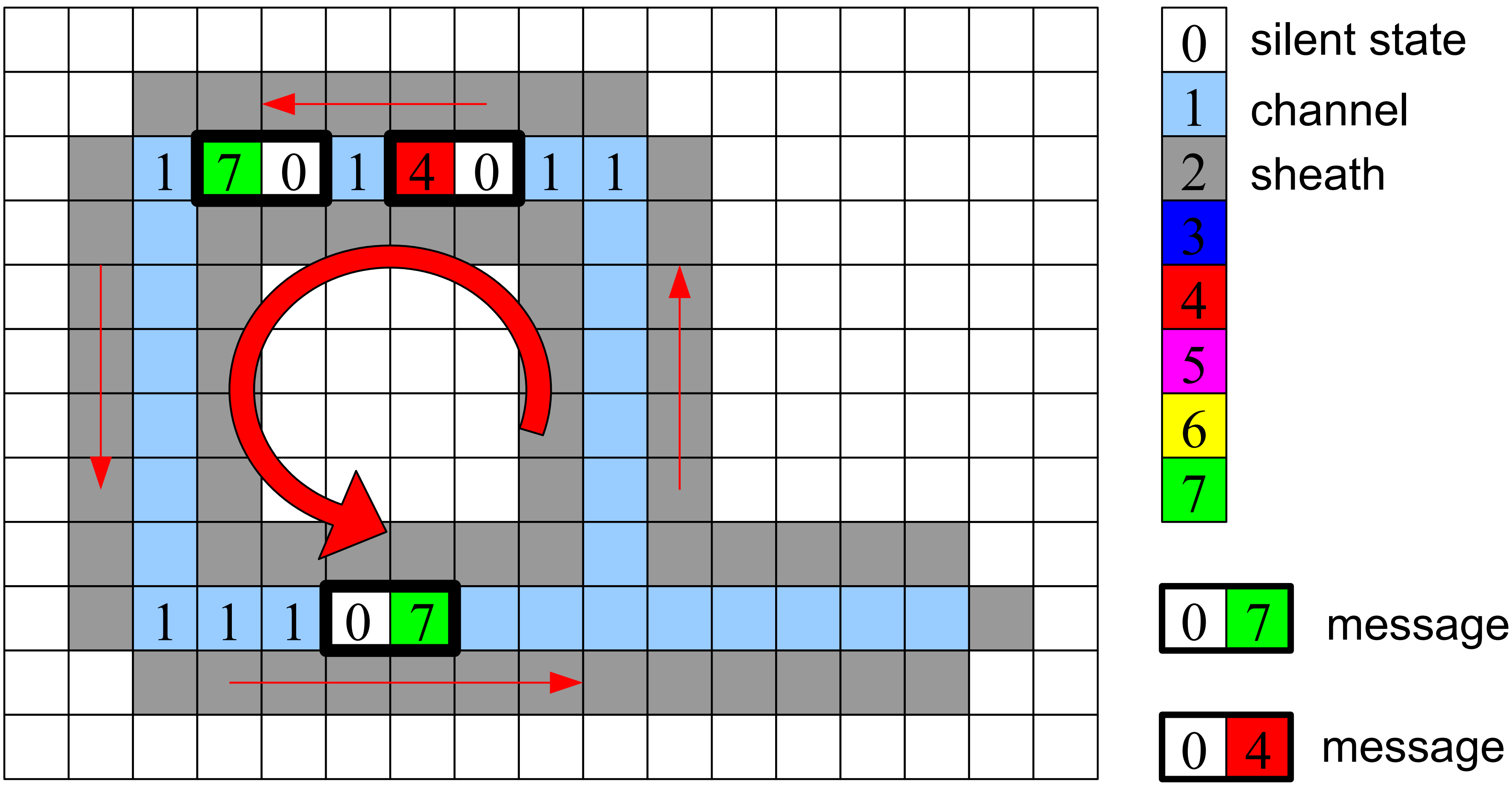
Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984



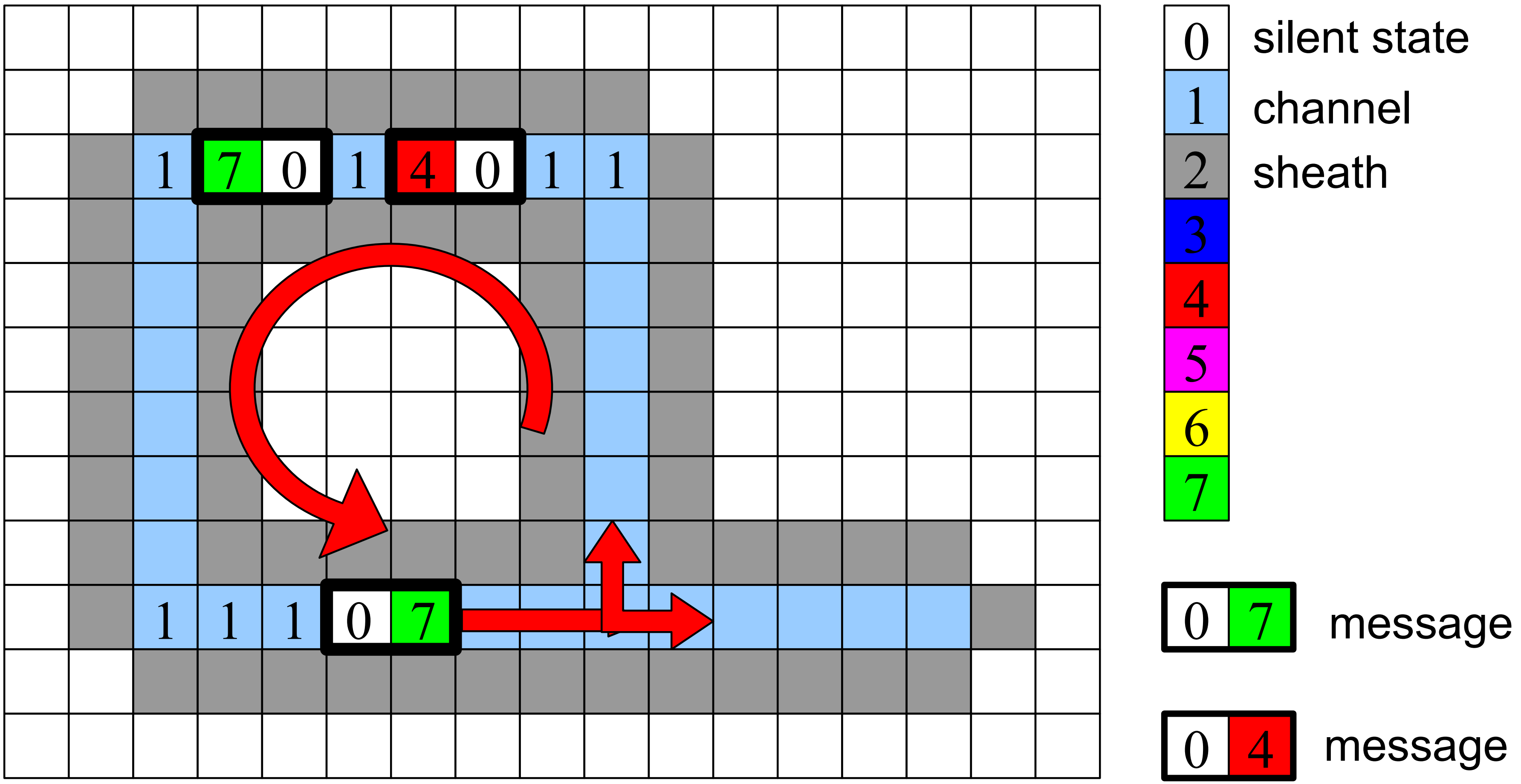
Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984



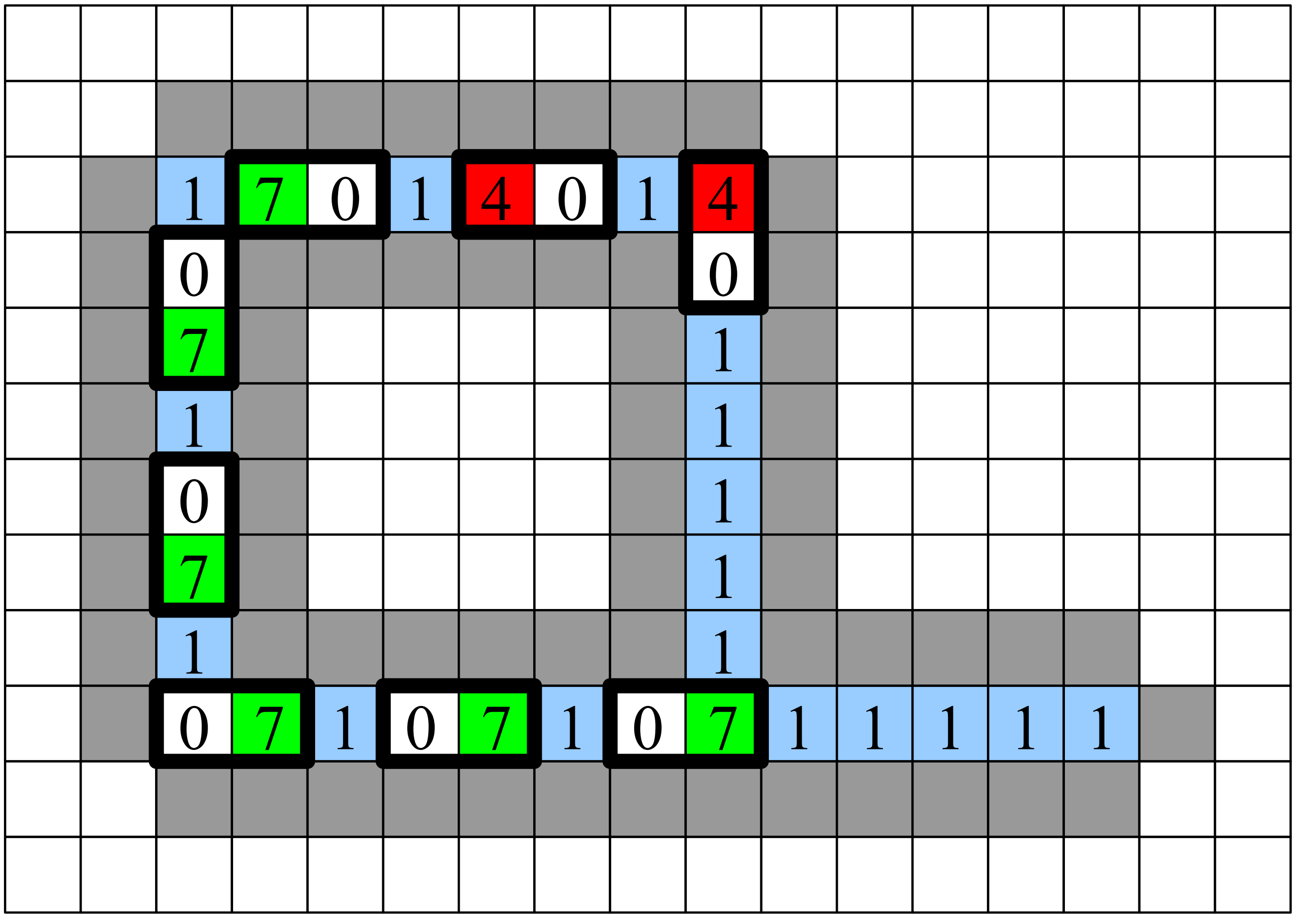
Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984



Langton's Self-Replicating Loop

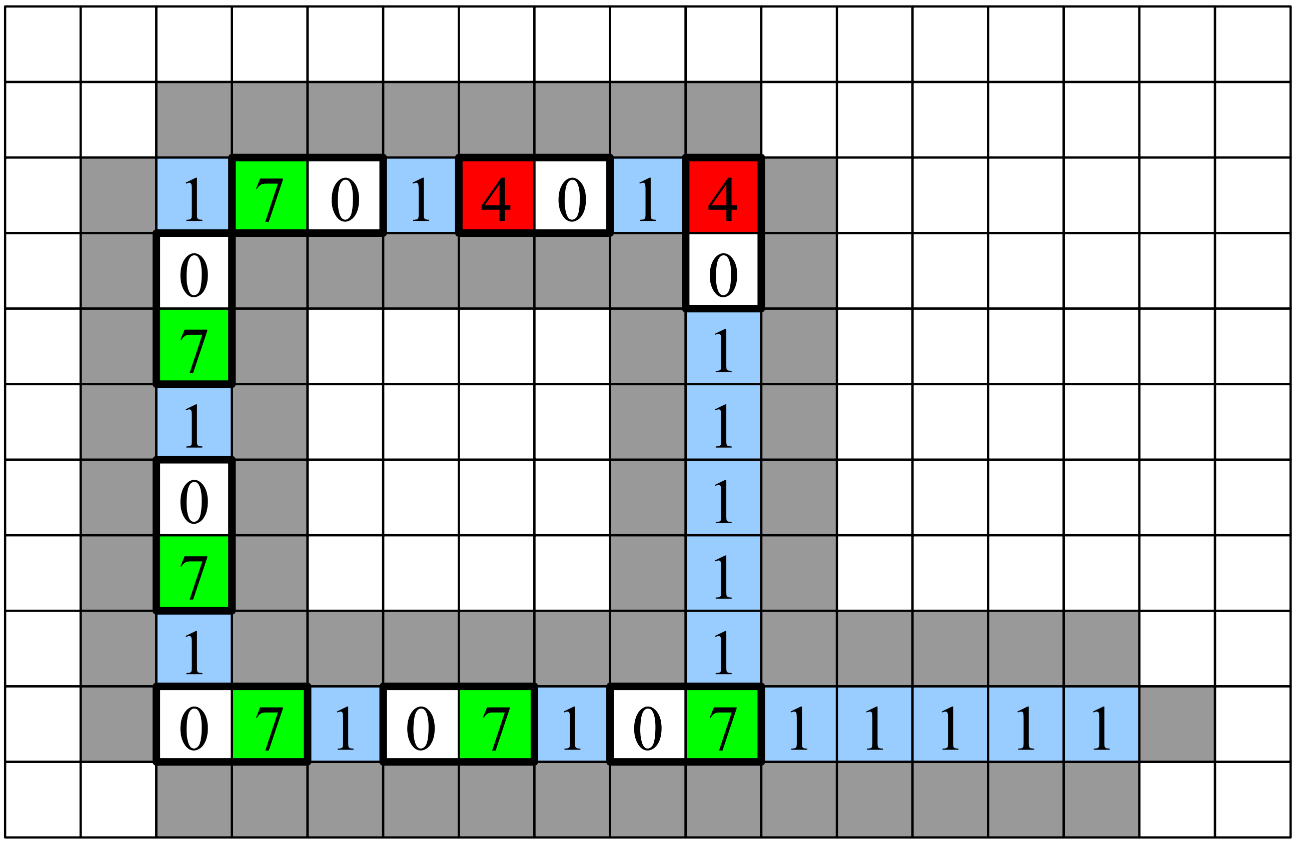
CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984



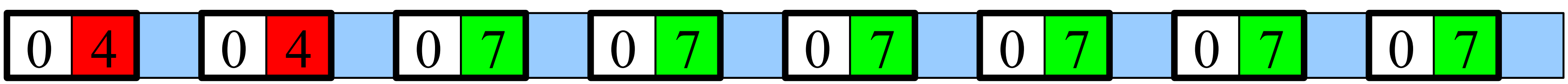
0	silent state
1	channel
2	sheath
3	tmp
4	msg. turn
5	msg. arm bkw.
6	msg. grow arm
7	msg. forward

Langton's Self-Replicating Loop

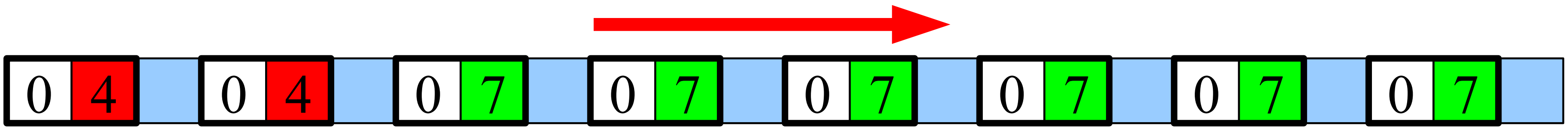
CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984



0	silent state
1	channel
2	sheath
3	tmp
4	msg. turn
5	msg. arm bkw.
6	msg. grow arm
7	msg. forward



Langton's Self-Replicating Loop



The inside of the circular shaped channel of the loop is filled with a string of „8“ messages seperated by „**1**“s, six times „**70**“ followed by two „**40**“::

70 – 70 – 70 – 70 – 70 – 70 – 40 – 40

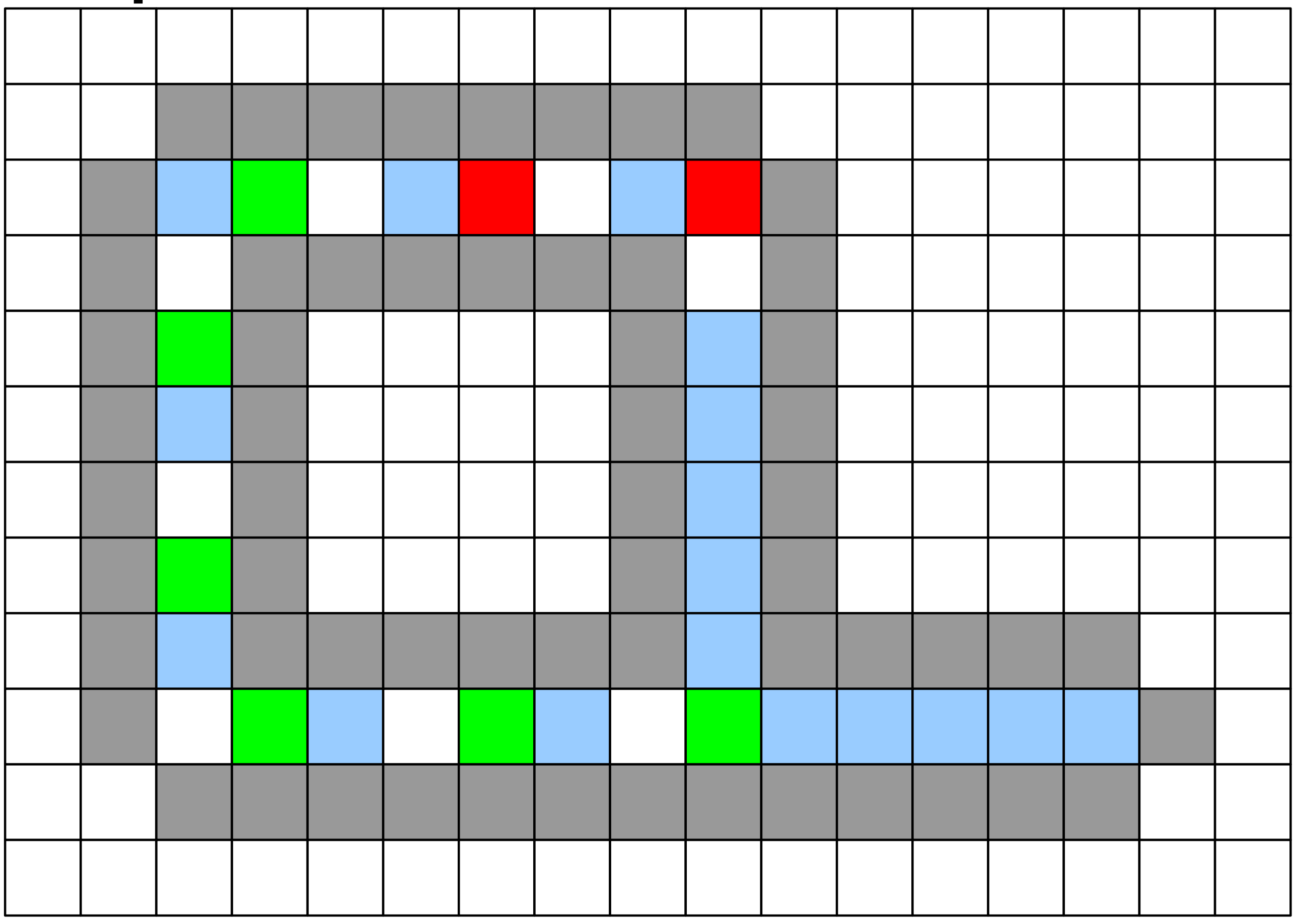
This string of messages is propagated counter clockwise through the channel, including the corners.

At the T-junction, the stream is duplicated and propagated into both branches.

Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

Step 0

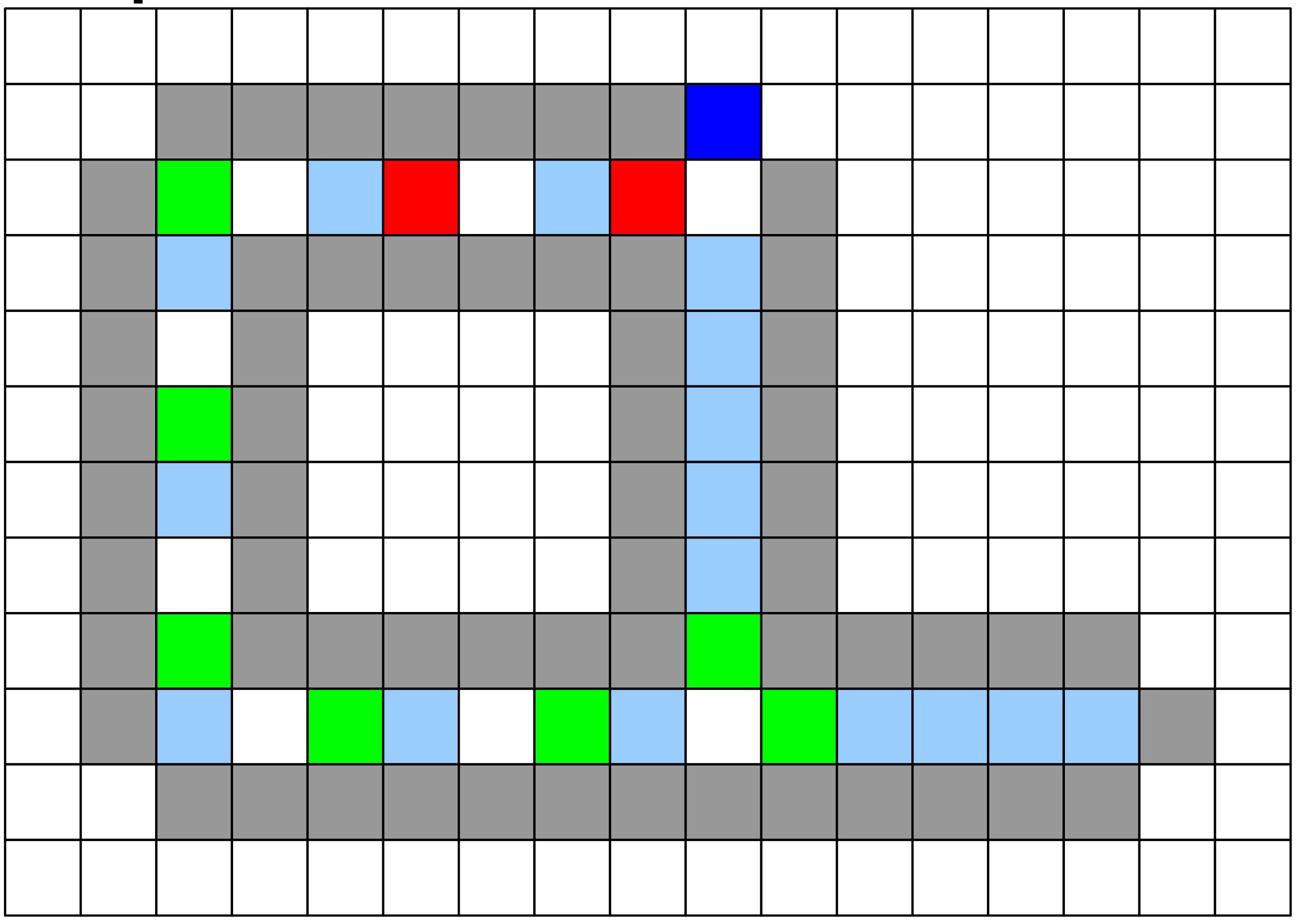


0	silent state
1	channel
2	sheath
3	tmp
4	msg. turn
5	msg. arm bkw.
6	msg. grow arm
7	msg. forward

Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

Step 1

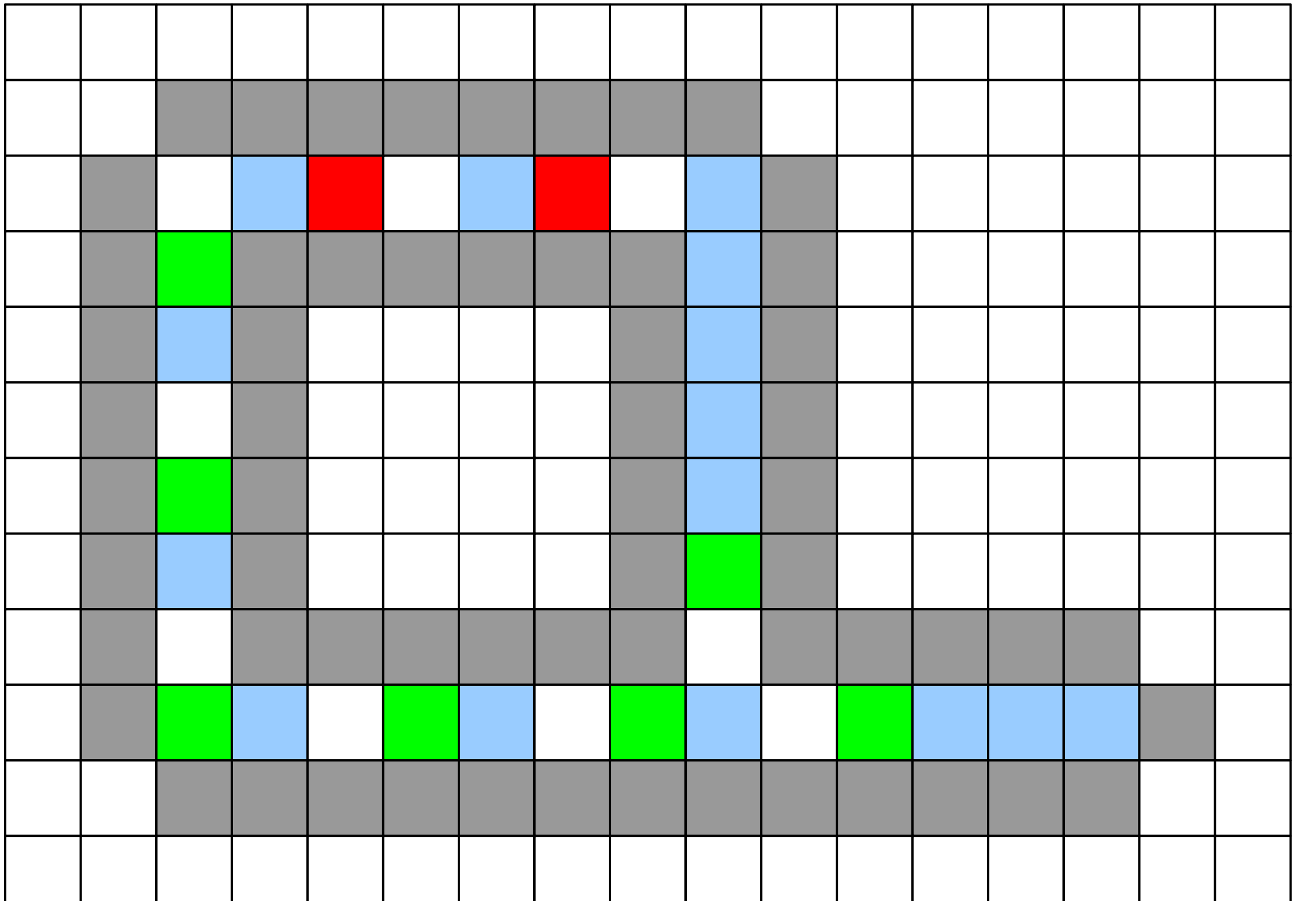


0	silent state
1	channel
2	sheath
3	tmp
4	msg. turn
5	msg. arm bkw.
6	msg. grow arm
7	msg. forward

Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

Step 2

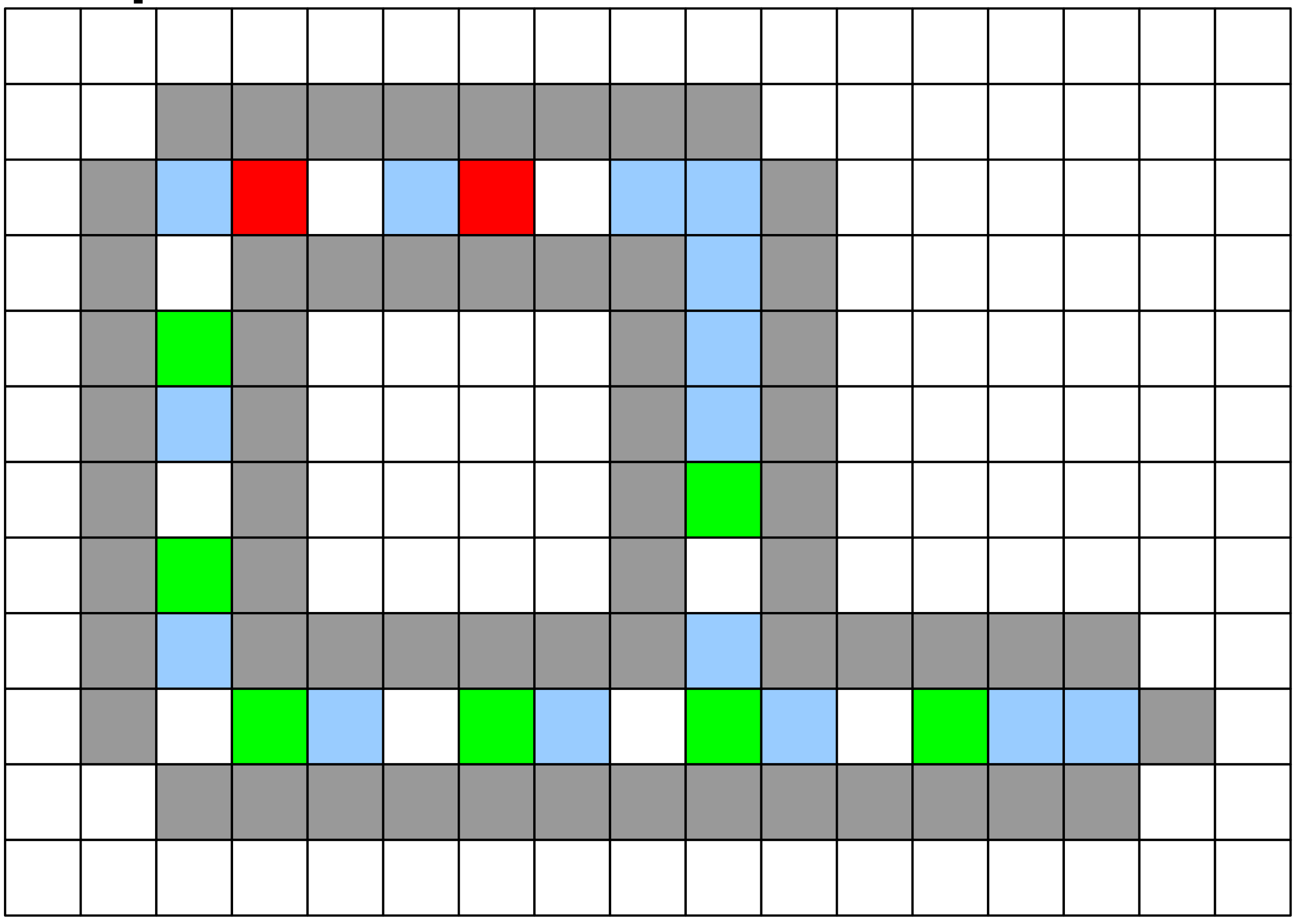


- 0 silent state
- 1 channel
- 2 sheath
- 3 tmp
- 4 msg. turn
- 5 msg. arm bkw.
- 6 msg. grow arm
- 7 msg. forward

Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

Step 3

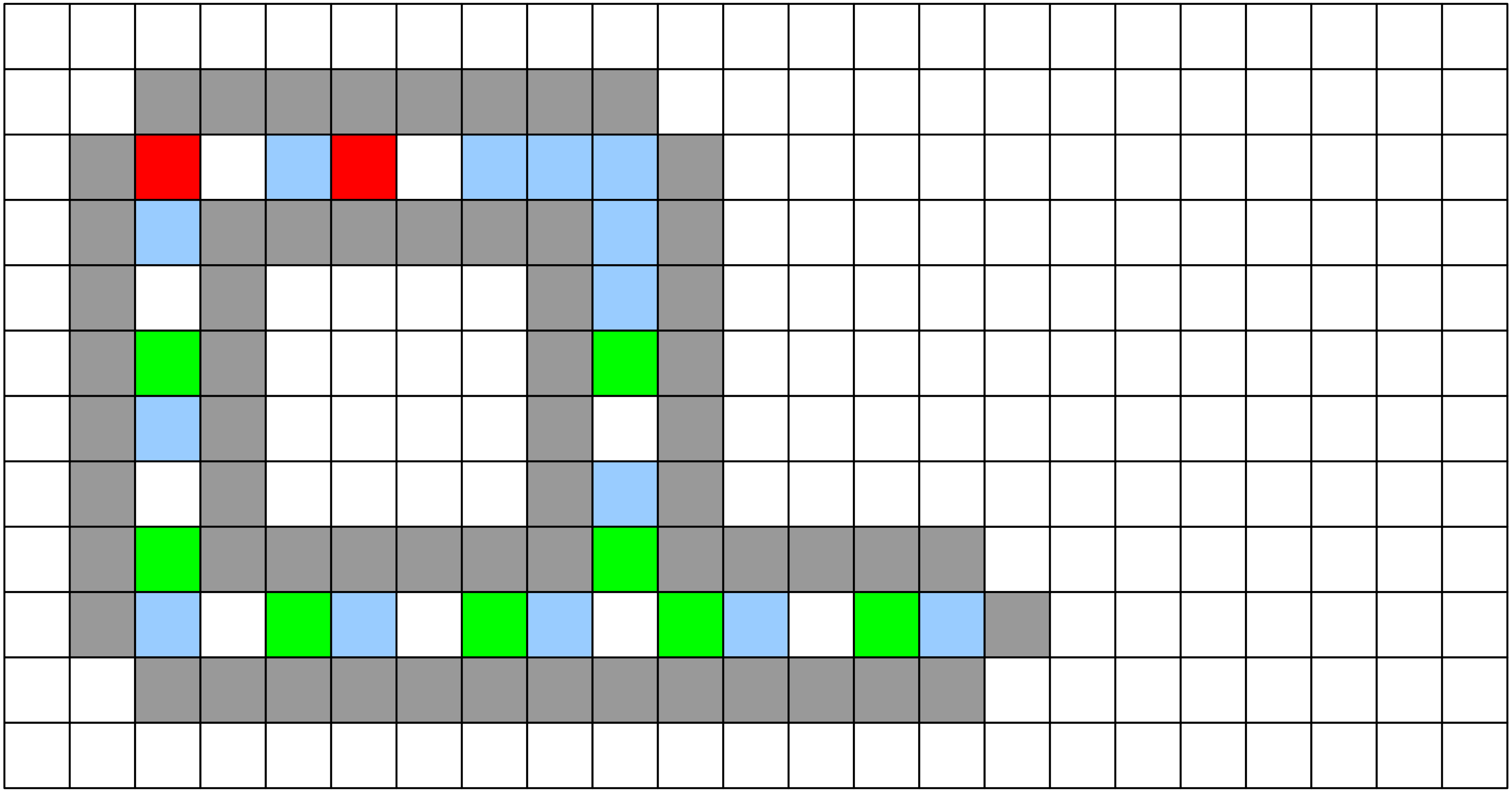


- 0 silent state
- 1 channel
- 2 sheath
- 3 tmp
- 4 msg. turn
- 5 msg. arm bkw.
- 6 msg. grow arm
- 7 msg. forward

Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

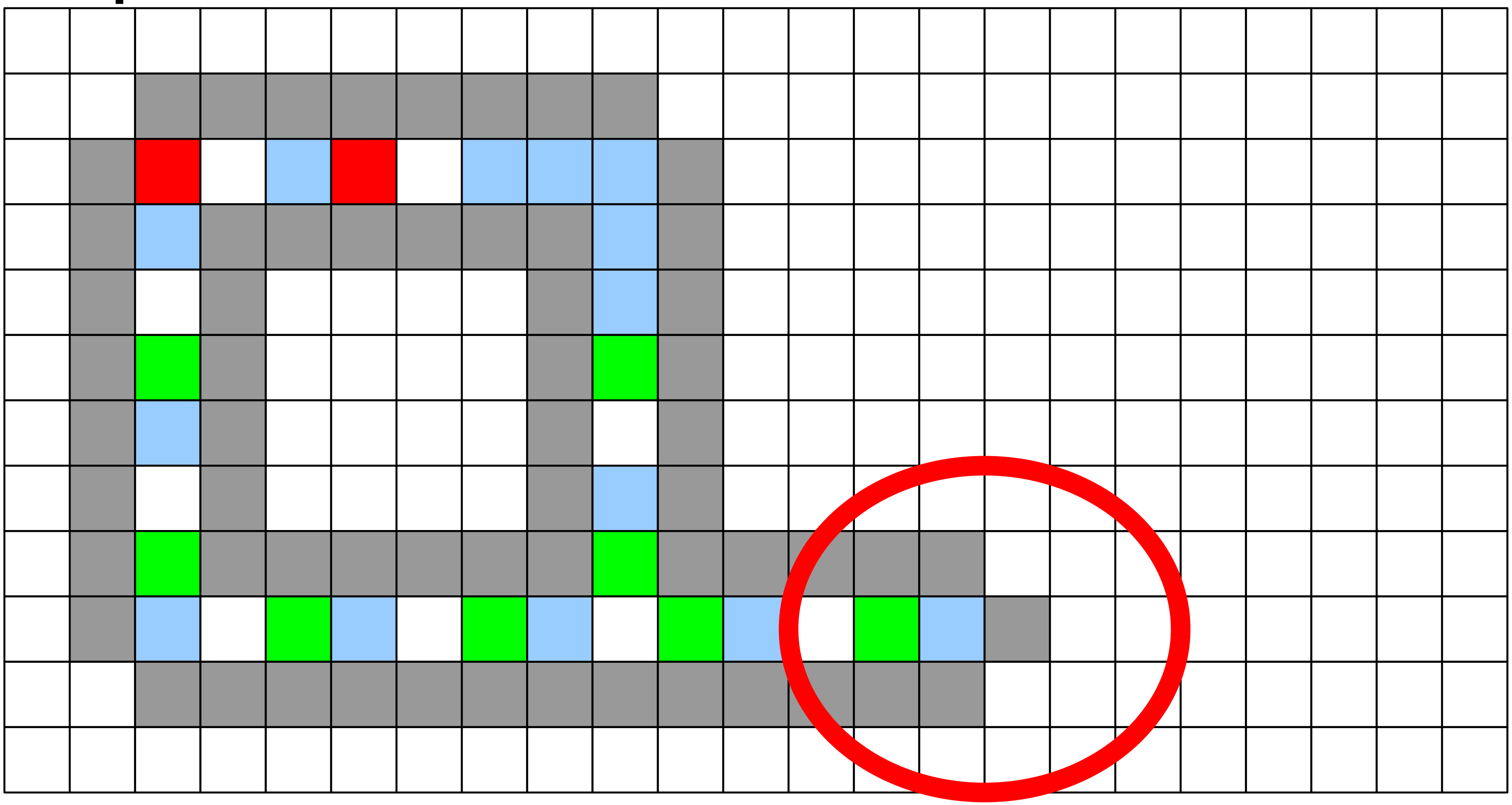
Step 4



Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

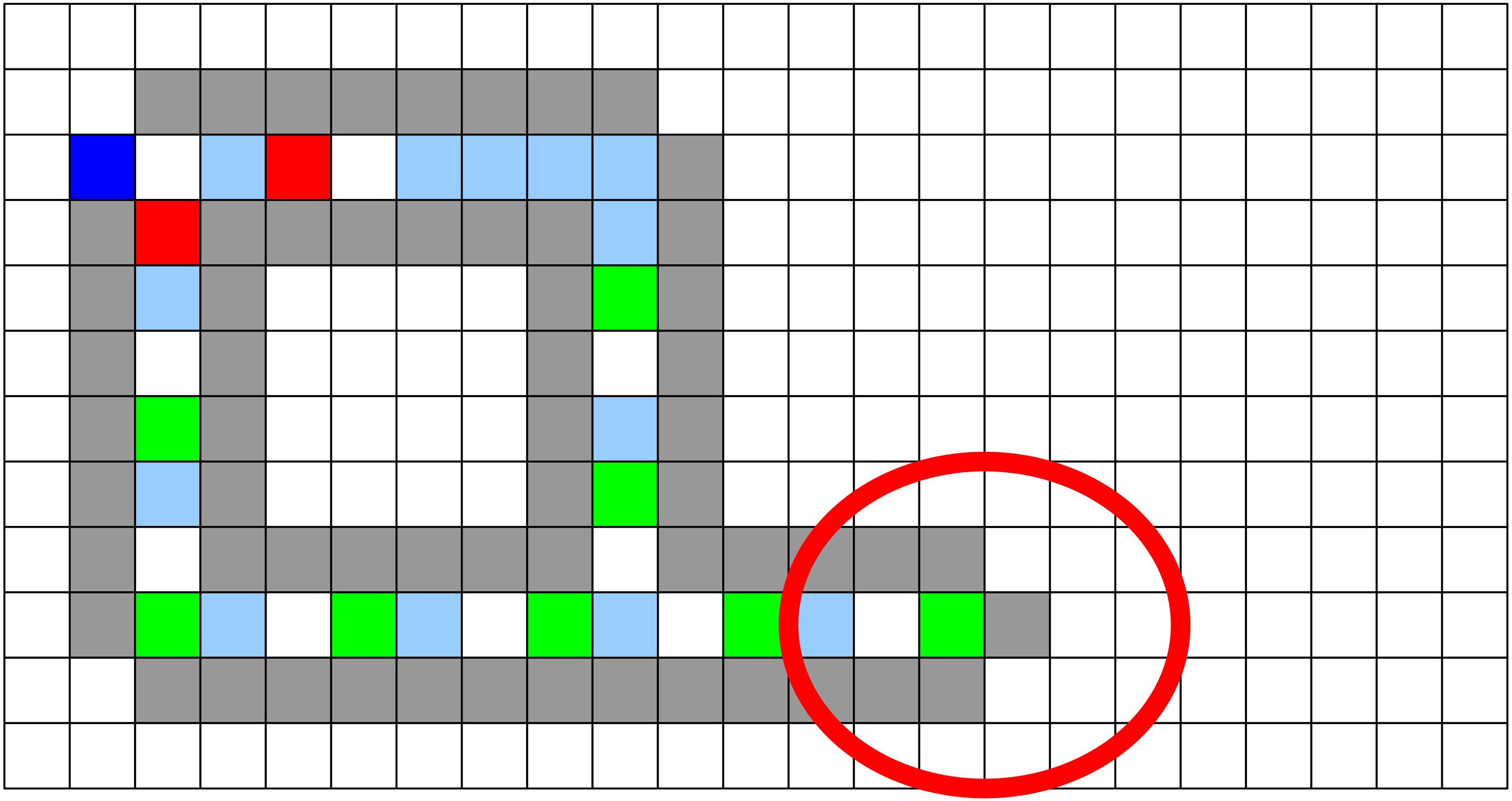
Step 4



Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

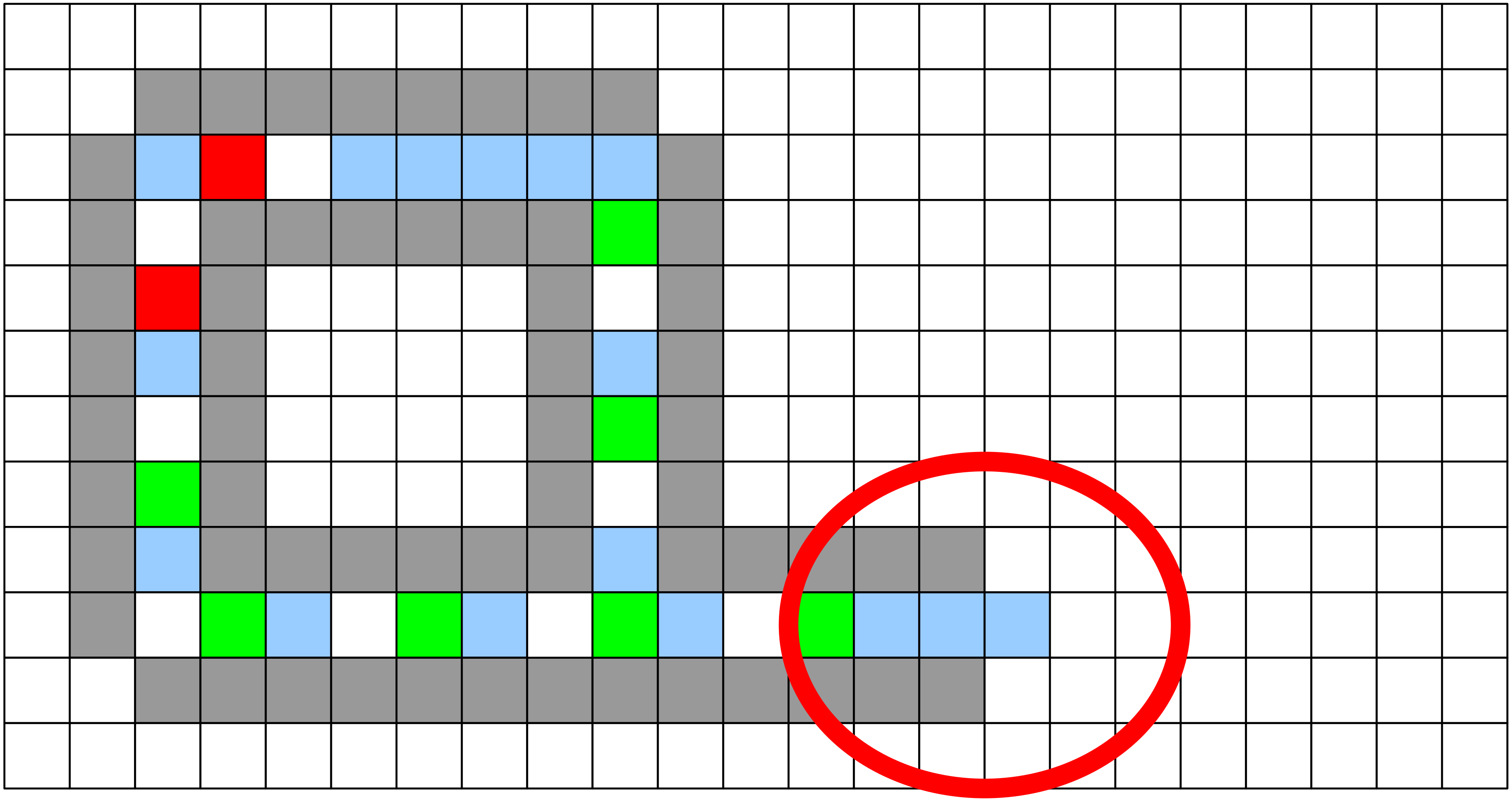
Step 5



Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

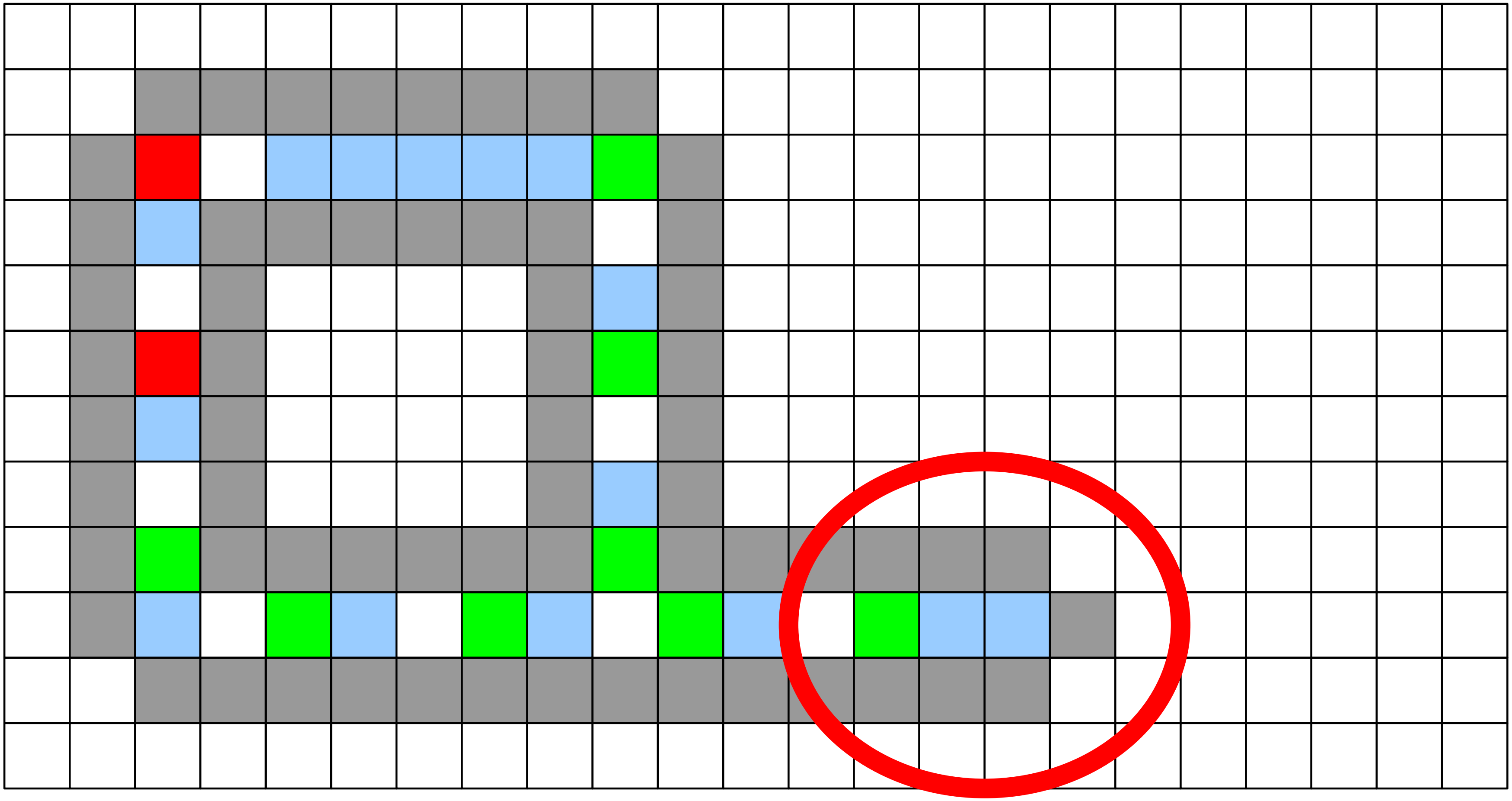
Step 6



Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

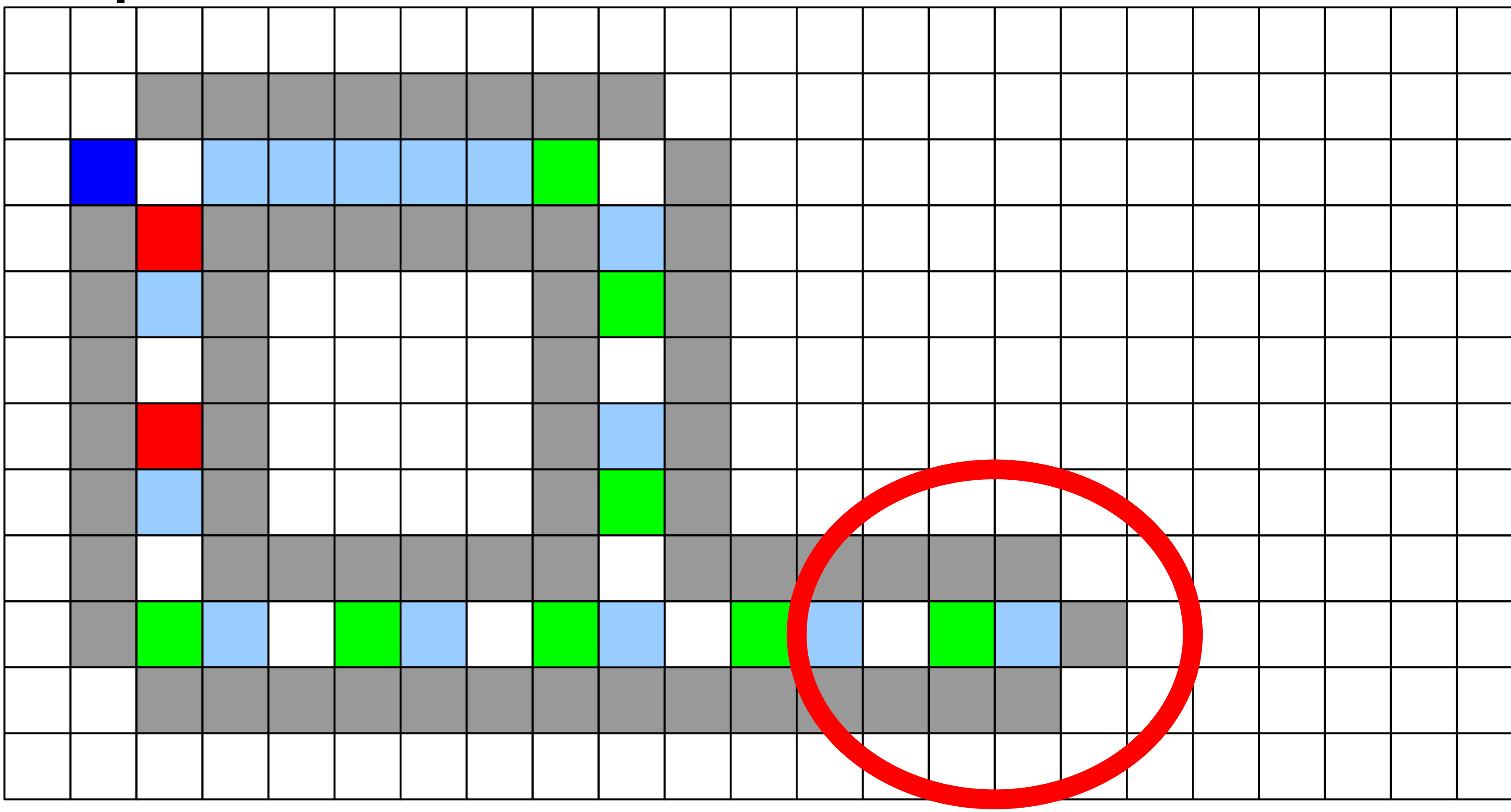
Step 7



Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

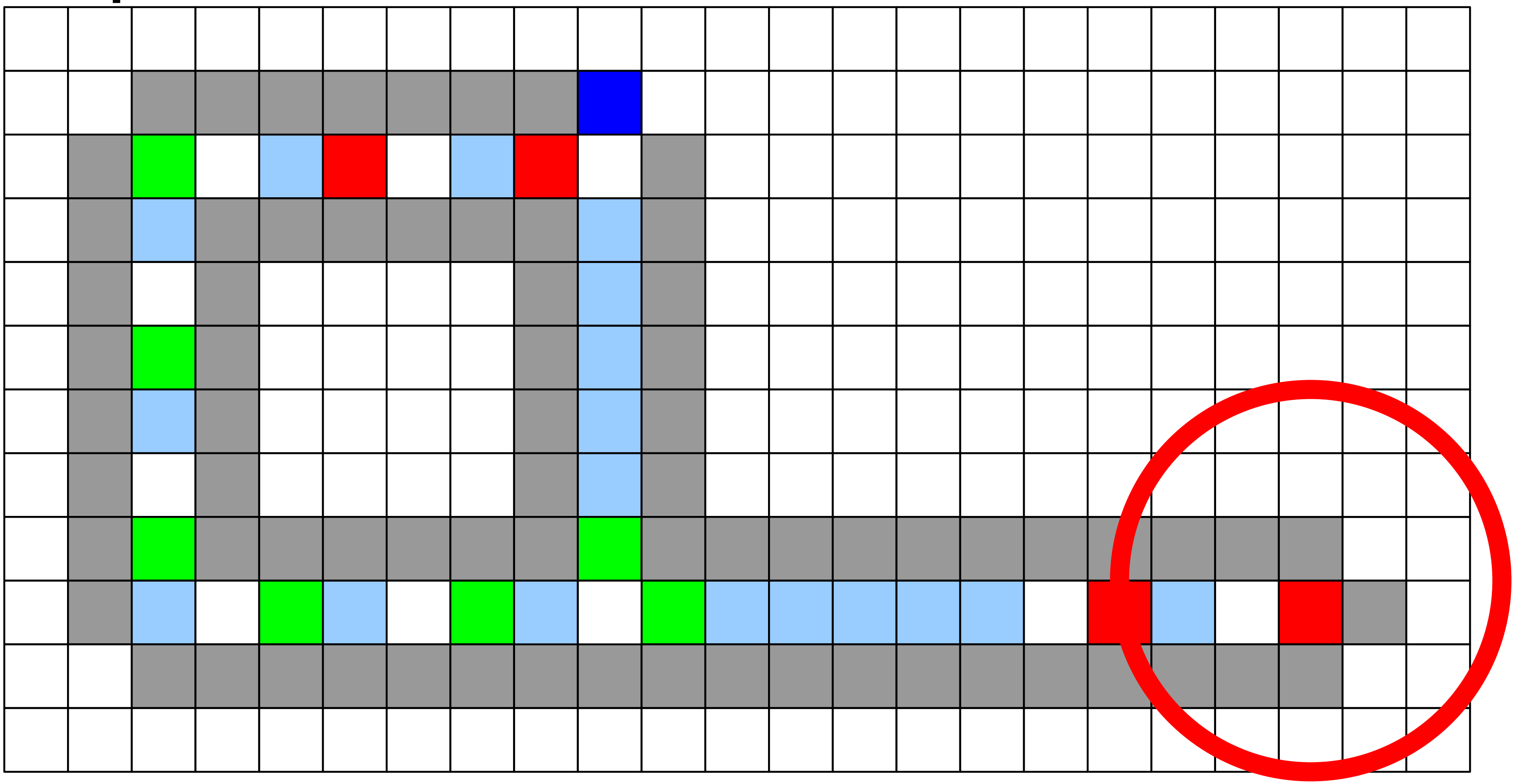
Step 8



Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

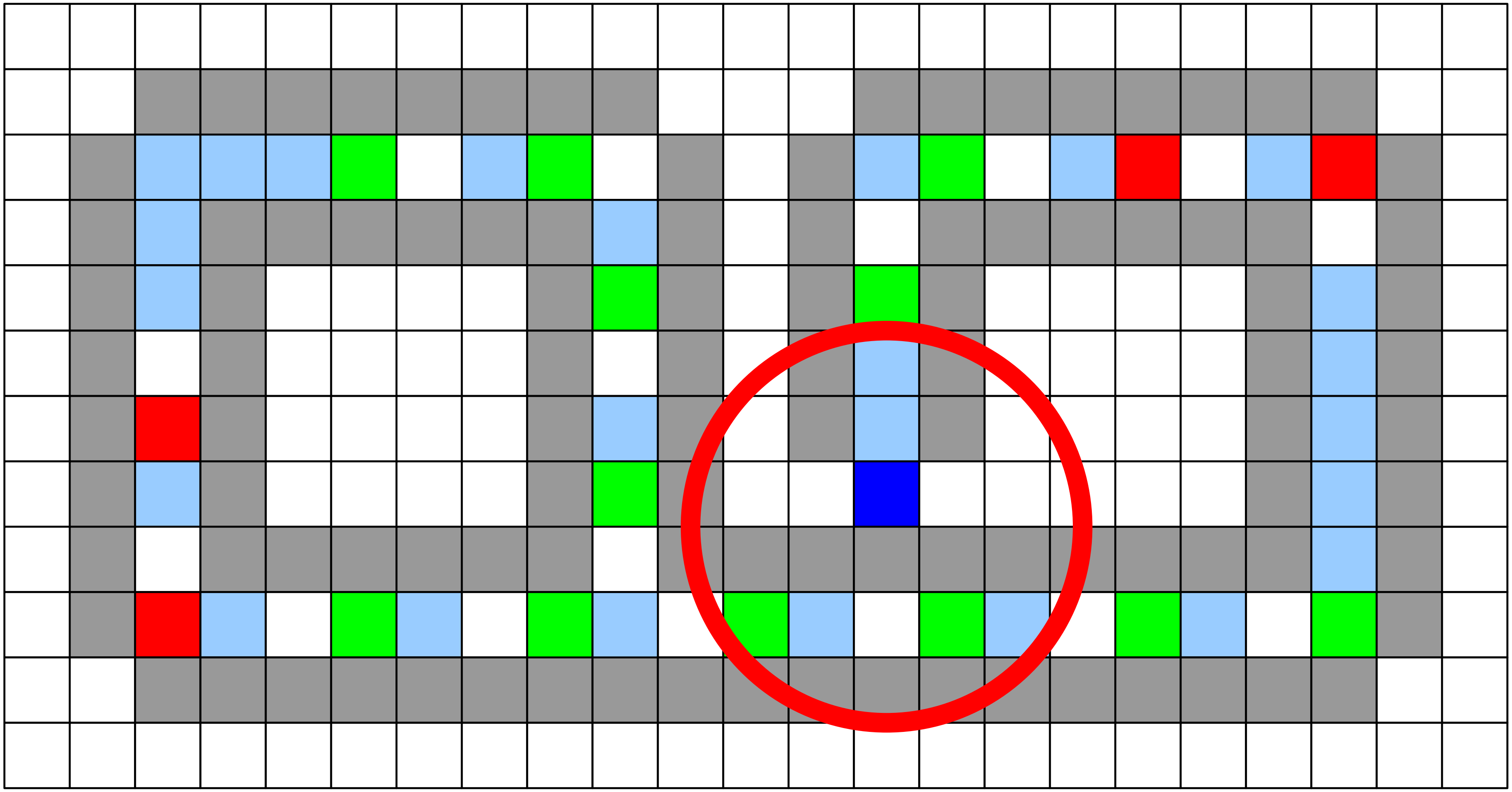
Step 29



Langton's Self-Replicating Loop

CA, d=2, von Neumann neighborhood, r=1, k=8, Christopher Langton 1984

Step 123



Langton's Self-Replicating Loop

The development steps of Langton's Loop:

Step	**0**:	inititalisation, start
Step	**7**:	first prolongation of arm
Steps	**29-34**:	generation of first corner
Step	**122**:	doughter loop closing
Steps	**125-129**:	doughter loop detached
Step	**151**:	both loops are operating
Step	**152**:	cycle 2 starts

Langton's Self-Replicating Loop

One „Mother Loop“ is generating a „Daughter Loop“

Mother Loop

Langton's Self-Replicating Loop

One „Mother Loop“ is generating a „Daughter Loop“

Mother Loop

Mother Loop → **Daughter-1**

Langton's Self-Replicating Loop

One „Mother Loop“ is generating a „Daughter Loop“
Then, the „Mother Loop“ AND the „Daughter Loop“ are producing further loops;

Mother Loop

Mother Loop → **Daughter-1**

Langton's Self-Replicating Loop

One „Mother Loop“ is generating a „Daughter Loop“
Then, the „Mother Loop“ AND the „Daughter Loop“
are producing further loops;

Mother Loop

Mother Loop → **Daughter-1**

Mother Loop → **Daughter-2**
Daughter-1 → **Grand-Daughter-1**

Langton's Self-Replicating Loop

One „Mother Loop“ is generating a „Daughter Loop“
Then, the „Mother Loop“ AND the „Daughter Loop“
are producing further loops;

Mother Loop

Mother Loop → **Daughter-1**

Mother Loop → **Daughter-2**
Daughter-1 → **Grand-Daughter-1**

Mother Loop → **Daughter-3**
Daughter-1 → **Grand-Daughter-2**
Daughter-2 → **Grand-Daughter-3**
Grand-Daughter-1 → **Great-Grand-Doughter-1**

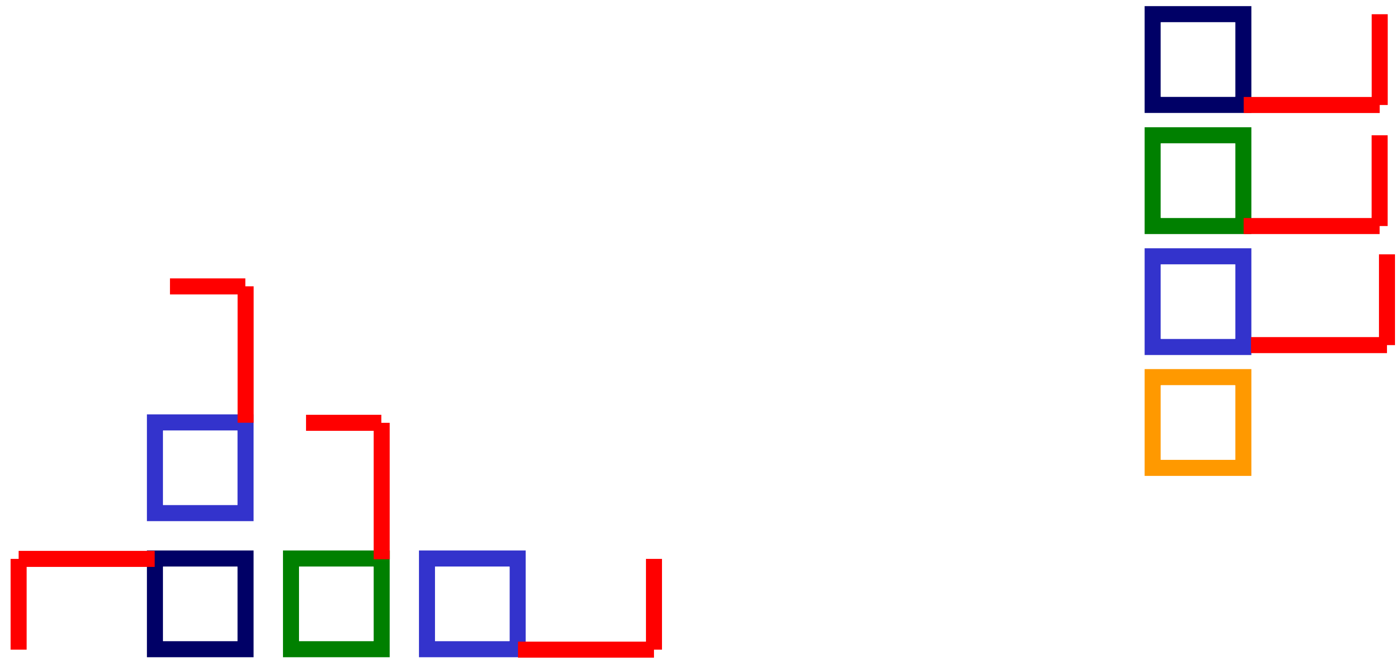
Langton's Self-Replicating Loop

Langton's Self-Replicating Loop

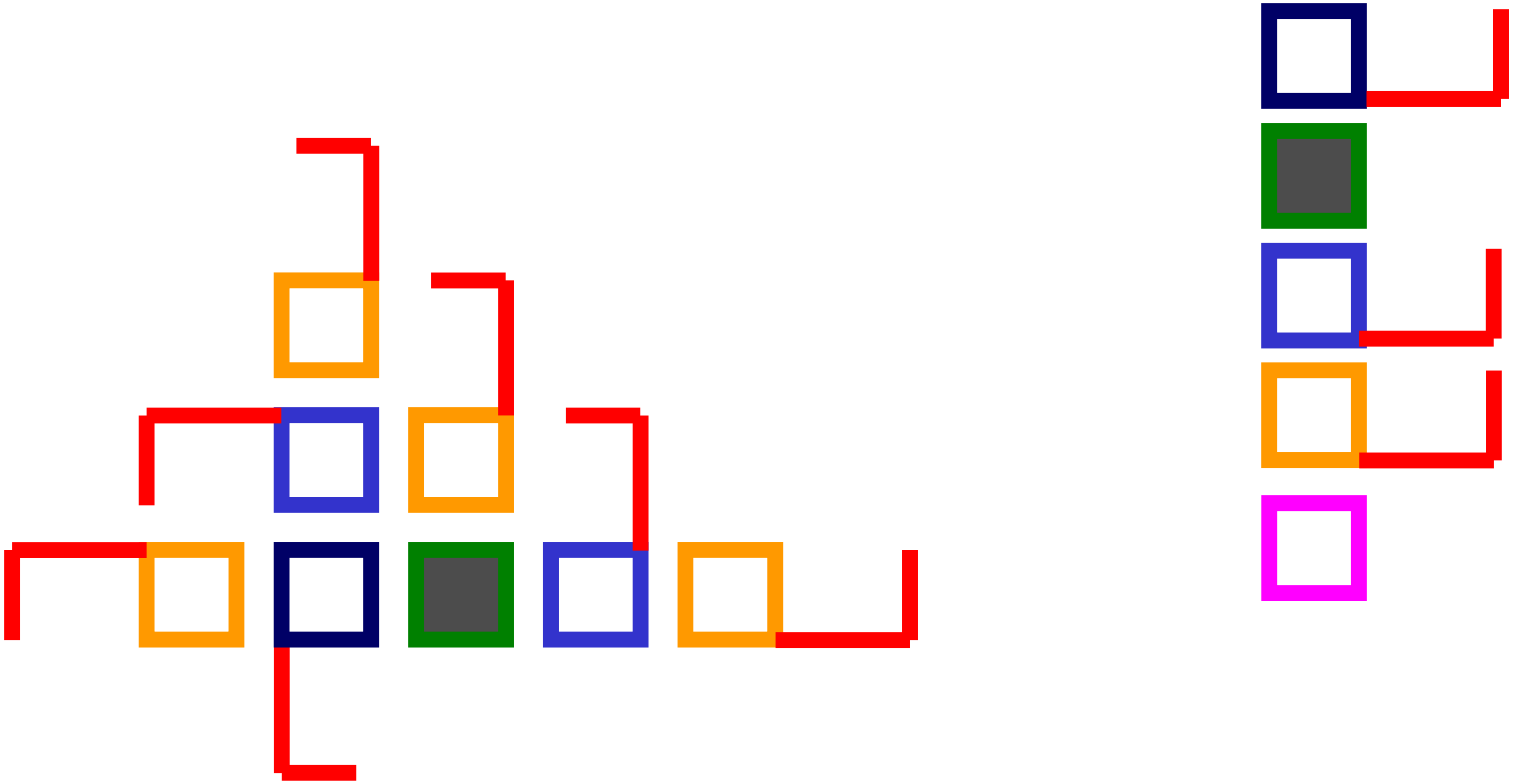
Langton's Self-Replicating Loop

Langton's Self-Replicating Loop

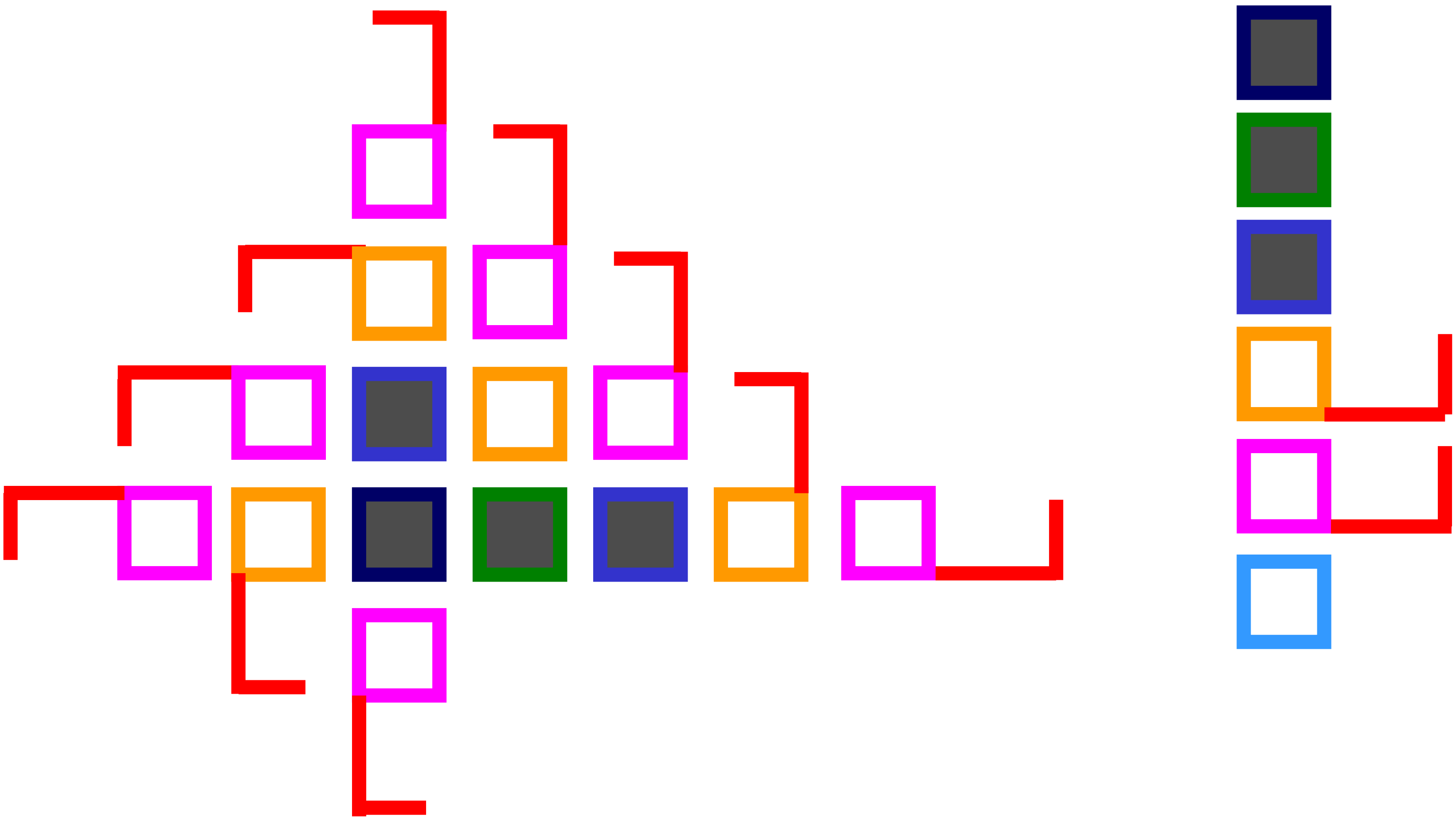
Langton's Self-Replicating Loop



Langton's Self-Replicating Loop



Langton's Self-Replicating Loop



Self Replication

Meanwhile a variety of different implementations for self-replicating machines have been created.

Today, we have an active community researching on various aspects of self-replication, improving the existing approaches, and creating new ones.

For additional information:
see http://en.wikipedia.org/wiki/Langton's_loops
see http://necsi.org/postdocs/sayama/sdsr/java/
see http://carg2.epfl.ch/Teaching/GDCA/loops-thesis.pdf

Self Replicating Loops

Langton's Loop (1984): k=8, von Neumann, S=86, p=151
The original self-reproducing loop.

Byl's Loop (1989): k=6, von Neumann, S=12, p=25
By removing the inner sheath, Byl reduced the size.

Chou-Reggia Loop (1993): k=8, von Neumann, S=5, p=15
A further reduction of the loop by removing all sheaths
Smallest self-reproducing loop known at the moment.

Tempesti Loop (1995): k=10, Moore, S=148, p=304
Tempesti added construction capabilities to his loop.

Perrier Loop (1996): k=64, von Neumann, S=158, p=235
Perrier added a program stack and an extensible data tape.

From: *http://en.wikipedia.org/wiki/Langton's_loops*

Overview:

- Self-Replication
- Langton's Self-Replicating Loop
- **Lindenmayer Systems**

Lindenmayer Systems

In 1968 the theoretical biologist **Aristid Lindenmayer** presented a mathematical formalism to model the growth process of simple cell systems, referred to as:

Lindenmayer Systems or **L-Systems**

Lindenmayer Systems are equivalent to formal grammars; context-free or context-dependent, replacement grammars. The basic L-System is a **D**eterministic, context-free (**zero**-context) grammar,
often denoted as **D0L-System.**

Lindenmayer Systems

The main functional principle of Lindenmayer Systems is the successive **replacement** of strings (single symbols for D0L) by other symbols or strings.

Lindenmayer Systems

The main functional principle of Lindenmayer Systems is the successive **replacement** of strings (single symbols for D0L) by other symbols or strings.

A Lindenmayer System is defined by a 4-tuple:

(V, C, ω, P) or by **(A, ω, P)**

V: set of allowed symbols that may change (**V**ariables)

C: set of allowed symbols that stay **C**onstant,

A = (**V** + **C**) build the allowed alphabet **A**.

ω: Axiom, starting symbol from the alphabet $\omega \in A$

P: Production rules, mapping from $A^n \rightarrow A^m$

context free (D0L) rules map from $A^1 \rightarrow A^m$

Lindenmayer Systems: Example

Lindenmayer Systems: Example

Variables:
Axiom:
Rule 1:
Rule 2:

Lindenmayer Systems: Example

Variables: **C**, **A** (**Child, Adult**)

Axiom:

Rule 1:

Rule 2:

Lindenmayer Systems: Example

Variables:	**C**, **A**	(**Child, Adult**)
Axiom:	**C**	
Rule 1:		
Rule 2:		

Lindenmayer Systems: Example

Variables:	**C**, **A**	(**Child, Adult**)
Axiom:	**C**	
Rule 1:	**C** $\longrightarrow$ **A**	grows up
Rule 2:		

Lindenmayer Systems: Example

Variables:	**C**, **A**		(**Child, Adult**)
Axiom:	**C**		
Rule 1:	**C** $\longrightarrow$	**A**	grows up
Rule 2:	**A** $\longrightarrow$	**CA**	offspring

Lindenmayer Systems: Example

Variables:	**C**, **A**			(**Child**, **Adult**)
Axiom:	**C**			
Rule 1:	**C**	$\longrightarrow$	**A**	grows up
Rule 2:	**A**	$\longrightarrow$	**CA**	offspring

Step string length

Lindenmayer Systems: Example

Variables:	C, A		**(Child, Adult)**
Axiom:	**C**		
Rule 1:	**C** $\longrightarrow$	**A**	grows up
Rule 2:	**A** $\longrightarrow$	**CA**	offspring

Step	**string**		**length**
0	**C**	(as defined in the axiom)	**1**

Lindenmayer Systems: Example

Variables:	**C**, **A**			**(Child, Adult)**
Axiom:	**C**			
Rule 1:	**C**	$\longrightarrow$	**A**	grows up
Rule 2:	**A**	$\longrightarrow$	**CA**	offspring

Step	**string**		**length**
0	**C**	(apply rule 1)	**1**

Lindenmayer Systems: Example

Variables:	**C**, **A**			(**Child**, **Adult**)
Axiom:	**C**			
Rule 1:	**C**	$\longrightarrow$	**A**	grows up
Rule 2:	**A**	$\longrightarrow$	**CA**	offspring

Step	string	length
0	**C**	**1**
1	**A**	**1**

Lindenmayer Systems: Example

Variables:	**C**, **A**			(**Child, Adult**)
Axiom:	**C**			
Rule 1:	**C**	⟶	**A**	grows up
Rule 2:	**A**	⟶	**CA**	offspring

Step	**string**		**length**
0	**C**		**1**
1	**A**	(apply rule 2)	**1**

Lindenmayer Systems: Example

Variables:	**C**, **A**		**(Child, Adult)**
Axiom:	**C**		
Rule 1:	**C** $\longrightarrow$	**A**	grows up
Rule 2:	**A** $\longrightarrow$	**CA**	offspring

Step	**string**	**length**
0	**C**	**1**
1	**A**	**1**
2	**CA**	**2**

Lindenmayer Systems: Example

Variables:	C, A		(**Child**, **Adult**)
Axiom:	C		
Rule 1:	C $\longrightarrow$	A	grows up
Rule 2:	A $\longrightarrow$	CA	offspring

Step	string		length
0	C		**1**
1	A		**1**
2	CA	(apply rule 1 and rule 2)	**2**

Lindenmayer Systems: Example

Variables:	**C**, **A**			**(Child, Adult)**
Axiom:	**C**			
Rule 1:	**C**	$\longrightarrow$	**A**	grows up
Rule 2:	**A**	$\longrightarrow$	**CA**	offspring

Step	**string**	**length**
0	**C**	**1**
1	**A**	**1**
2	**CA**	**2**
3	**ACA**	**3**

Lindenmayer Systems: Example

Variables:	**C**, **A**			**(Child, Adult)**
Axiom:	**C**			
Rule 1:	**C**	→	**A**	grows up
Rule 2:	**A**	→	**CA**	offspring

Step	**string**	**length**
0	**C**	**1**
1	**A**	**1**
2	**CA**	**2**
3	**ACA**	**3**
4	**CAACA**	**5**

Lindenmayer Systems: Example

Variables:	**C**, **A**			(**Child, Adult**)
Axiom:	**C**			
Rule 1:	**C**	→	**A**	grows up
Rule 2:	**A**	→	**CA**	offspring

Step	string	length
0	**C**	**1**
1	**A**	**1**
2	**CA**	**2**
3	**ACA**	**3**
4	**CAACA**	**5**
5	**ACACAACA**	**8**

Lindenmayer Systems: Example

Variables:	**C**, **A**			(**Child**, **Adult**)
Axiom:	**C**			
Rule 1:	**C**	$\longrightarrow$	**A**	grows up
Rule 2:	**A**	$\longrightarrow$	**CA**	offspring

Step	string	length
0	**C**	**1**
1	**A**	**1**
2	**CA**	**2**
3	**ACA**	**3**
4	**CAACA**	**5**
5	**ACACAACA**	**8**
6	**CAACAACACAACA**	**13**

Lindenmayer Systems: Example

Variables:	C, A			(Child, Adult)
Axiom:	C			
Rule 1:	C	→	A	grows up
Rule 2:	A	→	CA	offspring

Step	string	length
0	C	1
1	A	1
2	CA	2
3	ACA	3
4	CAACA	5
5	ACACAACA	8
6	CAACAACACAACA	13
7	ACACAACACAACAACACAACA	21
8	... ...	

Lindenmayer Systems: Example

Variables:	**C**, **A**			(**Child**, **Adult**)
Axiom:	**C**			
Rule 1:	**C**	$\longrightarrow$	**A**	grows up
Rule 2:	**A**	$\longrightarrow$	**CA**	offspring

Step	string	length
0	C	1
1	A	1
2	CA	2
3	ACA	3
4	CAACA	5
5	ACACAACA	8
6	CAACAACACAACA	13
7	ACACAACACAACAACACAACA	21
8	... ...	

Lindenmayer Systems: Example

Variables:	**C**, **A**		(**Child**, **Adult**)
Axiom:	**C**		
Rule 1:	**C** →	**A**	grows up
Rule 2:	**A** →	**CA**	offspring

Step	string		length
0	**C**	(apply rule 1)	**1**
1	**A**	(apply rule 2)	**1**
2	**CA**	(apply rule 1 and rule 2)	**2**
3	**ACA**		**3**
4	**CAACA**		**5**
5	**ACACAACA**		**8**
6	**CAACAACACAACA**		**13**
7	**ACACAACACAACAACACAACA**		**21**
8	**... ... ...**		

Lindenmayer Systems

In 1974 a graphical visualization scheme has been proposed for the **L-Systems** (P. Hogeweg und B. Hesper) and in 1984 (A.R.Smith) it was extended to describe and model the growth process of complete plants.

To describe and model the morphogenesis of the plants, the **spatial position** of the variables, is getting important.

Lindenmayer Systems: Example 2

Variables:

Constants:

Axiom:

Rule 1:

Rule 2:

Lindenmayer Systems: Example 2

Variables:	B, F
Constants:	none
Axiom:	B
Rule 1:	B $\longrightarrow$ BFB
Rule 2:	F $\longrightarrow$ FF

Lindenmayer Systems: Example 2

Variables:	B, F
Constants:	none
Axiom:	B
Rule 1:	B $\longrightarrow$ BFB
Rule 2:	F $\longrightarrow$ FF

position, space

time, steps

Lindenmayer Systems: Example 2

Variables:	B, F
Constants:	none
Axiom:	B
Rule 1:	B → BFB
Rule 2:	F → FF

position, space

0 B

time, steps

Lindenmayer Systems: Example 2

Variables:	B, F	
Constants:	none	
Axiom:	B	
Rule 1:	B →	BFB
Rule 2:	F →	FF

→ position, space

0 B

1 BFB

↓ time, steps

Lindenmayer Systems: Example 2

Variables:	B, F
Constants:	none
Axiom:	B
Rule 1:	B → BFB
Rule 2:	F → FF

position, space

0 B

1 BFB

2 BFB FF BFB

time, steps

Lindenmayer Systems: Example 2

Variables:	B, F		
Constants:	none		
Axiom:	B		
Rule 1:	B	$\longrightarrow$	BFB
Rule 2:	F	$\longrightarrow$	FF

position, space $\longrightarrow$

time, steps $\downarrow$

0 B

1 BFB

2 BFB FF BFB

3 BFB FF BFB FFFF BFB FF BFB

Lindenmayer Systems: Example 2

Variables:	B, F		
Constants:	none		
Axiom:	B		
Rule 1:	B	→	BFB
Rule 2:	F	→	FF

position, space →

time, steps ↓

0	B
1	BFB
2	BFB FF BFB
3	BFB FF BFB FFFF BFB FF BFB
4	BFBFFBFBFFFFBFBFFBFBFFFFFFFFBFBFFBFBFFFFBFBFFBFB

Lindenmayer Systems: Example 2

Variables:	B, F		
Constants:	none		
Axiom:	B		
Rule 1:	B	$\longrightarrow$	BFB
Rule 2:	F	$\longrightarrow$	FF

$\longrightarrow$ position, space

$\downarrow$ time, steps

0 B

1 BFB

2 BFB FF BFB

3 BFB FF BFB FFFF BFB FF BFB

4 BFBFFBFBFFFFBFBFFBFBFFFFFFFFBFBFFBFBFFFFBFBFFBFB

5 … … … … … … … … … … …

Lindenmayer Systems

To describe and model the morphogenesis of the plants, the **spatial position** of the variables, is getting important.

Unfortunately the arrangement of the symbols is getting confusing rather quick.
To circumvent this, a **visualization** in 2- and 3-dimensions has been proposed.

Lindenmayer Systems

The graphical visualization of L-Systems is aligned with the syntax of the **Turtle Graphics** of the **Logo** programming language.
The variables of the alphabet are regarded as drawing commands, and additional constants like (**+, -, [,]**) have been defined to serve as commands to turn and position the turtle in 2- or 3-dim. space.

+ turn left, **-** turn right, by α degree, scale by **s**
[remember this position in space (push to stack)
] restore the last position (pull from stack)

Lindenmayer Systems: Example 2

Variables:	B, F			
Constants:	+, -, [,]			
Axiom:	B			
Rule 1:	B	→	F[-B]+B	(former BFB)
Rule 2:	F	→	FF	

0 B

1' F [- B]+ B

2' FF [- F[-B]+B]+ F[-B]+B

3' FFFF [- FF [- F[-B]+B]+ F[-B]+B]+ FF [- F[-B]+B]+ F[-B]+B

3 BFB FF BFB FFFF BFB FF BFB

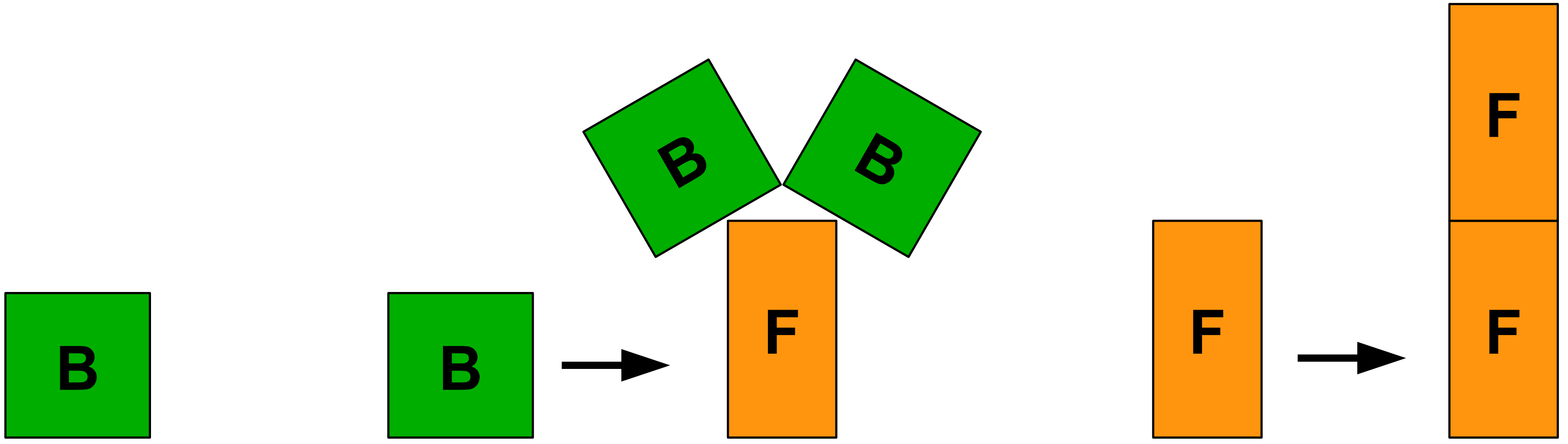
Lindenmayer Systems: Example 2

Axiom: B

Rule 1: B ⟶ F[-B]+B

Rule 2: F ⟶ FF

$\alpha = 30°$



Axiom:
B

Rule 1:
B ⟶ F[-B]+B

Rule 2:
F ⟶ FF

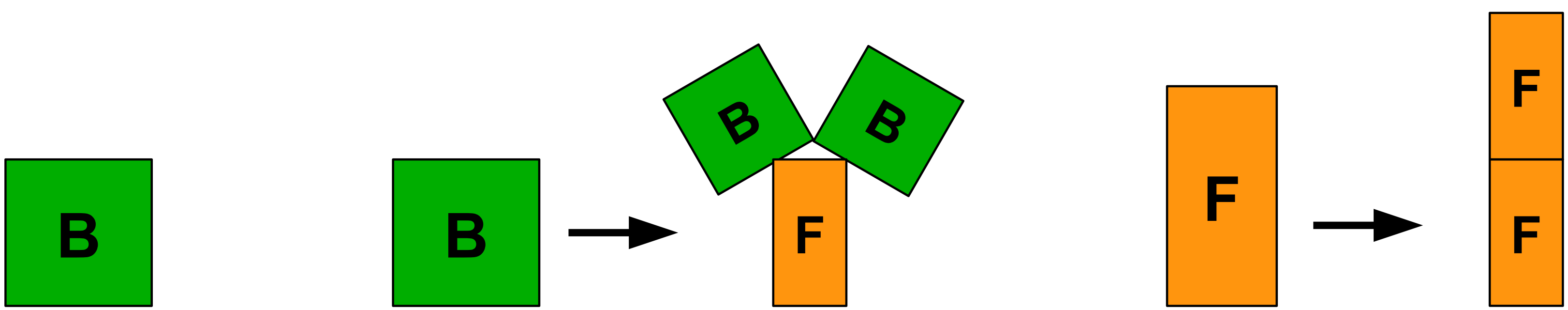
Lindenmayer Systems: Example 2

Axiom: B

Rule 1: B ⟶ F[-B]+B

Rule 2: F ⟶ FF

$\alpha = 30°$



Axiom:
B

Rule 1:
B ⟶ F[-B]+B

Rule 2:
F ⟶ FF

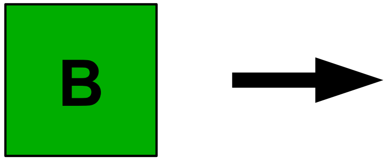
Lindenmayer Systems: Example 2

Axiom: B

Rule 1: B $\longrightarrow$ F[-B]+B

Rule 2: F $\longrightarrow$ FF

$\alpha = 30°$



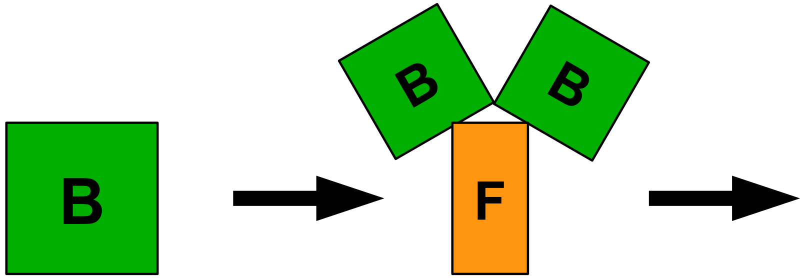
Lindenmayer Systems: Example 2

Axiom: B

Rule 1: B ⟶ F[-B]+B

Rule 2: F ⟶ FF

$\alpha = 30°$



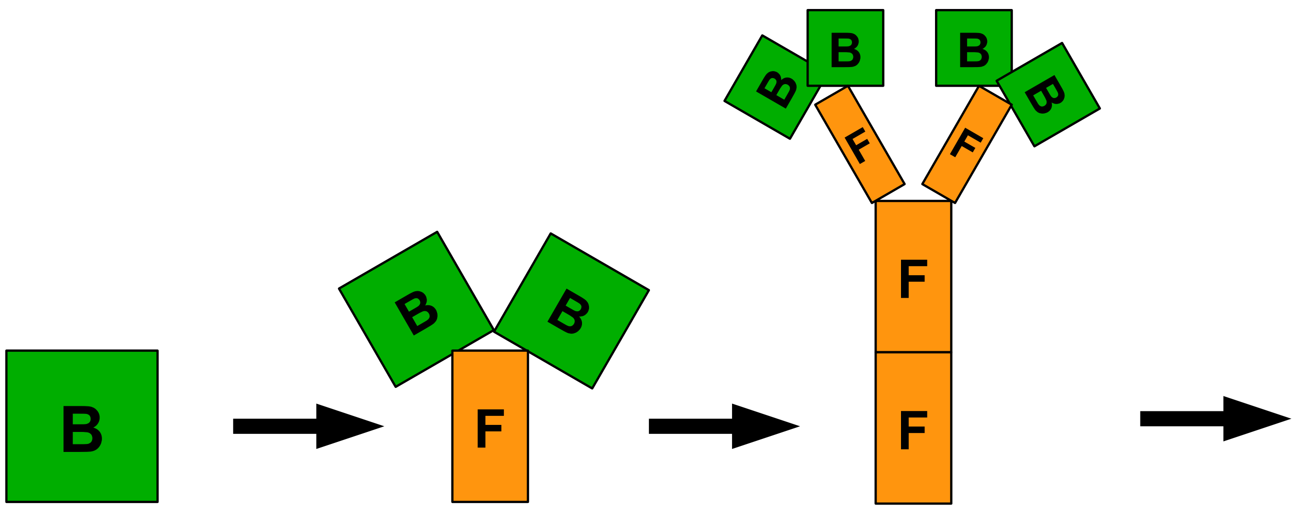
Lindenmayer Systems: Example 2

Axiom: B

Rule 1: B ⟶ F[-B]+B

Rule 2: F ⟶ FF

$\alpha = 30°$



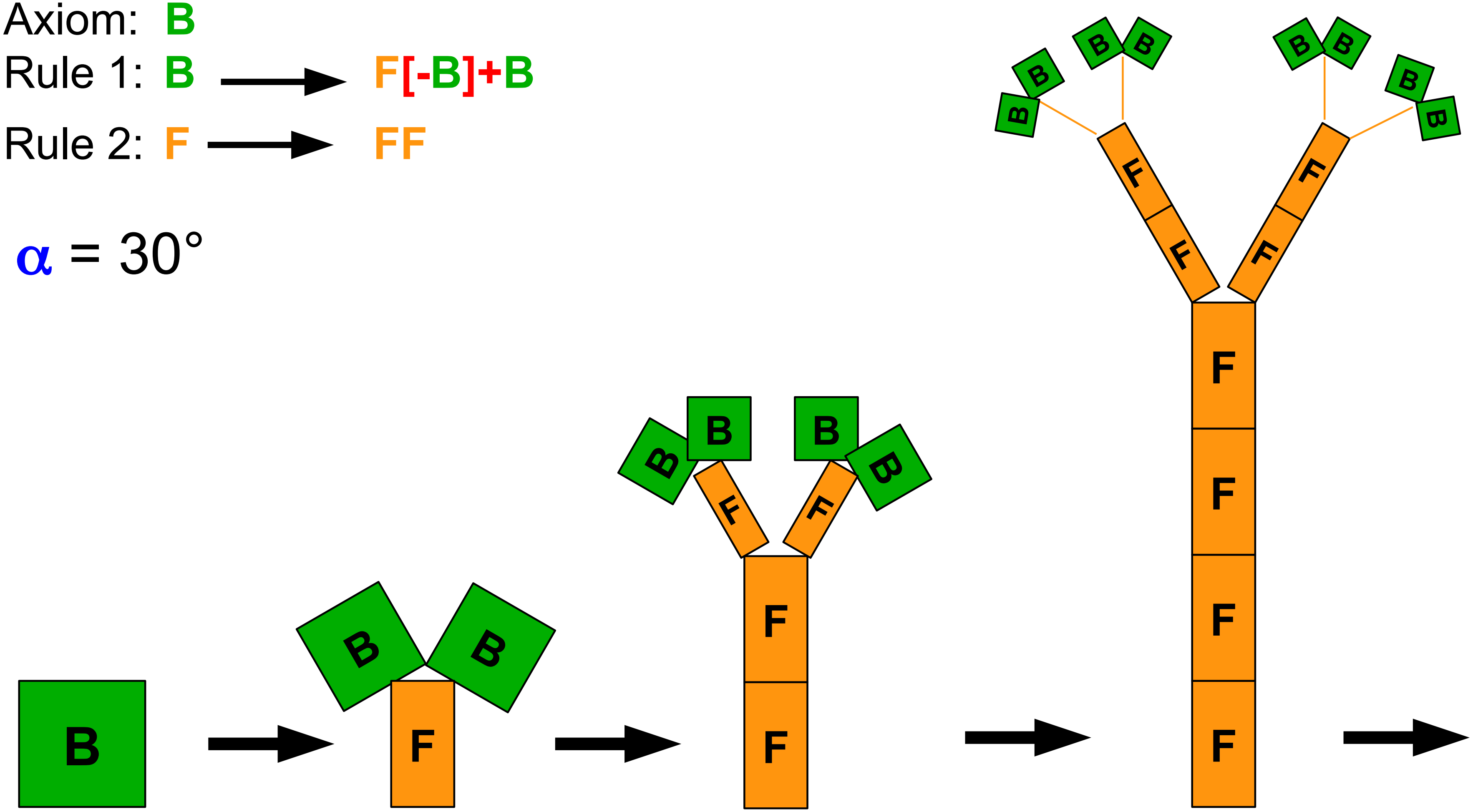
Lindenmayer Systems: Example 2

Axiom: B

Rule 1: B ⟶ F[-B]+B

Rule 2: F ⟶ FF

$\alpha = 30°$



BFB FF BFB FFFF BFB FF BFB

Lindenmayer Systems: Examples

variables: **A, B**

constants: **+ −**

axiom: **A**

rules: **A** → **B−A−B**

B → **A+B+A**

angle: $\alpha = 60°$

Lindenmayer Systems: Examples

variables: **A, B**
constants: **+ −**
axiom: **A**
rules: **A** → **B−A−B**
B → **A+B+A**
angle: α **= 60°**
step: **n**

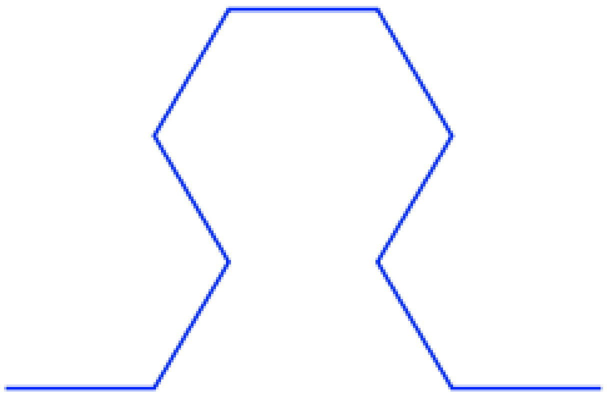
A, B mean "draw forward"

+ means "turn left by α",
− means "turn right by α"

The angle α changes sign at each iteration so that the base of the triangular shapes are always in the bottom.

Lindenmayer Systems: Examples

variables: **A, B**
constants: **+ −**
axiom: **A**
rules: **A** → **B−A−B**
B → **A+B+A**
angle: $\alpha = 60°$
step: $n = 2$



A, B mean "draw forward"

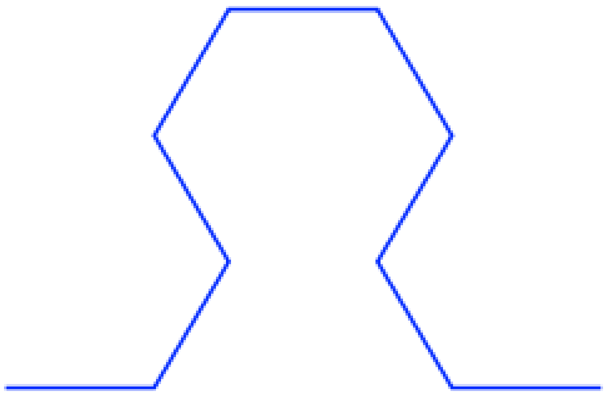
+ means "turn left by α",
− means "turn right by α"

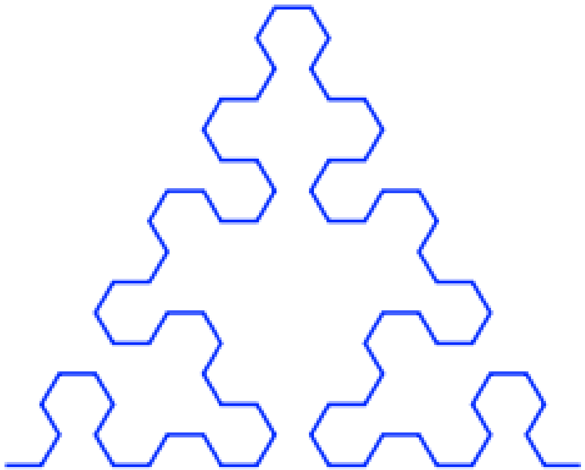
The angle α changes sign at each iteration so that the base of the triangular shapes are always in the bottom.

From:*http://en.wikipedia.org/wiki/L-system*

Lindenmayer Systems: Examples

variables: **A, B**
constants: **+ −**
axiom: **A**
rules: **A** $\rightarrow$ **B−A−B**
B $\rightarrow$ **A+B+A**
angle: α **= 60°**
steps: **n = 2, 4**





A, B mean "draw forward"

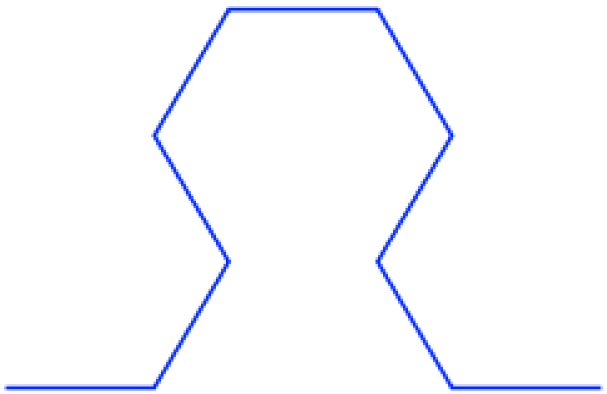
+ means "turn left by α",
− means "turn right by α"

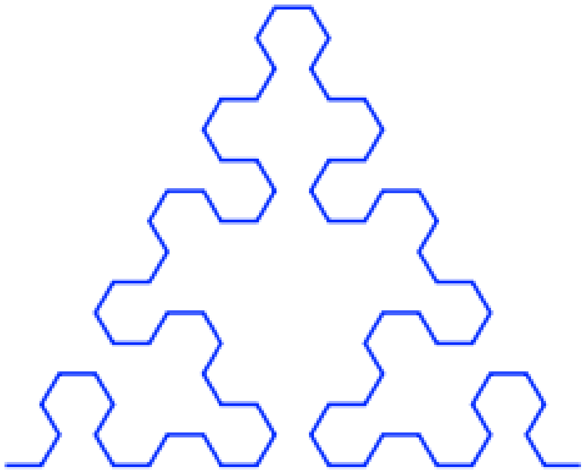
The angle α changes sign at each iteration so that the base of the triangular shapes are always in the bottom.

From:*http://en.wikipedia.org/wiki/L-system*

Lindenmayer Systems: Examples

variables: **A, B**
constants: **+ −**
axiom: **A**
rules: **A** $\rightarrow$ **B−A−B**
B $\rightarrow$ **A+B+A**
angle: α **= 60°**
steps: **n = 2, 4, 6**

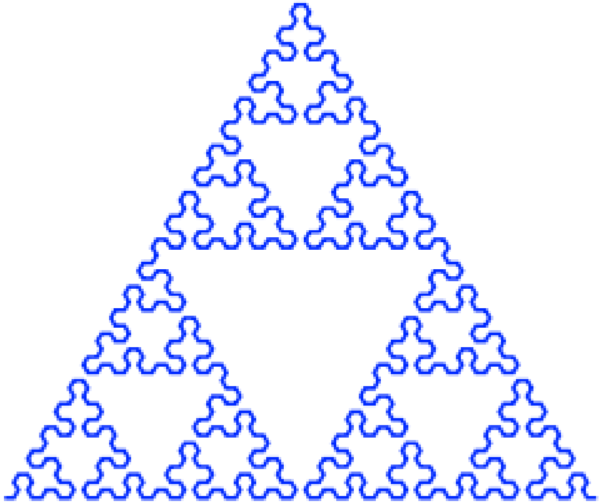




A, B mean "draw forward"

+ means "turn left by α",
− means "turn right by α"

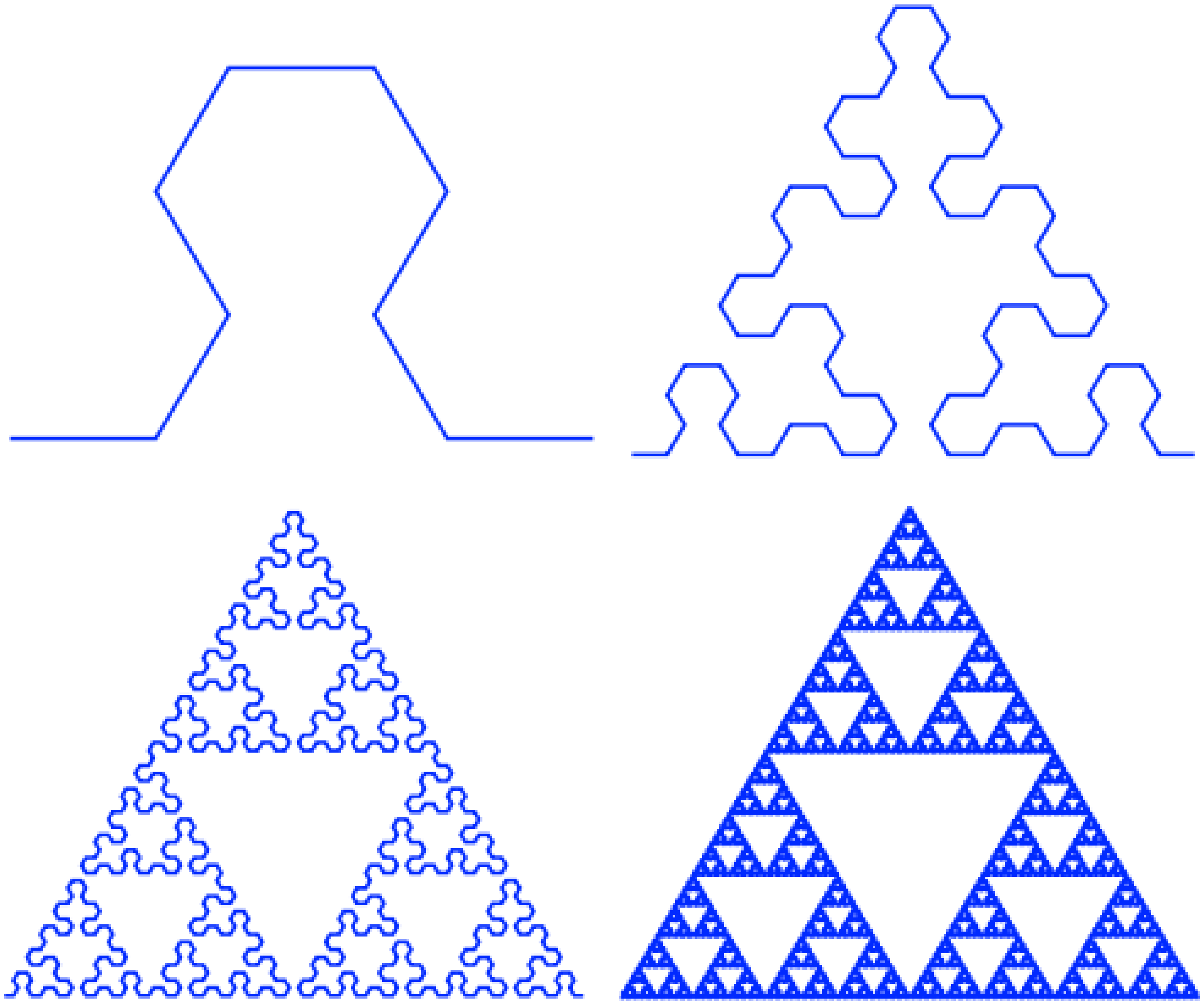
The angle α changes sign at each iteration so that the base of the triangular shapes are always in the bottom.



From:*http://en.wikipedia.org/wiki/L-system*

Lindenmayer Systems: Examples

variables: **A, B**
constants: **+ −**
axiom: **A**
rules: **A → B−A−B**
B → A+B+A
angle: α **= 60°**
steps: **n = 2, 4, 6, 8**



A, B mean "draw forward"

+ means "turn left by α",
− means "turn right by α"

The angle α changes sign at each iteration so that the base of the triangular shapes are always in the bottom.

From:*http://en.wikipedia.org/wiki/L-system*

Lindenmayer Systems: Examples

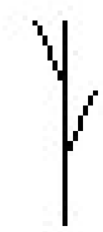
variables: **F, X**
constants: **+ −**
axiom: **X**
rules: **X → F-[[X]+X]+F[+FX]-X**
F → FF
angle: α **= 25°**
step: **n = 6**

F means "draw forward"
+ means "turn left by α",
− means "turn right by α"
[remember position
] restore position

From:*http://en.wikipedia.org/wiki/L-system*

Lindenmayer Systems: Examples

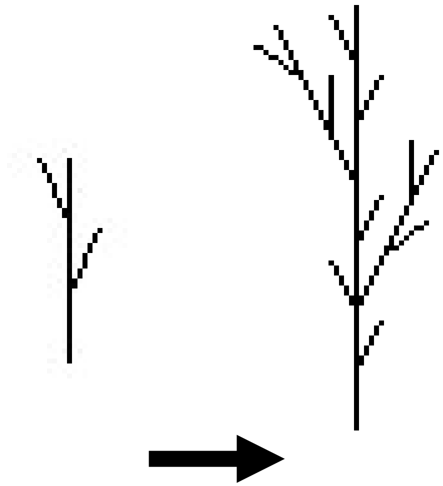
$F \rightarrow F[-F]F[+F][F]$



From:*http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html*

Lindenmayer Systems: Examples

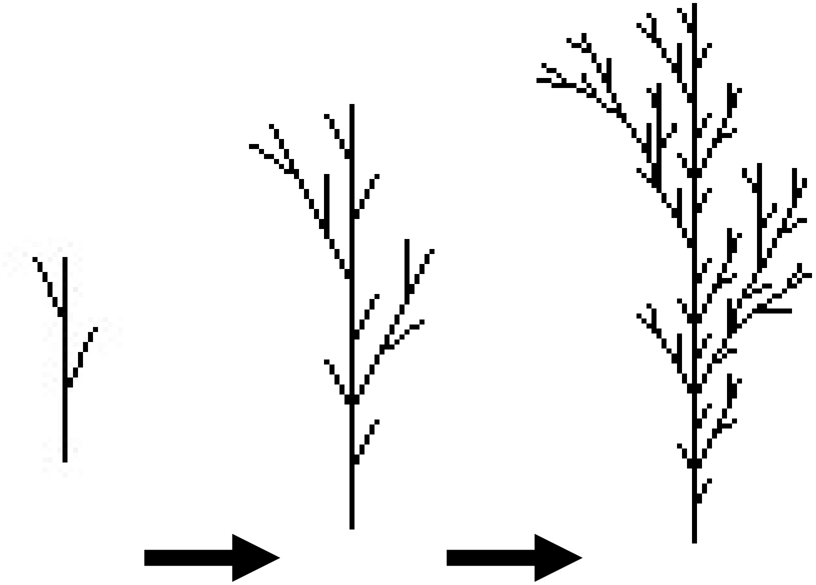
$F \rightarrow F[-F]F[+F][F]$



From:*http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html*

Lindenmayer Systems: Examples

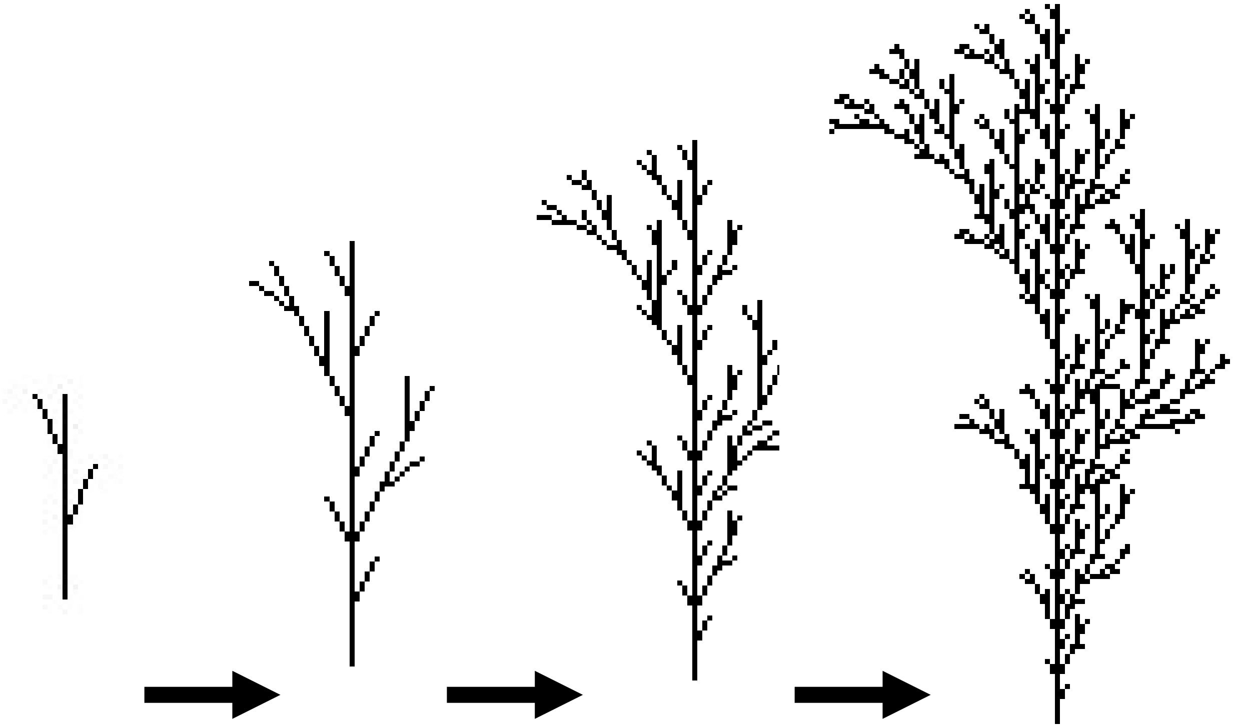
$F \longrightarrow F[-F]F[+F][F]$



From: *http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html*

Lindenmayer Systems: Examples

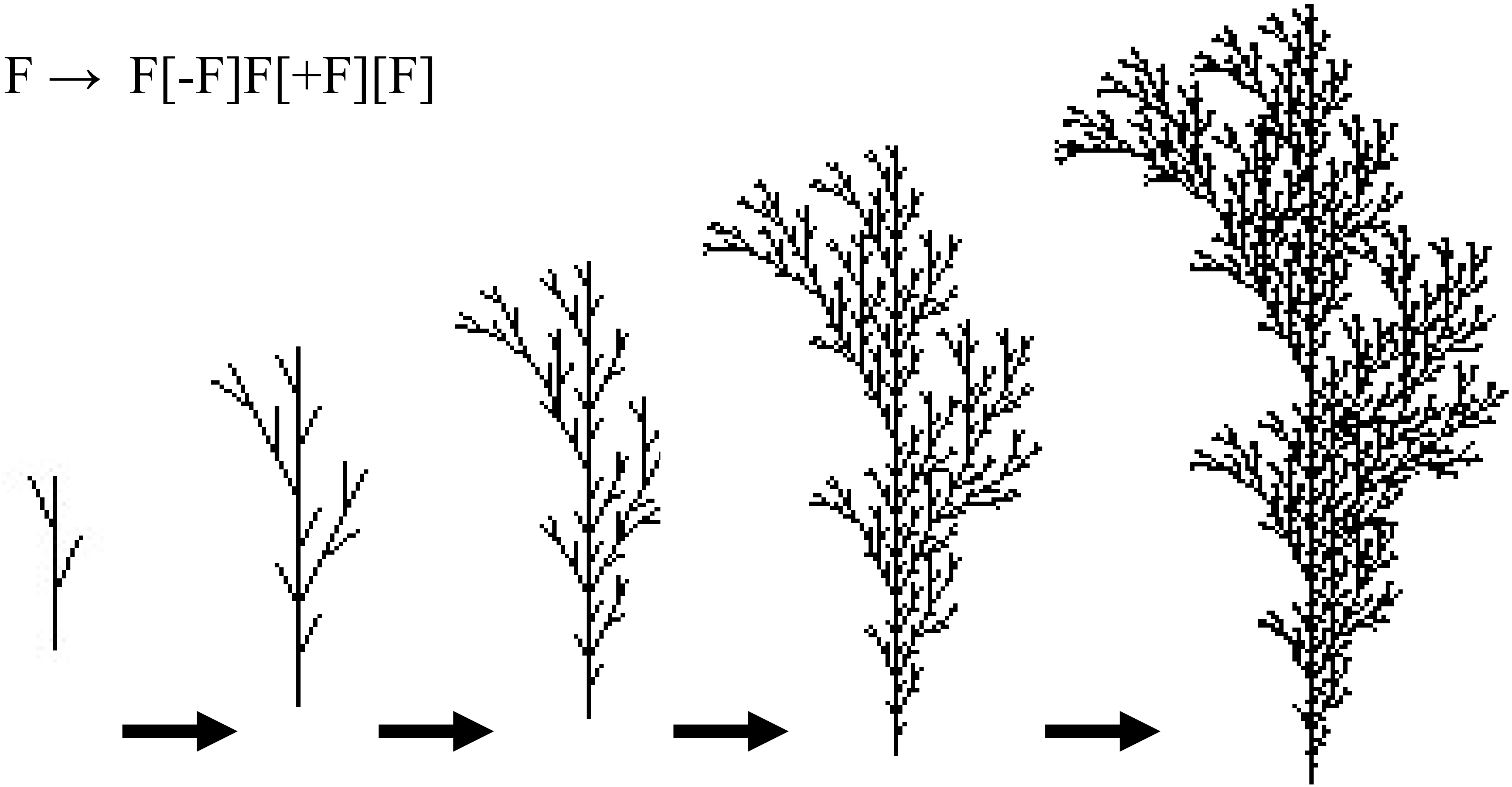
$$F \rightarrow F[-F]F[+F][F]$$



From:*http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html*

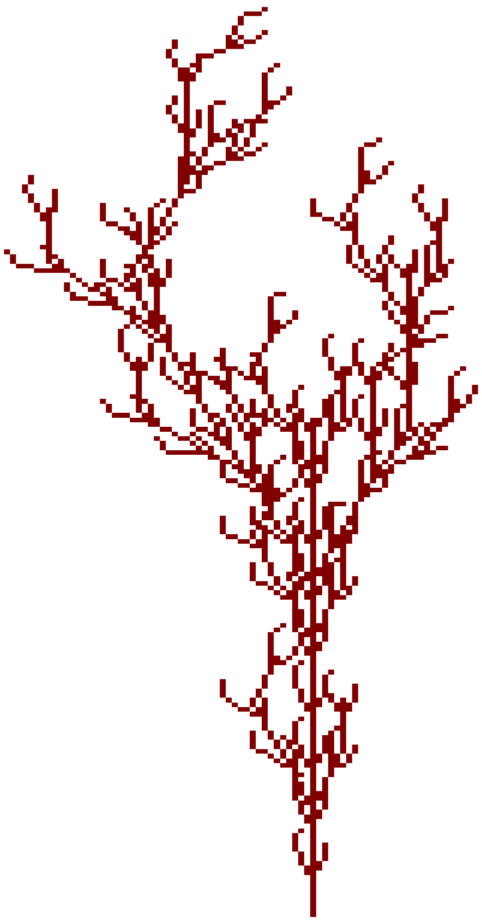
Lindenmayer Systems: Examples

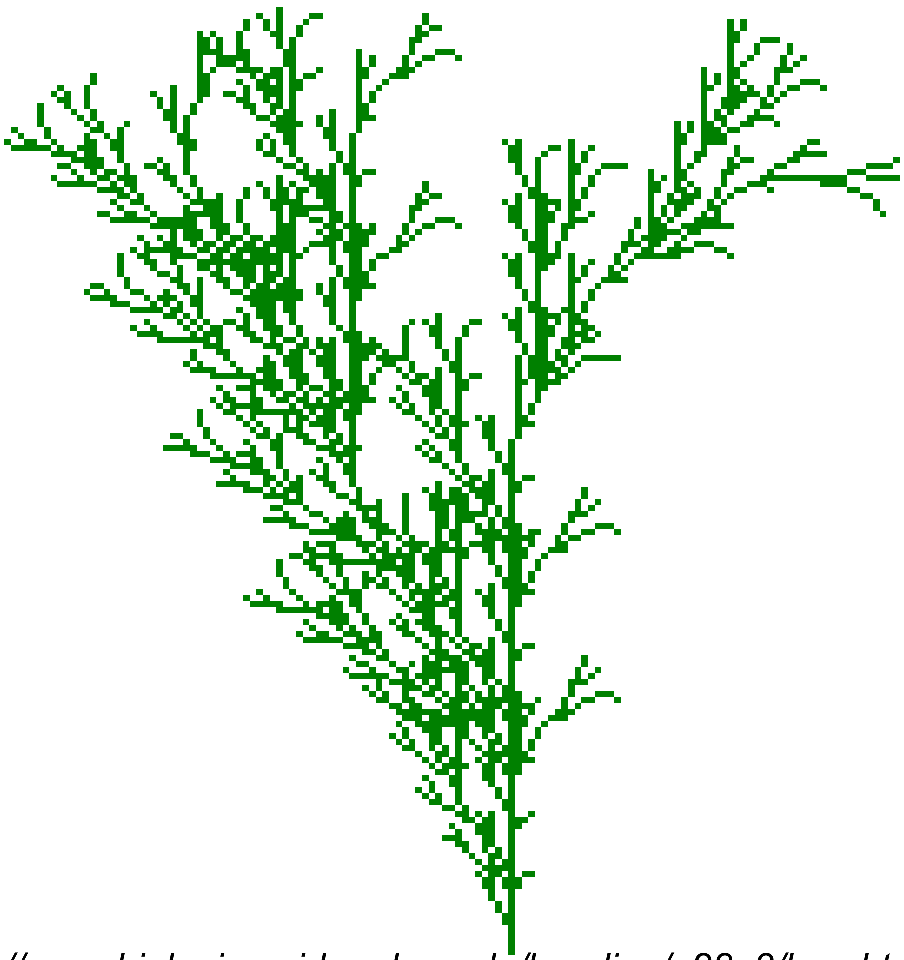
$$F \rightarrow F[-F]F[+F][F]$$



From:*http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html*

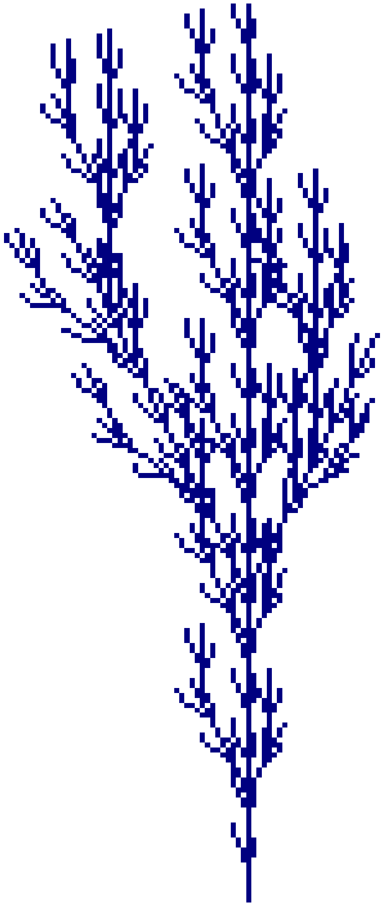
Lindenmayer Systems: Examples

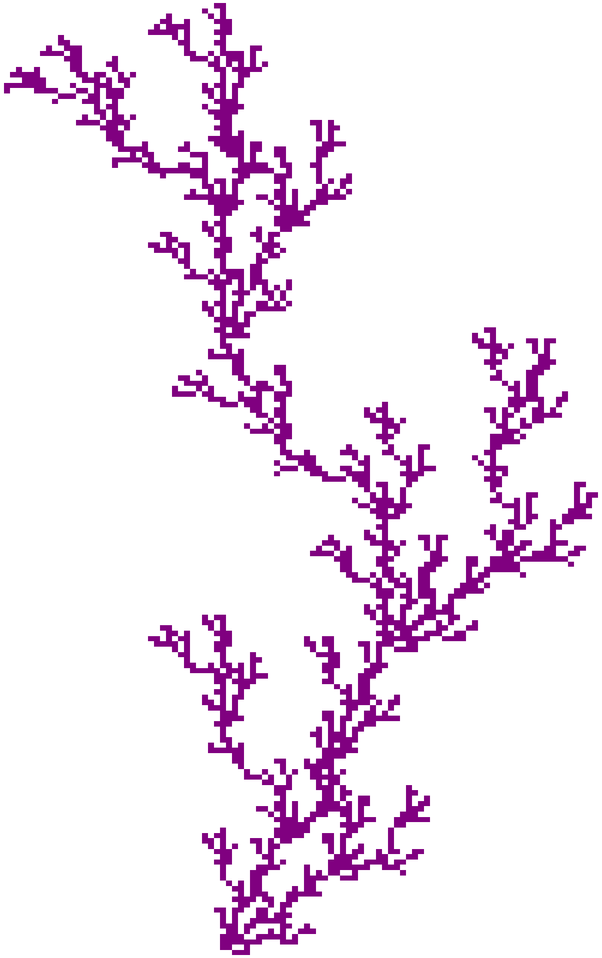


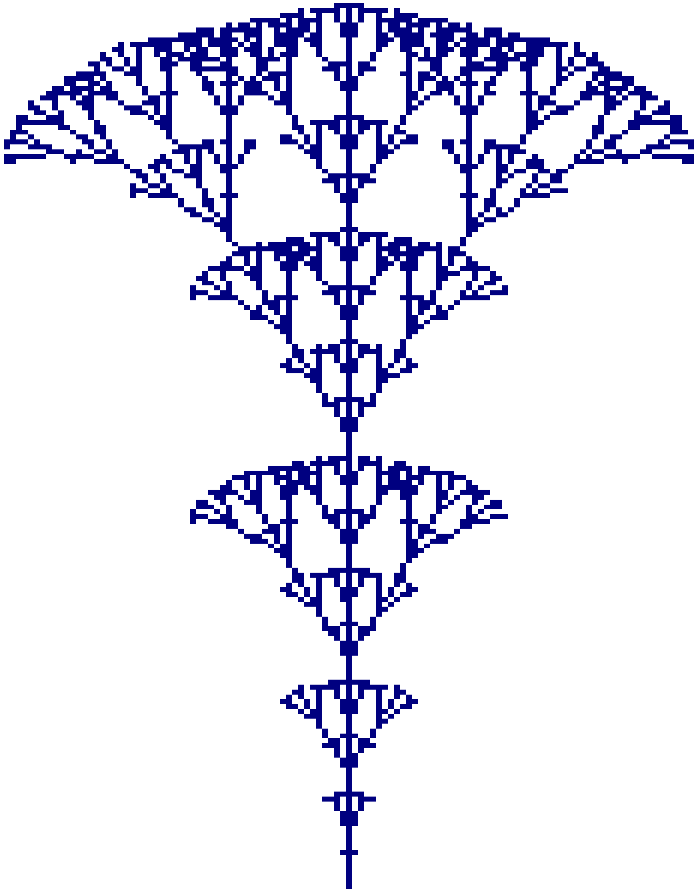


From:*http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html*

Lindenmayer Systems: Examples







From:*http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html*

Lindenmayer Systems: Examples

Lindenmayer Systems: Examples

Lindenmayer Systems: Examples

n=4, d=8, α=30°
Axiom: T

T $\longrightarrow$ R+[T]--[--L]R[++L]-[T]++T

R $\longrightarrow$ F[--L][++L]F

L $\longrightarrow$ [{+FX-FX-FX+I+FX-FX-FX}]

FX $\longrightarrow$ FX

F $\longrightarrow$ FF

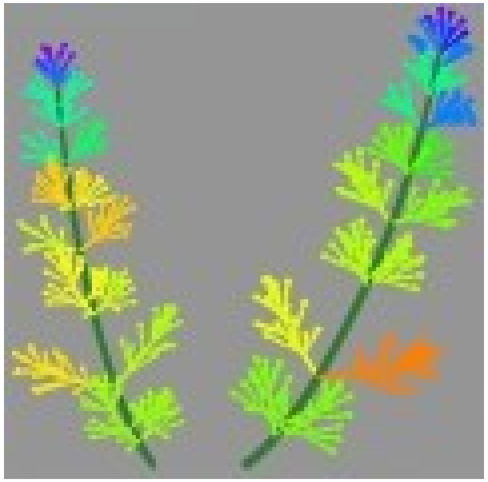
Fig 4. Example of a plant generated by a pL-system.
Parentheses are grouping edges which define boundaries of filled polygons.

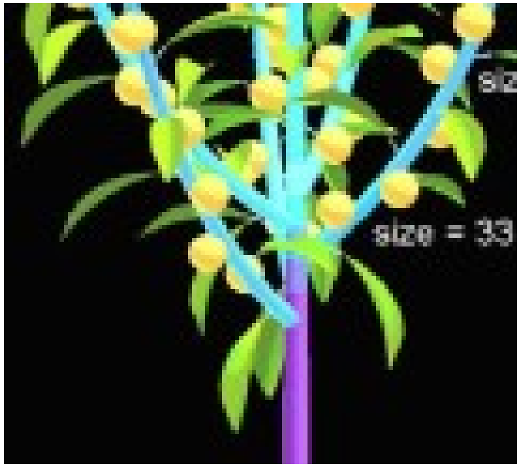
P. Prusinkiewicz: Graphical applications of L−systems.
Proceedings of Graphics Interface '86 / Vision Interface '86, pp. 247−253.
http://algorithmicbotany.org/papers/graphical.gi86.pdf

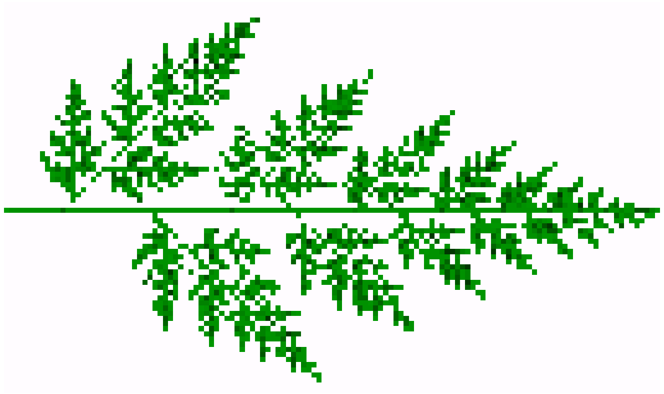
Lindenmayer Systems: Examples

From: Przemyslaw Prusinkiewicz, Aristid Lindenmayer: *The Algorithmic Beauty of Plants*
http://algorithmicbotany.org/papers/#abop

Lindenmayer Systems: Examples







From: *http://algorithmicbotany.org/papers/*

Lindenmayer Systems: Extensions

Several extensions to the original D0L-Systems have been proposed meanwhile:

- Bracketed L-Systems
- Visualization in higher dimensions (2-dim or 3-dim)
- Context depenent L-Systems
- Stochastic/Probabilistic L-Systems
- Parametric L-Systems
- … and others.

Lindenmayer Systems: Applications

Today applications of Lindenmayer Systems can be found in several areas; some examples are here:

- Theoretical Biology
- Computer Graphics
- Modeling of Buildings, Architecture
- Computer Music
- … and others.

Theoretical Biology:

G.S.Hornby, J.B.Pollack: *Evolving L-systems to generate virtual creatures*, Computers & Graphics, Volume 25, Issue 6, December 2001, Pages 1041-1048

Hansrudi Noser: *A Behavioral Animation System Based on L-systems and Synthetic Sensors for Actors, PhD Thesis,* 1609 Lausanne, EPFL, (1997),

Computer Graphics:

Przemyslaw Prusinkiewicz, Aristid Lindenmayer, and James Hanan: *Developmental Models of Herbaceous Plants for Computer Imagery Purposes*. Computer Graphics 22(4), pp. 141-150, 1988.

Przemyslaw Prusinkiewicz, *Applications of L-systems to computer imagery, LNCS: Volume 291/1987, pp.534-548*

O.Terraz, G.Guimberteau, S.Mérillou, D.Plemenos, D.Ghazanfarpour: *3Gmap L-systems: an application to the modeling of wood,* The Visual Computer, Vol 25, No. 2 / Feb. 2009

Architecture:

P.Muller, P.Wonka, S.Haegler, A.Ulmer, L.Van Gool: *Procedural modeling of buildings,* ACM transactions on graphics vol:25 issue:3 pages:614-623 (2006)

Michael Hansmeyer, *Computational Architecture*, Website,http//michael-hansmeyer.com

Computer Music:

P.Worth, and S.Stepney: *Growing Music: Musical Interpretations of L-Systems,* LNCS, Volume 3449/2005, pp.545-550 (2005)

Stelios Manousakis: *Musical L-Systems,* Master's Thesis – Sonology, The Royal Conservatory, The Hague, June 2006, Den Haag, Netherlands

Overview:

- Self-Replication
- Langton's Self-Replicating Loop
- Lindenmayer Systems

Artificial Life
Summer 2025

Self Replication
Langton's Loop
Lindenmayer Systems

Master Computer Science [MA-INF 4201]
Mon 14c.t. – 15:45, HS-2

Dr. Nils Goerke, Autonomous Intelligent Systems,
Department of Computer Science, University of Bonn

Artificial Life
Summer 2025

Self Replication
Langton's Loop
Lindenmayer Systems

Thank you for listening

Dr. Nils Goerke, Autonomous Intelligent Systems,
Department of Computer Science, University of Bonn