13. Cellular automata

Generative Music AI

THE SOUND OF AI

upf. Universitat Pompeu Fabra Barcelona

MTG Music Technology Group

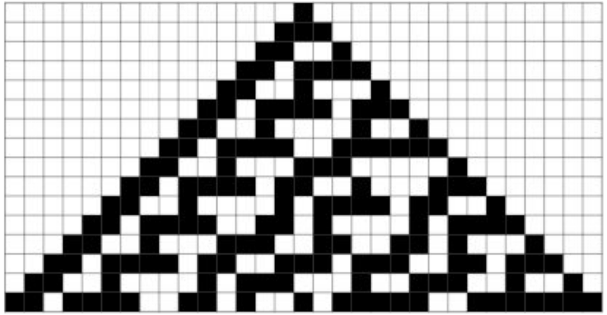
Overview

1. Intuition
2. Formalisation
3. Music generation with CA
4. Strengths and limitations

What’s a cellular automaton (CA)?

Cellular automata are models used to simulate complex systems using rules on a grid of cells

What’s a cellular automaton (CA)?



What’s a cellular automaton (CA)?

- CA evolve in discrete time steps

What’s a cellular automaton (CA)?

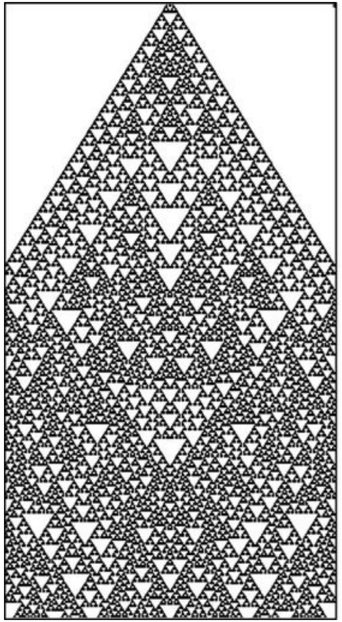
- CA evolve in discrete time steps
- Cells change state based on their own and neighbors' states

What’s a cellular automaton (CA)?

- CA evolve in discrete time steps
- Cells change state based on their own and neighbors' states
- Complex patterns emerge from simple rules

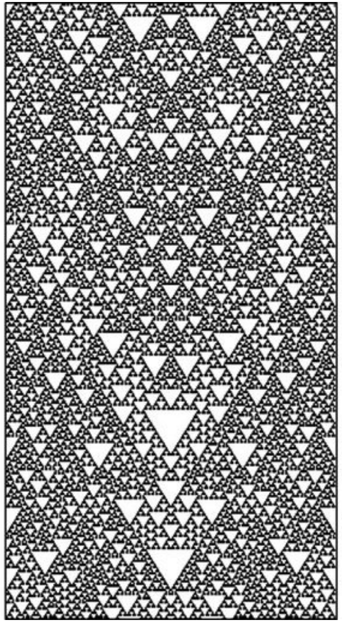
What’s a cellular automaton (CA)?

step 1



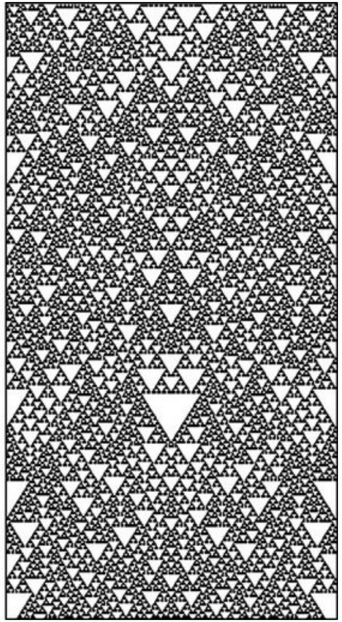
What’s a cellular automaton (CA)?

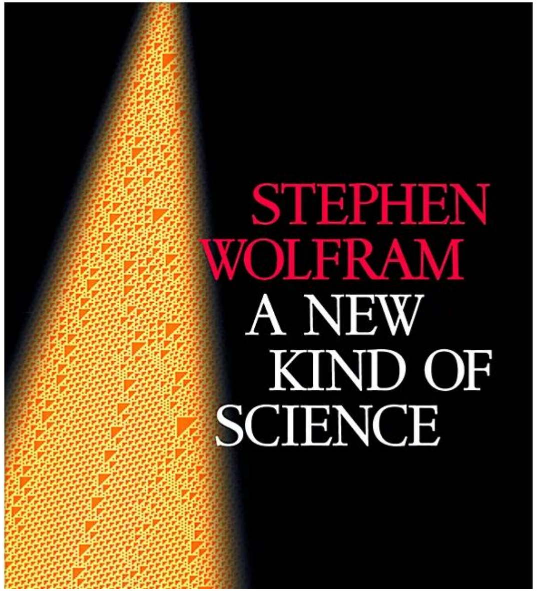
step 2

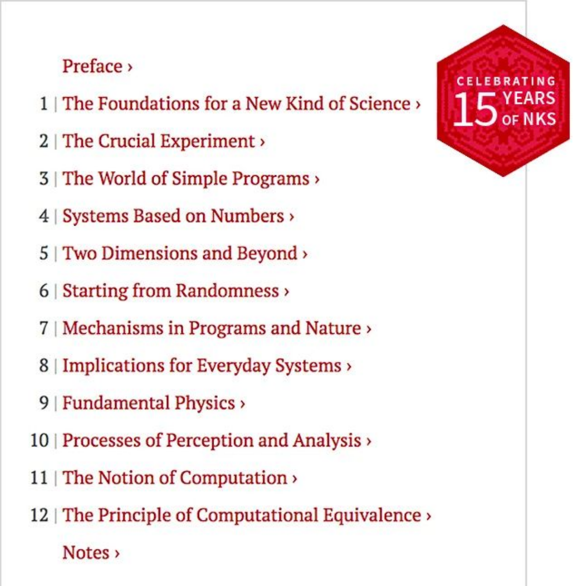


What’s a cellular automaton (CA)?

step 3



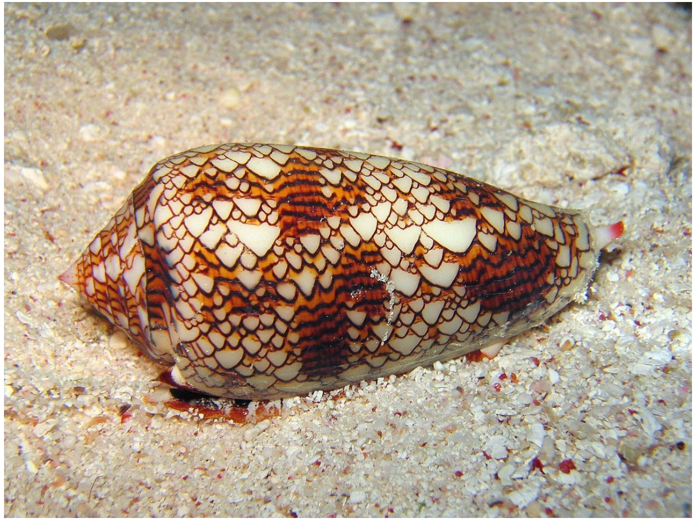

STEPHEN
WOLFRAM
A NEW
KIND OF
SCIENCE


CELEBRATING
15 YEARS
OF NKS

CA and nature

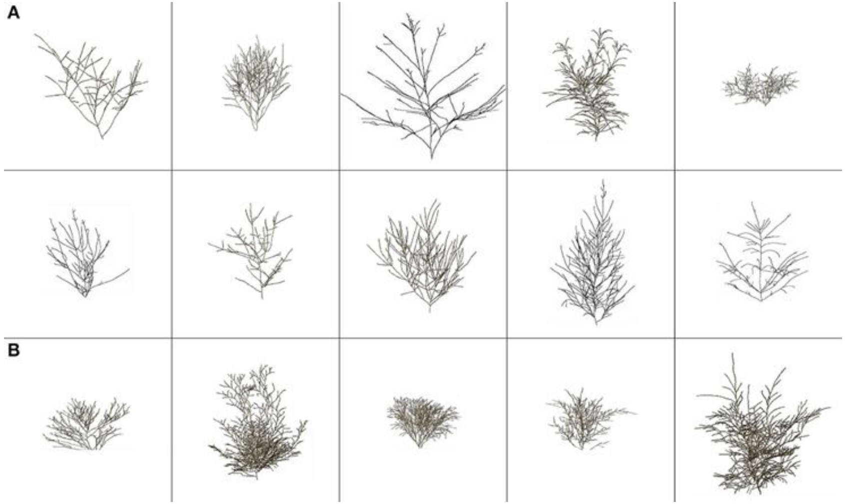
CA and nature

- Patterns on seashells



CA and nature

- Patterns on seashells
- Branching patterns in plants



CA and nature

- Patterns on seashells
- Branching patterns in plants
- Crystal growth

CA formalisation

CA formalisation

- *Grid*: line of cells (1D), plane of cells (2D)

CA formalisation

- *Grid*: line of cells (1D), plane of cells (2D)
- *Cell*: each cell is identified by its row + column position

CA formalisation

- *Grid*: line of cells (1D), plane of cells (2D)
- *Cell*: each cell is identified by its row + column position
- *States*: each cell can be in one of a finite number of states

CA formalisation

- *Grid*: line of cells (1D), plane of cells (2D)
- *Cell*: each cell is identified by its row + column position
- *States*: each cell can be in one of a finite number of states
- *Neighborhood*: set of cells around cell whose states influence the cell's next state

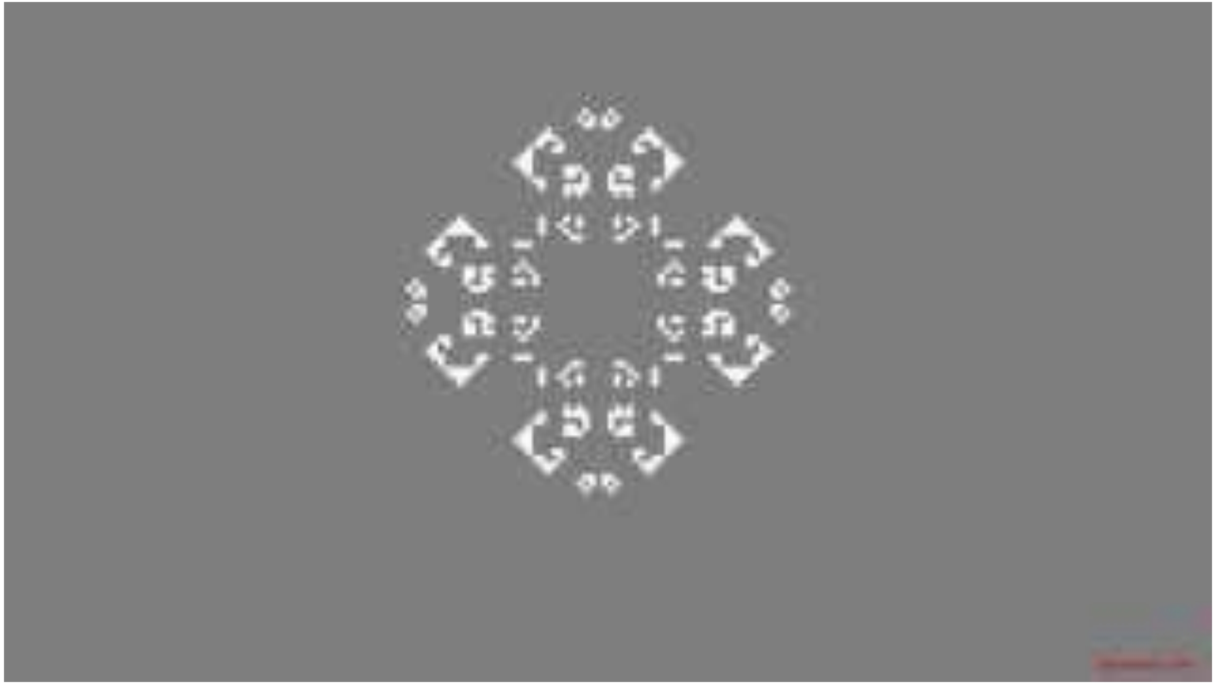
CA formalisation

- *Grid*: line of cells (1D), plane of cells (2D)
- *Cell*: each cell is identified by its row + column position
- *States*: each cell can be in one of a finite number of states
- *Neighborhood*: set of cells around cell whose states influence the cell's next state
- *Transition rules*:
    a. dictate how the state of a cell changes
    b. functions of the states of the cell and its neighbors at time *t* to determine the state at time *t+1*

CA formalisation

- *Grid*: line of cells (1D), plane of cells (2D)
- *Cell*: each cell is identified by its row + column position
- *States*: each cell can be in one of a finite number of states
- *Neighborhood*: set of cells around cell whose states influence the cell's next state
- *Transition rules*:
  1. dictate how the state of a cell changes
  2. functions of the states of the cell and its neighbors at time *t* to determine the state at time *t+1*
- *Initial conditions*: initial states of the grid (e.g., random, uniform, criteria)

Conway’s Game of Life



Conway’s Game of Life

- *Grid*: 2D

Conway’s Game of Life

- *Grid*: 2D
- *States*: alive (1) or dead (0)

Conway’s Game of Life

- *Grid*: 2D
- *States*: alive (1) or dead (0)
- *Neighborhood*: 8 surrounding cells

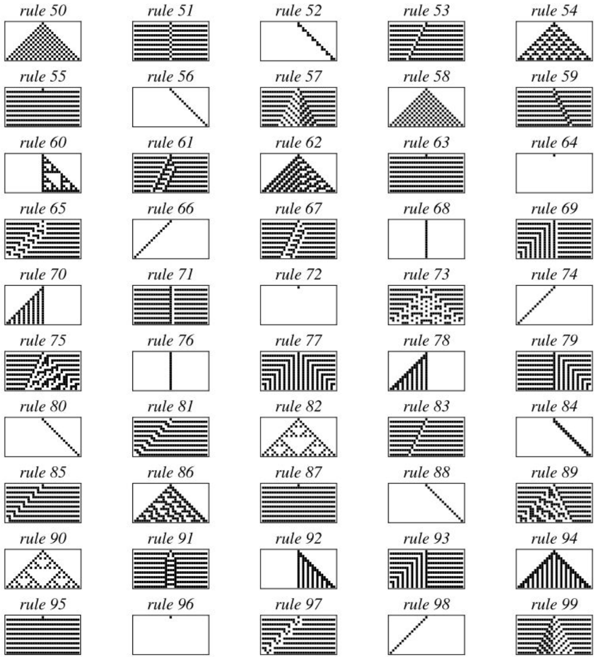
Conway’s Game of Life

- *Grid*: 2D
- *States*: alive (1) or dead (0)
- *Neighborhood*: 8 surrounding cells
- *Transition rules*:
  a. Birth: A dead cell (0) becomes alive (1) at t+1 if exactly three of its neighbors are alive
  b. Survival: A living cell (1) stays alive at t+1 if two or three of its neighbors are alive
  c. Death: In all other cases, a cell is dead at t+1

Conway’s Game of Life

- *Grid*: 2D
- *States*: alive (1) or dead (0)
- *Neighborhood*: 8 surrounding cells
- *Transition rules*:
    a. Birth: A dead cell (0) becomes alive (1) at t+1 if exactly three of its neighbors are alive
    b. Survival: A living cell (1) stays alive at t+1 if two or three of its neighbors are alive
    c. Death: In all other cases, a cell is dead at t+1
- *Initial conditions*: random or by design

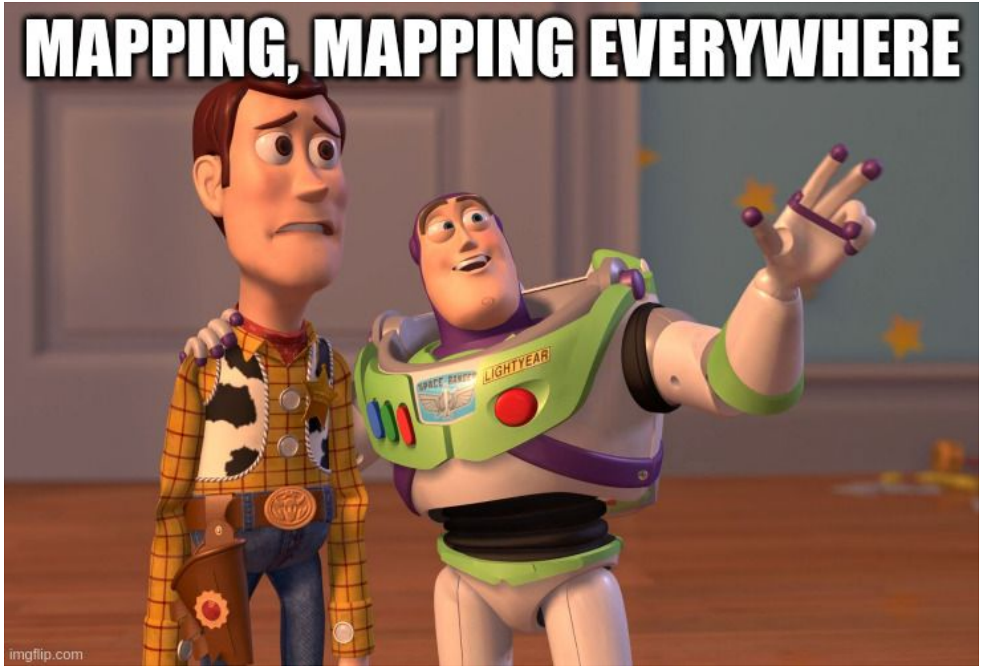
Want more CA?



Elementary cellular automaton

- 256 rules (Wolfram)

CA for music generation



CA for music generation

1. Map axes to different musical params
   (e.g., pitch, instruments, time)

CA for music generation

1. Map axes to different musical params (e.g., pitch, instruments, time)
2. Assign states to musical events (e.g., on / off, pitches)

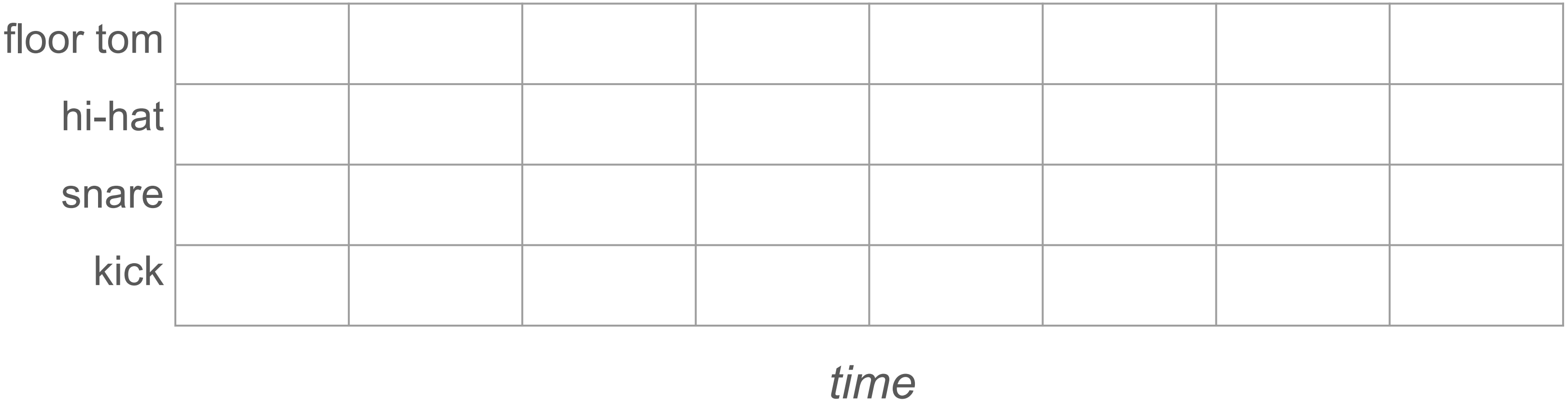
CA for music generation

1. Map axes to different musical params (e.g., pitch, instruments, time)
2. Assign states to musical events (e.g., on / off, pitches)
3. Design rules for musical evolution - may or may not be music-based rules

CA for music generation

1. Map axes to different musical params (e.g., pitch, instruments, time)
2. Assign states to musical events (e.g., on / off, pitches)
3. Design rules for musical evolution - may or may not be music-based rules
4. Map time (e.g., 1 beat = 1 step)

CA for drum generation



floor tom

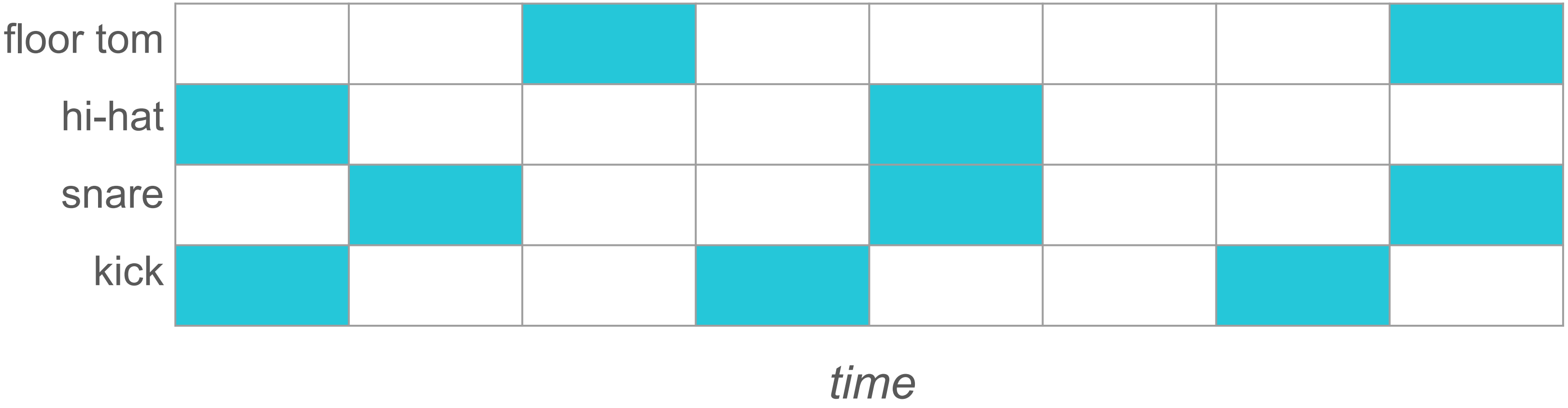
hi-hat

snare

kick

time

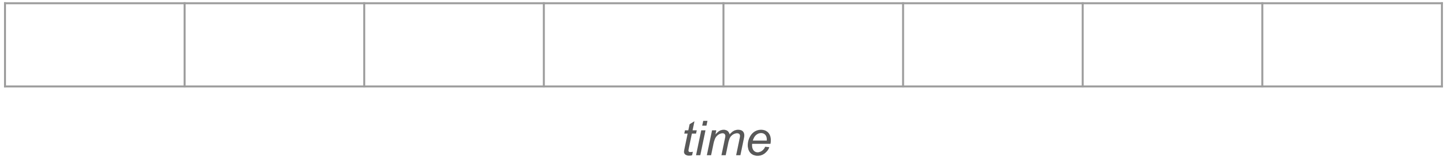
CA for drum generation



floor tom			■					■
hi-hat	■				■			
snare		■			■			■
kick	■			■			■	

CA for melody generation

$$States = \{C, D, E, F, G, A, None\}$$



CA for melody generation

$$States = \{C, D, E, F, G, A, None\}$$

C		E		F	D	C	

time

CA for expressive chord generation

$$States = \{pp, p, mf, f, ff, None\}$$

synth					
piano					
organ					
	C	D	E	G	A

CA for expressive chord generation

$$States = \{pp, p, mf, f, ff, None\}$$

step 1 = beat 1

	C	D	E	G	A
synth	*pp*				*p*
piano		*f*		*p*	
organ			*ff*		*mf*

CA for expressive chord generation

$$States = \{pp, p, mf, f, ff, None\}$$

step 2 = beat 2

synth			*pp*		
piano		*mf*		*p*	*p*
organ	*ff*			*f*	
	C	D	E	G	A

Music strategies for CA

- Generate entire score

Music strategies for CA

- Generate entire score
- Guideline for improvisation

Music strategies for CA

- Generate entire score
- Guideline for improvisation
- Integrate CA-generated instrumentation into a composition

Pros and cons of CA

+

- Flexible
- Experimentation
- OK for raw material

Pros and cons of CA

+

- Flexible
- Experimentation
- OK for raw material

−

- Bad musical output
- No music knowledge

Key takeaways

- CA simulate complex behaviour on a grid

Key takeaways

- CA simulate complex behaviour on a grid
- Complex patterns emerge from simple rules

Key takeaways

- CA simulate complex behaviour on a grid
- Complex patterns emerge from simple rules
- Grid, cell, states, neighborhood, rules and initial conditions

Key takeaways

- CA simulate complex behaviour on a grid
- Complex patterns emerge from simple rules
- Grid, cell, states, neighborhood, rules and initial conditions
- Mapping is key to use CA for music generation

Key takeaways

- CA simulate complex behaviour on a grid
- Complex patterns emerge from simple rules
- Grid, cell, states, neighborhood, rules and initial conditions
- Mapping is key to use CA for music generation
- States and grid mapped to different musical params

Key takeaways

- CA simulate complex behaviour on a grid
- Complex patterns emerge from simple rules
- Grid, cell, states, neighborhood, rules and initial conditions
- Mapping is key to use CA for music generation
- States and grid mapped to different musical params
- CA can be used for many generation tasks

Key takeaways

- CA simulate complex behaviour on a grid
- Complex patterns emerge from simple rules
- Grid, cell, states, neighborhood, rules and initial conditions
- Mapping is key to use CA for music generation
- States and grid mapped to different musical params
- CA can be used for many generation tasks
- Music output is OK as raw material

What’s up next?

Drum generation with cellular automata