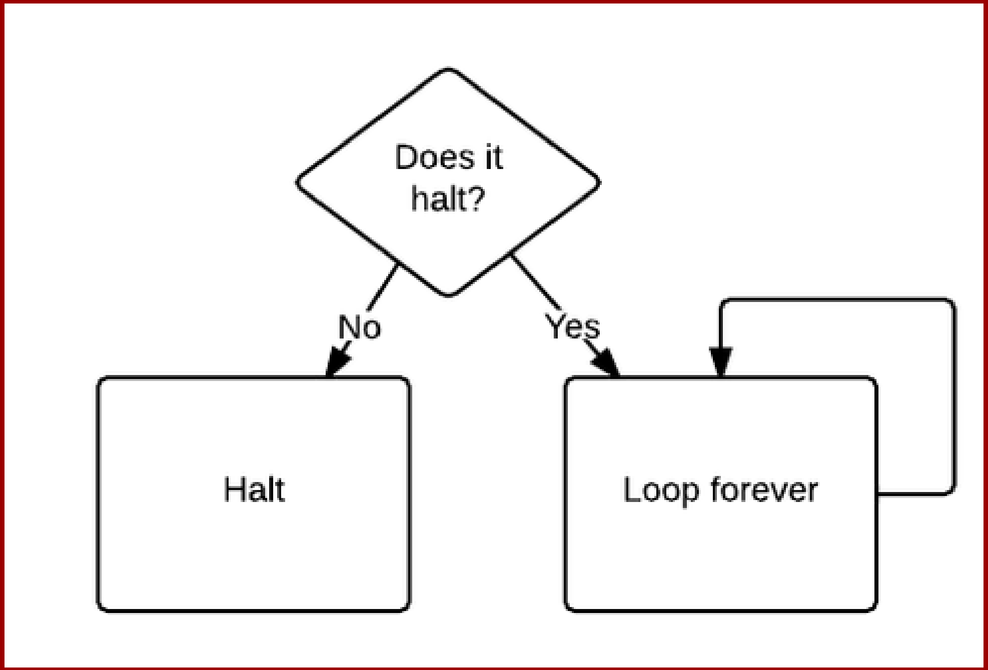
15-251: Great Theoretical Ideas in Computer Science

Lecture 7

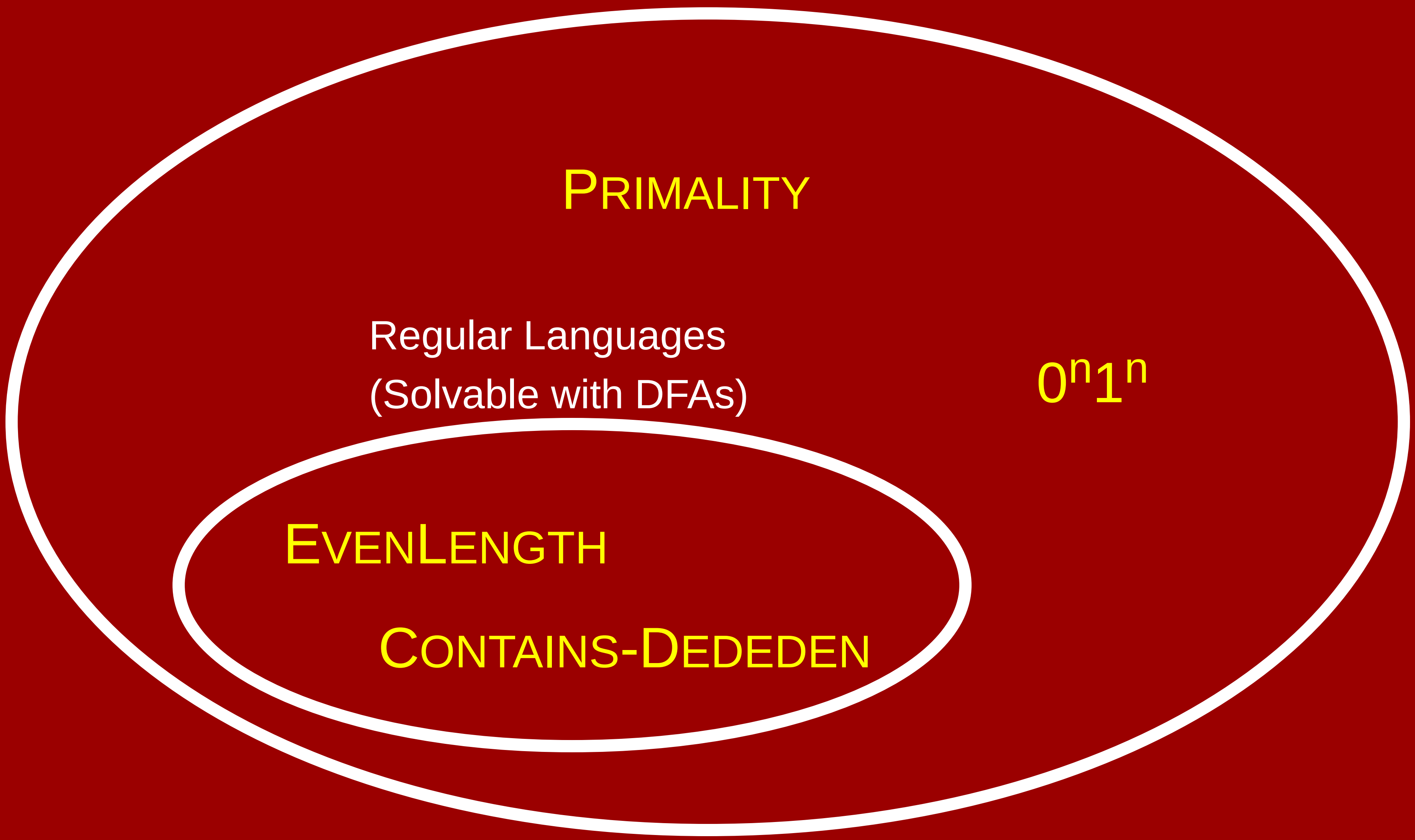
Turing's Legacy Continues



Solvable with Python
= Solvable with C
= Solvable with Java
= Solvable with SML

=

Decidable Languages
(decidable by Turing Machienes)



Robustness of Decidability

Decidability power is the same for TMs with:

- one-sided or double-sided infinite tape
- ability to stay in addition to going left / right
- even a fixed (oblivious) moving pattern works
- binary or larger finite tape alphabet
- one tape or a finite number of tapes/heads

Decidability power is also the same as:

- Python, C, Java, Assembly (any other language)
- Random Access Machiene + other comp. models
- Lambda-Calculus

Side note: Efficiency

Model details (and encodings) do play a role when it comes to efficiency, e.g., how many computation steps are needed.

Examples:

- a TM with one tape can simulate any multi-tape TM with a quadratic slowdown (sometimes needed)
- Random Access Machines can be simulated by a multi-tape TM with logarithmic slowdown
- Quantum computation can be simulated with exponential slowdown. It is unknown whether a super-polynomial slowdown is needed)

Robustness of Decidability

Most computational models, including those abstracted from any natural phenomenon, tend to be either wimpy or Turing equivalent, i.e., exactly equivalent in computational power to TMs.

No candidates of potentially implementable / natural computational models that are more powerful than a TM have been suggested.

Church–Turing Thesis (1936):

“Any natural / reasonable notion of computation can be simulated by a TM.”

Cellular Automata

Most systems / the world can be described as many (tiny) parts interacting with other close-by parts.

Formal computational model:

A Cellural automaton (CA) consists of:

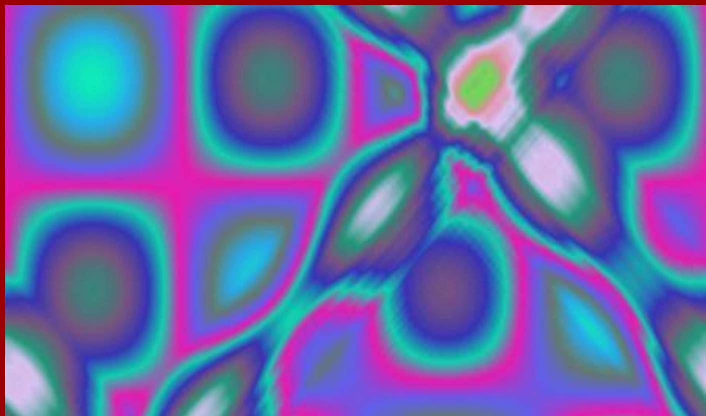
- cells with a finite set of states Q
- a neighborhood relation between cells
- a transition function $\delta_v: Q^{deg(v)+1} \longrightarrow Q$

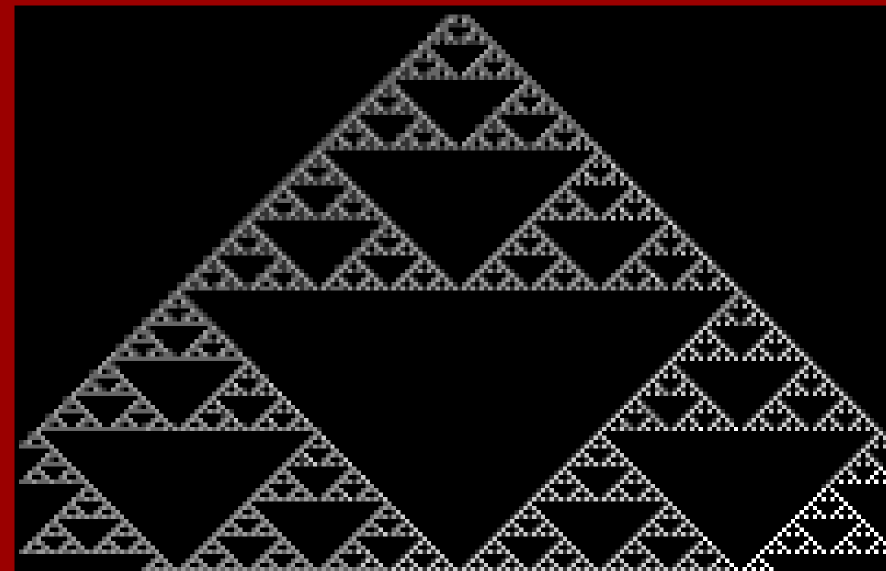
Computation: In every round every cell v (synchronously) transitions its state according to δ_v based on its and its neighbors' state.

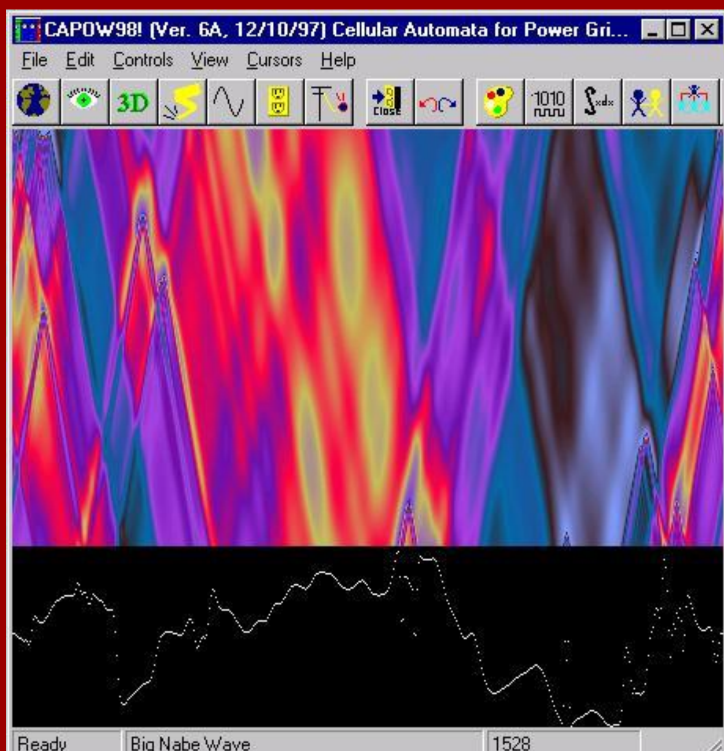
Applications of Cellular Automata

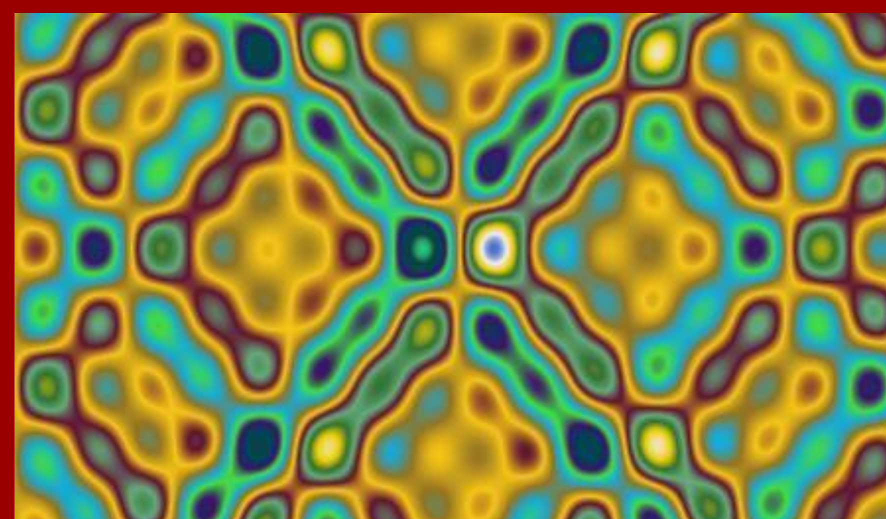
- Simulation of Biological Processes
- Simulation of Cancer cells growth
- Predator – Prey Models
- Art
- Simulation of Forest Fires
- Simulations of Social Movement
- …many more..

Cellular Automata: Examples





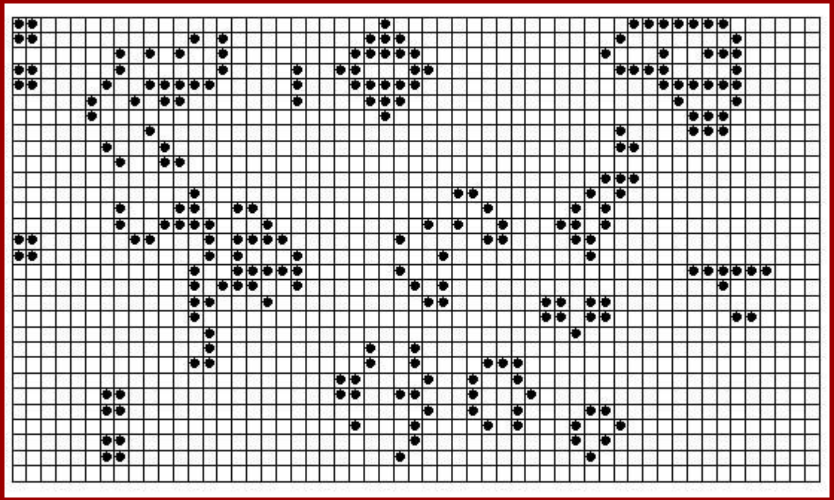




Example CA: Conway's Game of Life

Example CA: Conway's Game of Life

Cells form the infinite 2D-Grid



Q = {alive,dead}

3 transition rules ($\delta: Q^9 \rightarrow Q$):

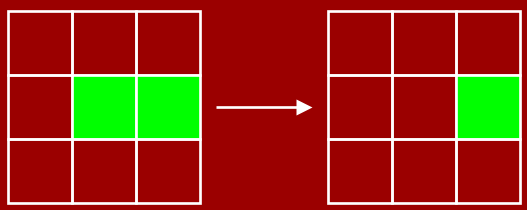
Loneliness: Life cell with fewer than 2 neighbors dies.

Overcrowding: Life cell with at least 4 life neighbors dies.

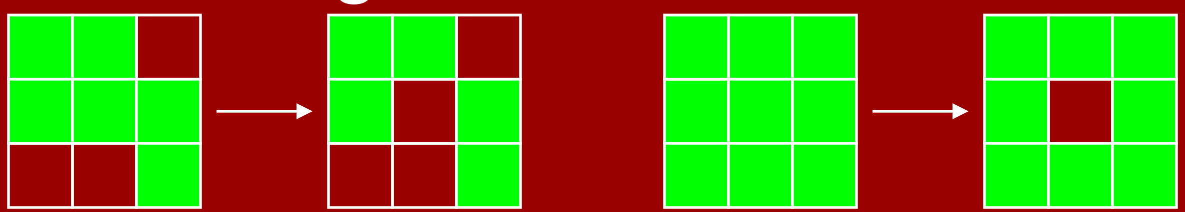
Procreation: Dead cell with exactly 3 neighbors gets born.

Conway's Game of Life: Rule examples

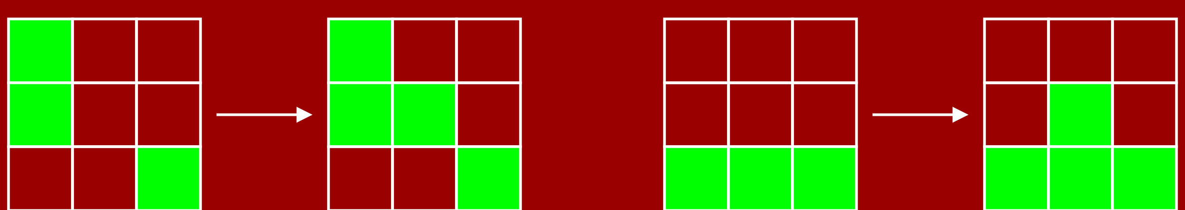
- **loneliness**



- **overcrowding**

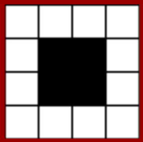


- **procreation**

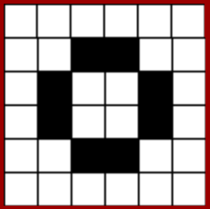


Conway's Game of Life: Patterns

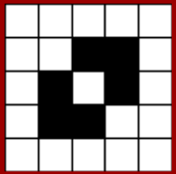
Stable



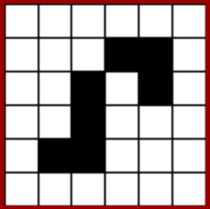
block



pond

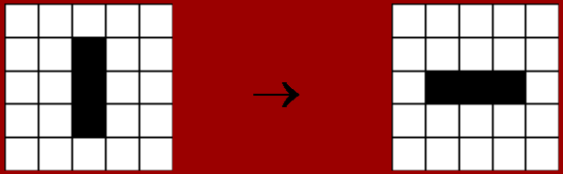


ship



eater

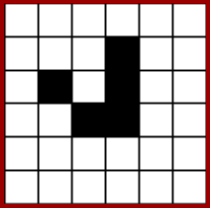
Periodic



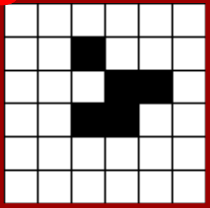
time = 1

time = 2

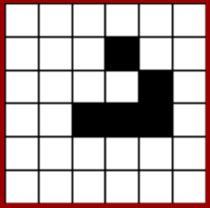
Moving



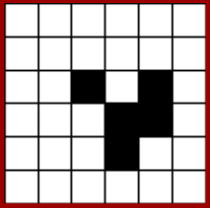
Time = 1



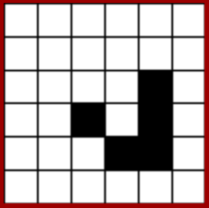
time = 2



time = 3

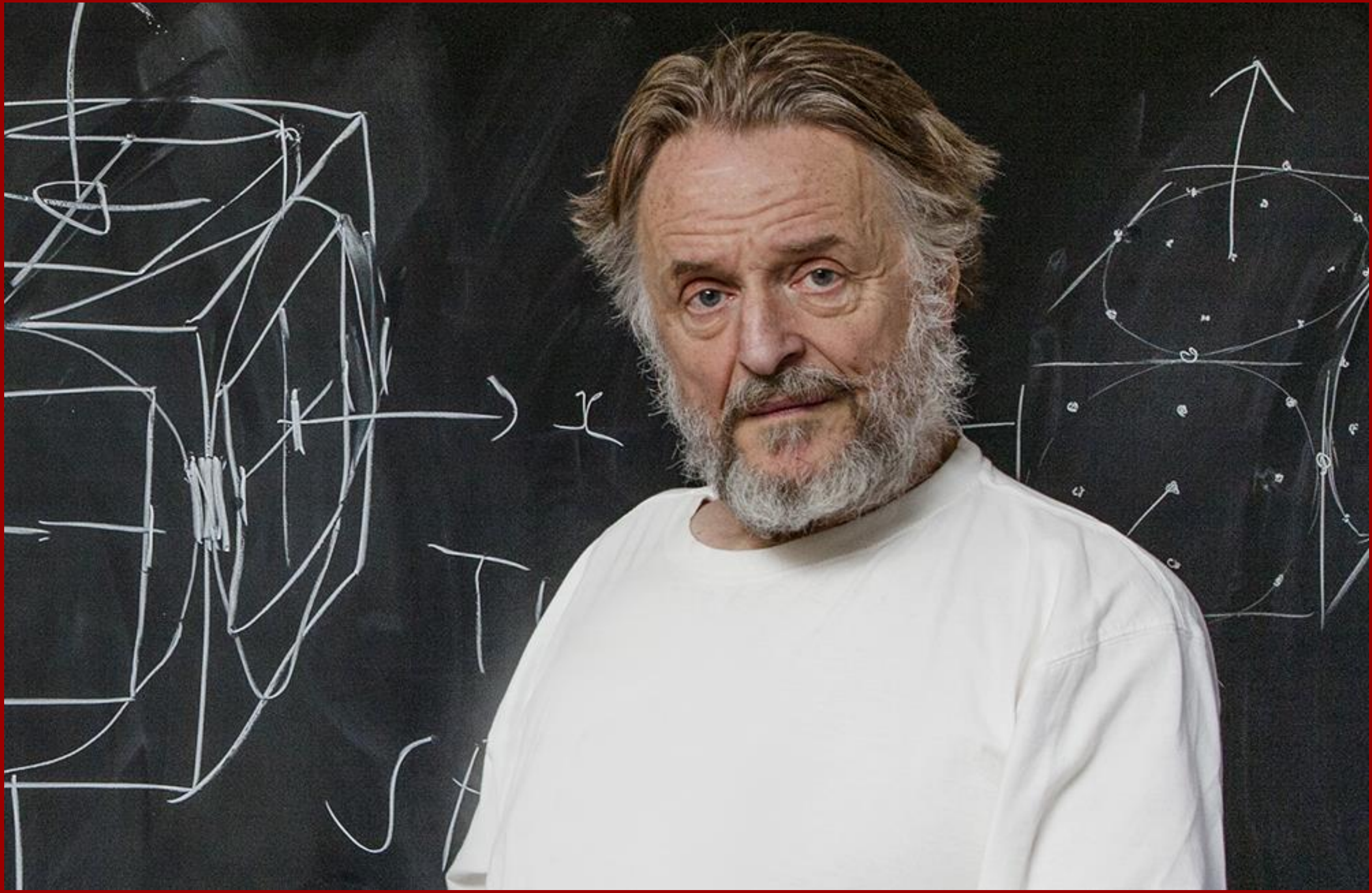


time = 4



time = 5

Example CA: Conway’s Game of Life



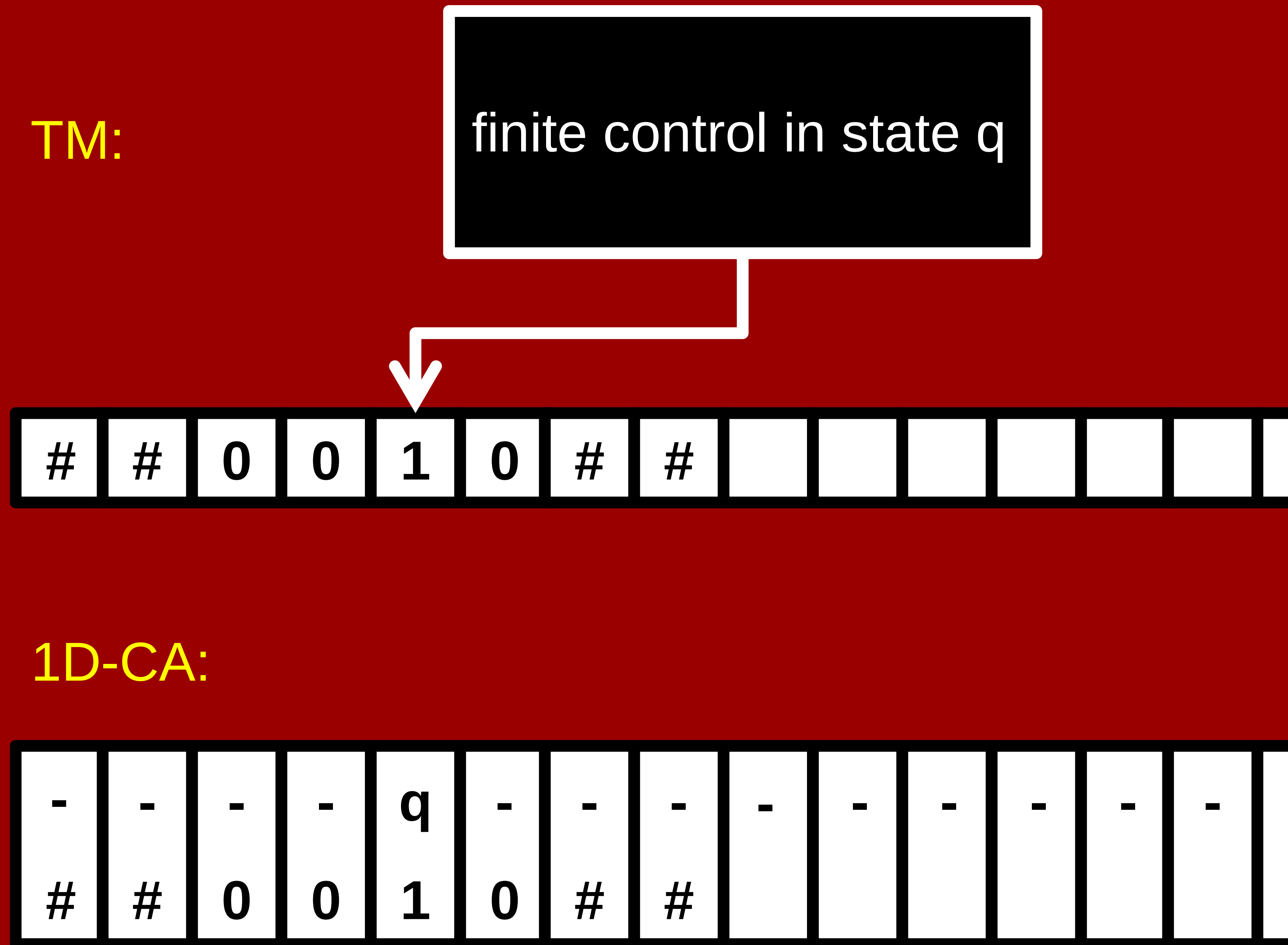
CA Turing Equivalence

Theorem: Python / a TM can simulate any CA.

Theorem: For any TM there is a 1D-CA simulating it.

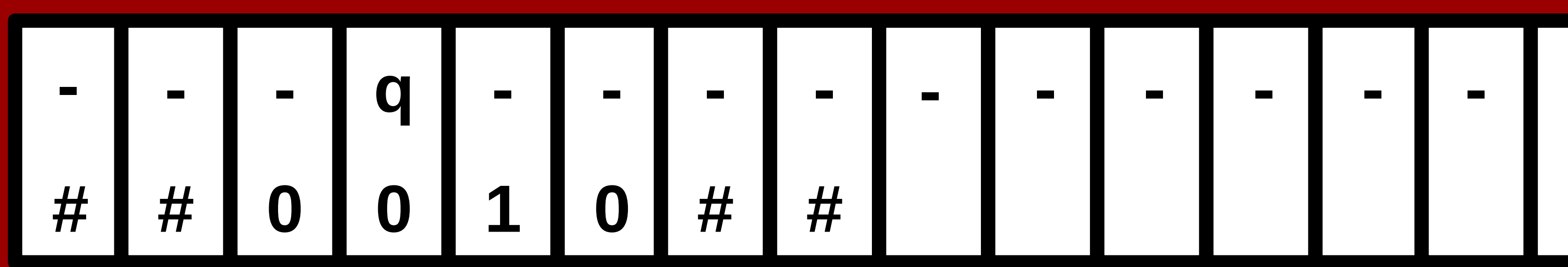
Construction Sketch:

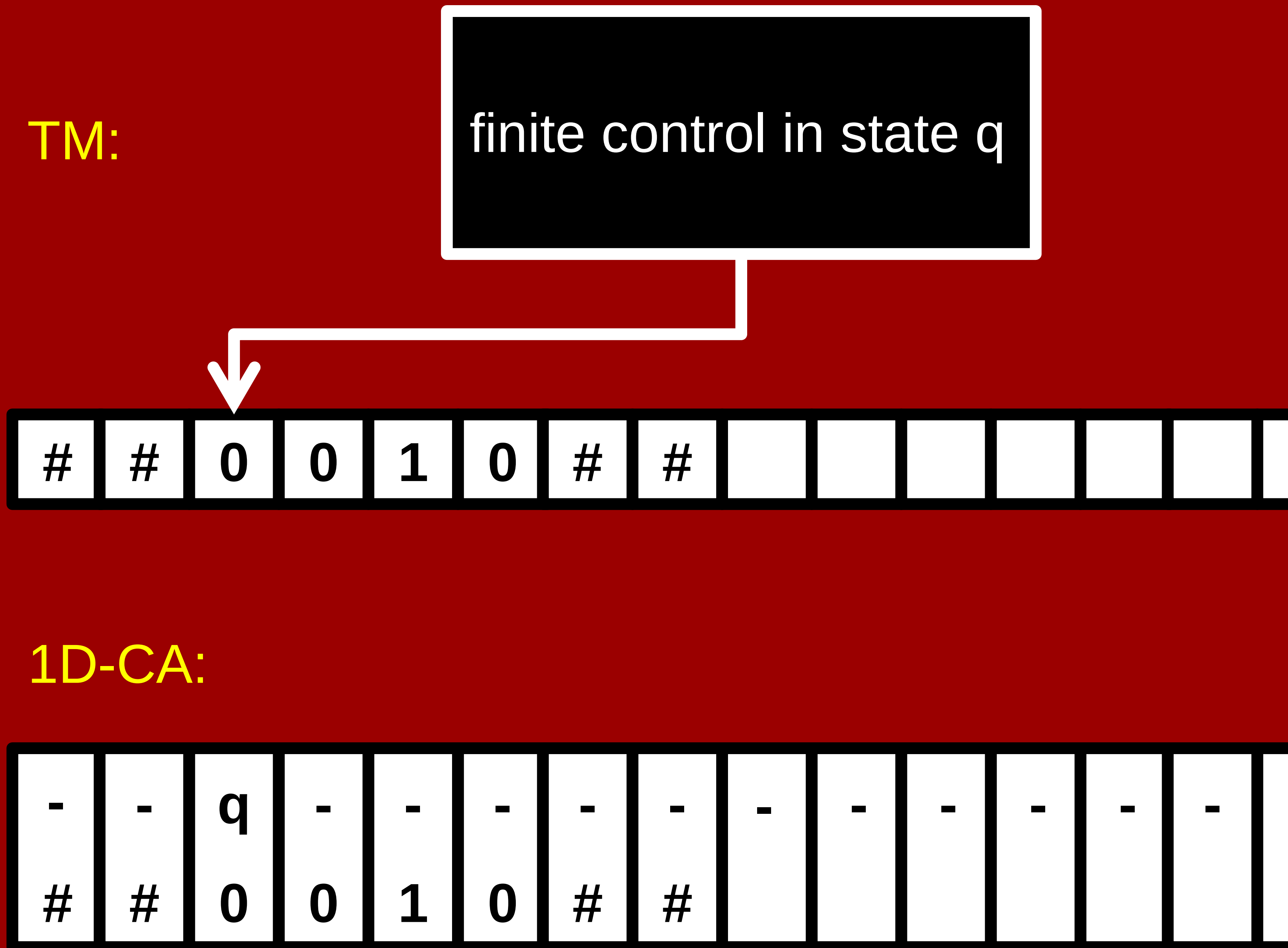
- For TM with state set Q and tape alphabet Γ create 1D-CA with state space $\Gamma \times (Q \cup \{-\})$.
- Cells simulate the tape and exactly one cell indicates the position of the a head and the TM state.
- Cells only transition if a neighboring cell contains the head.
- Transitions are based on the TM transition function.


TM:
finite control in state q
0 0 1 0 #
1D-CA:
- - - - q - - - - - - - - -
0 0 1 0 #

TM:

1D-CA:

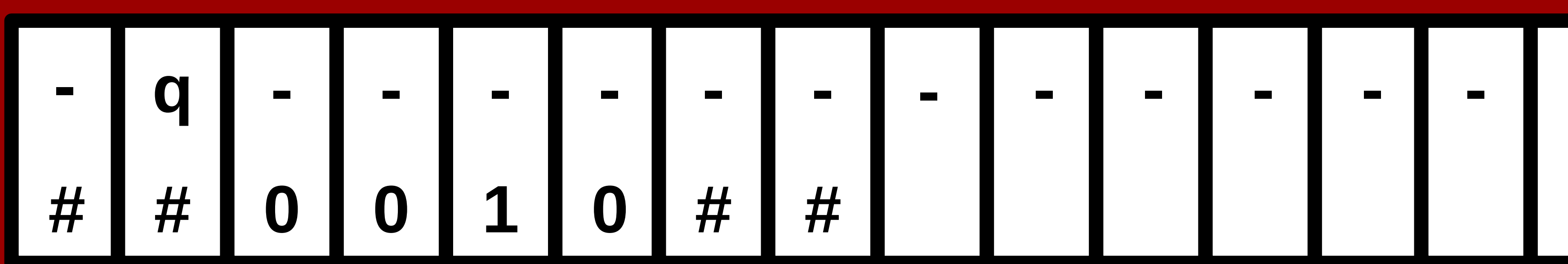


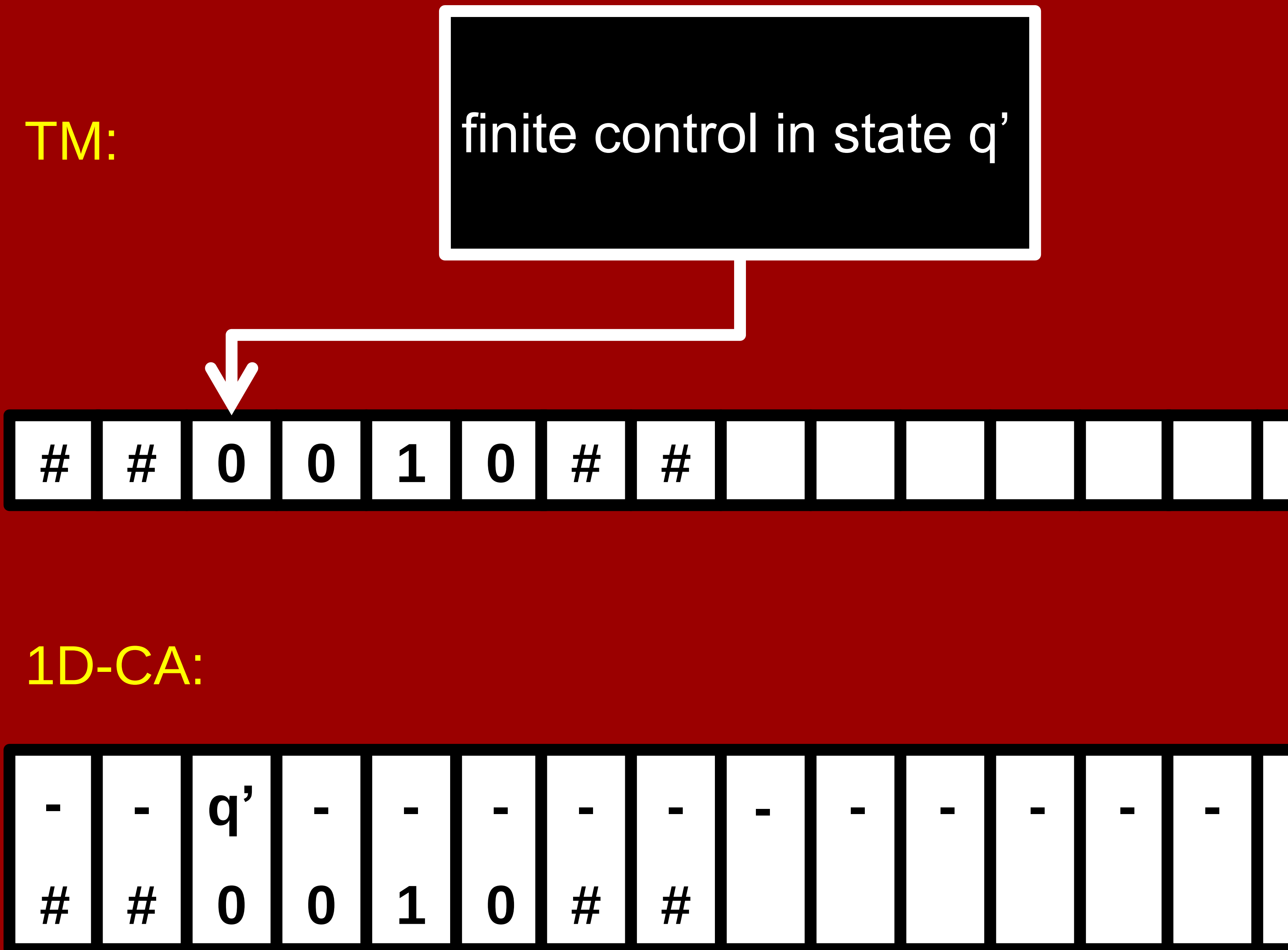


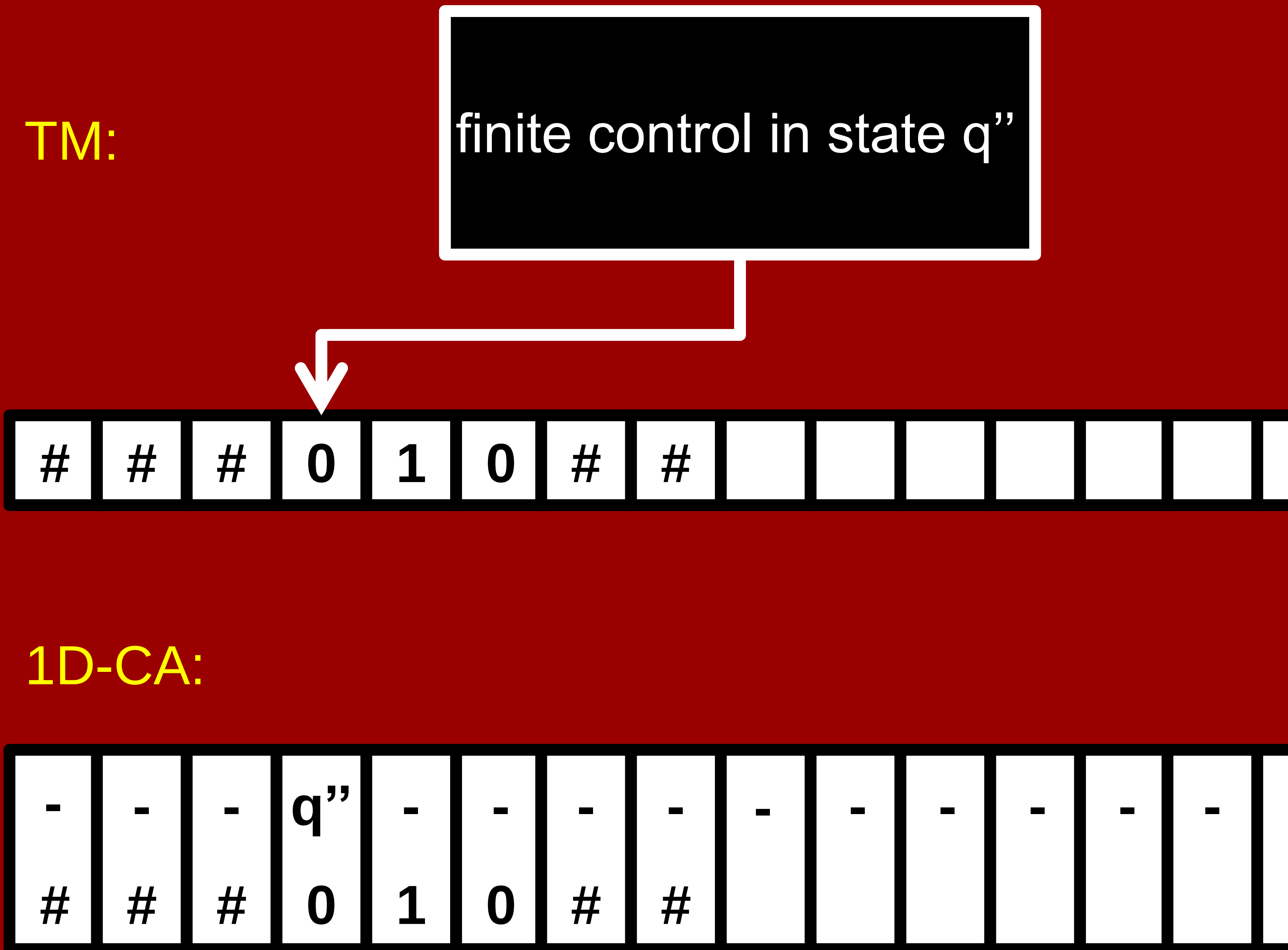
TM:
finite control in state q
0 0 1 0 #
1D-CA:
- - q - - - - - - - - - - -
0 0 1 0 #

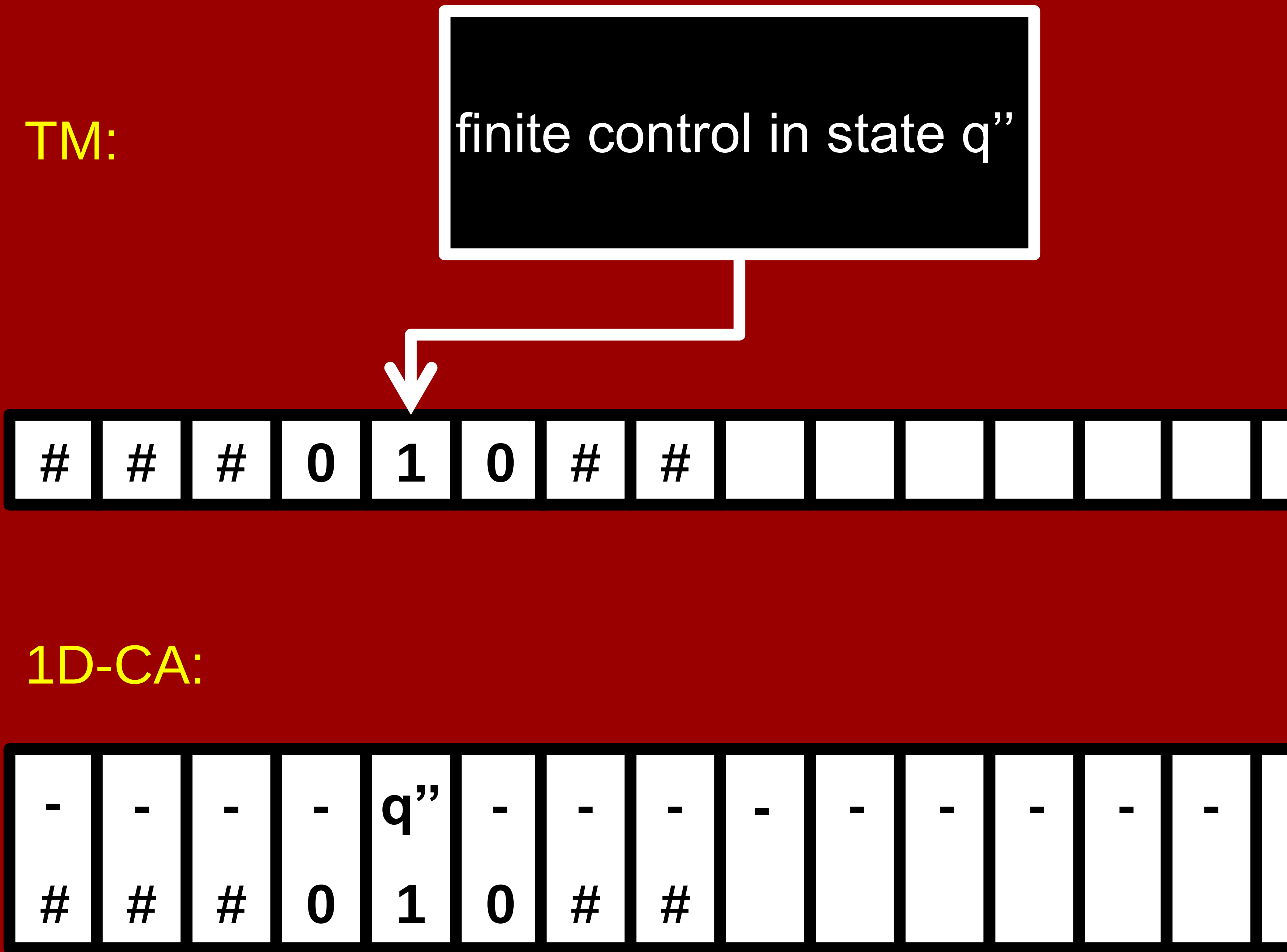
TM:

1D-CA:

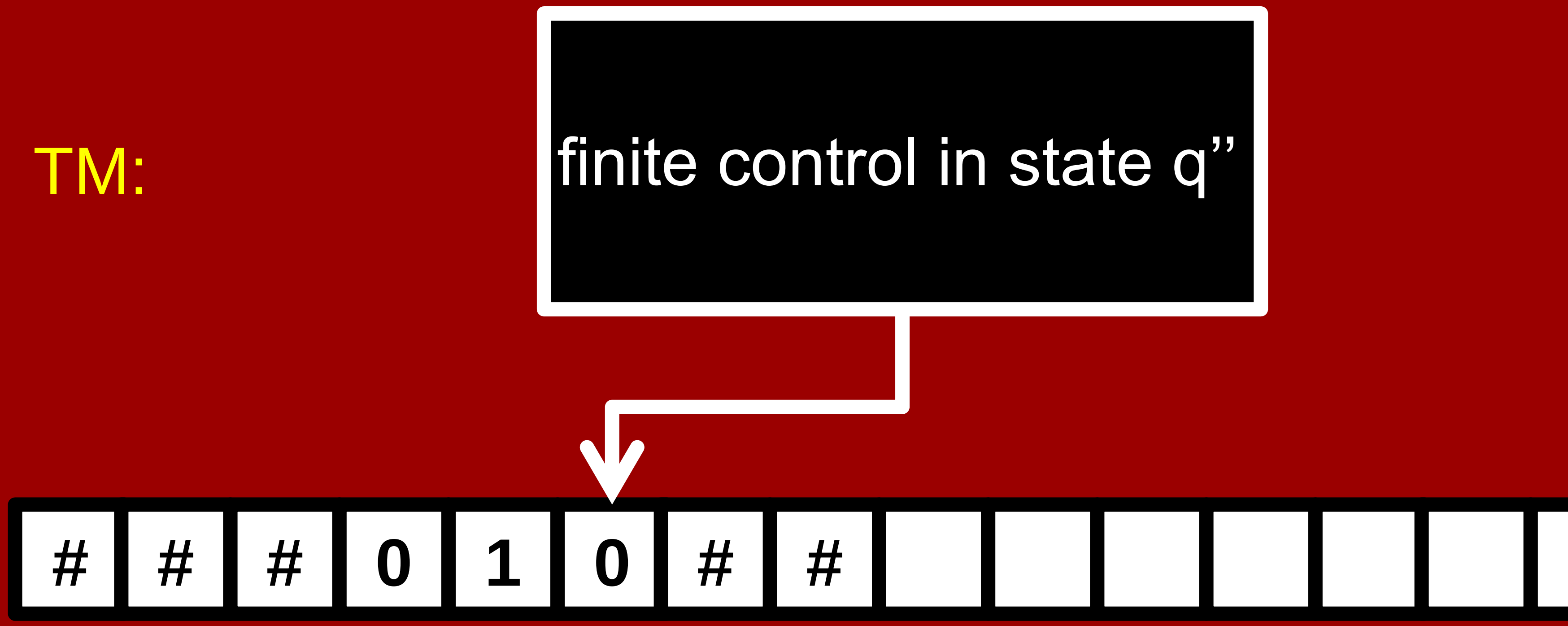



TM:
finite control in state q'
0 0 1 0 #
1D-CA:
- - q' - - - - - - - - - - -
0 0 1 0 #

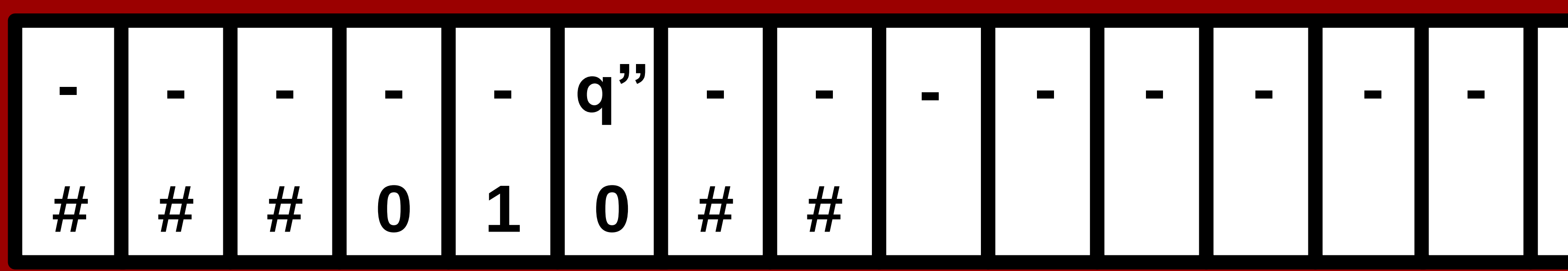

TM:
finite control in state q’’
#
#
#
0
1
0
#
#
1D-CA:
- #
- #
- #
q’’ 0
- 1
- 0
- #
- #
-
-
-
-
-
-


TM:
finite control in state q”
0 1 0 #
1D-CA:
- - - - q” - - - - - - - - -
0 1 0 #

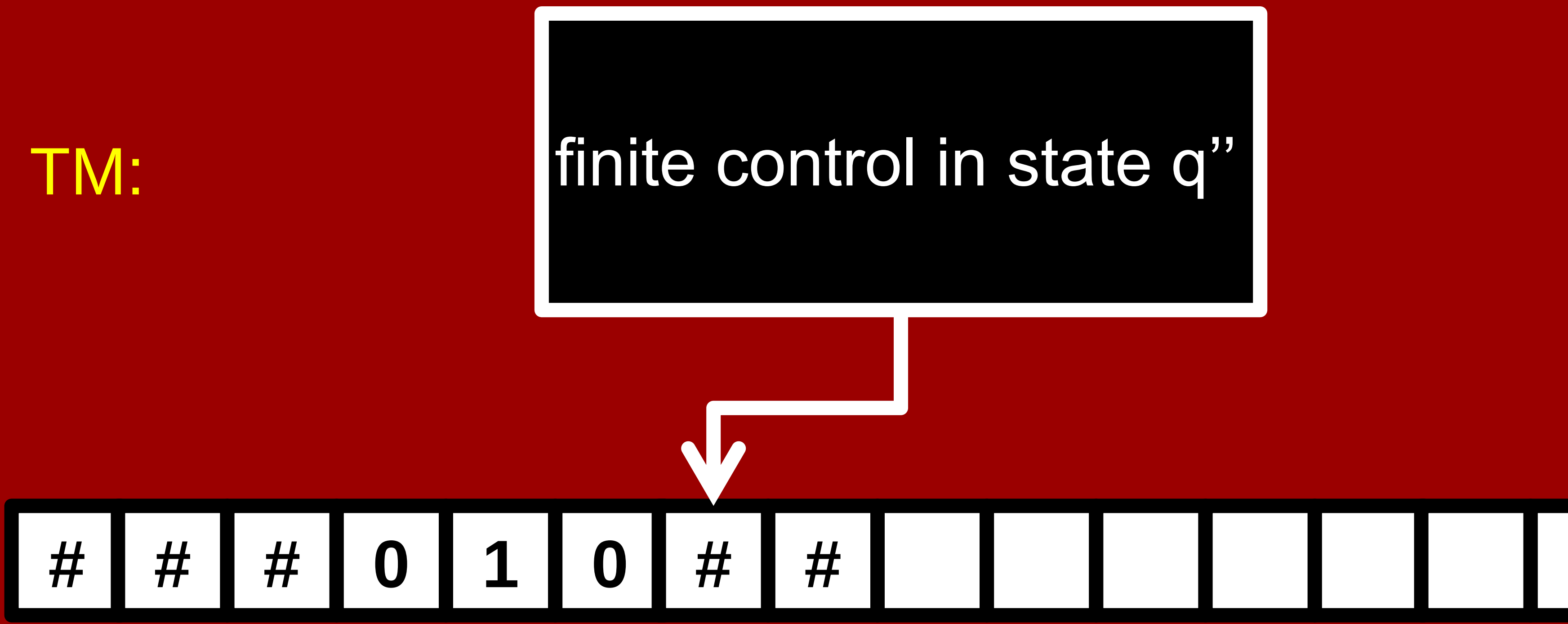
TM:



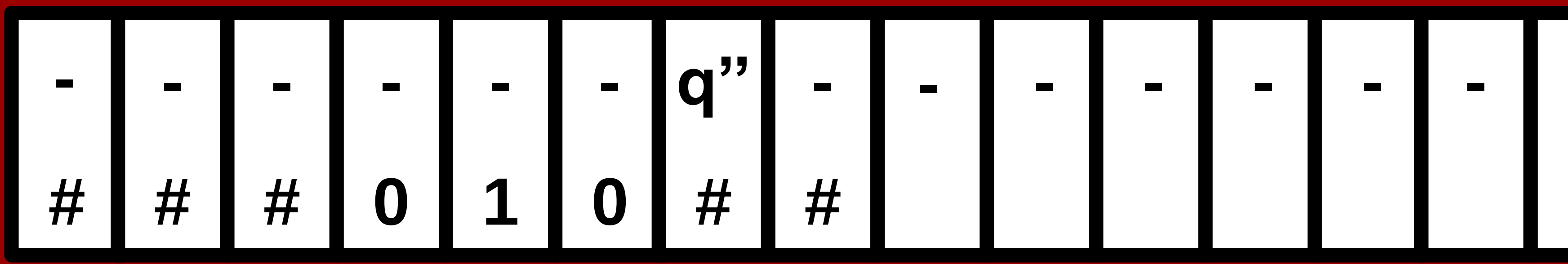
1D-CA:



TM:



1D-CA:



CA Turing Equivalence

Theorem: Python / a TM can simulate any CA.

Theorem: For any TM there is a 1D-CA simulating it.

Construction Sketch:

For TM with state set Q and tape alphabet Γ create 1D-CA with state space Γ x (Q ∪ {-}).. Cells simulate the tape and exactly one cell indicates the position of the a head and the TM state. Cells only transition if a neighboring cell contains the head. Transitions are based on the TM transition function.

Theorem: Game of Life can simulate a universal TM.

Church–Turing Thesis:

“Any natural / reasonable notion of computation can be simulated by a TM.”

Decidability

Decidable languages

Definition:

A language $L \subseteq \Sigma^*$ is **decidable** if there is a Turing Machine M which:

1. **Halts on every input** $x \in \Sigma^*$.
2. Accepts inputs $x \in L$ and rejects inputs $x \notin L$.

Such a Turing Machine is called a **decider**.

It 'decides' the language L.

We like deciders. We don't like TM's that sometimes loop.

Decidability: Poll

ACCEPT$_{\text{DFA}}$ = { $\langle D, x \rangle$ | D is a DFA that accepts x}

SELF-ACCEPT$_{\text{DFA}}$ = {$\langle D \rangle$ | D is a DFA that accepts $\langle D \rangle$}

EMPTY$_{\text{DFA}}$ = { $\langle D \rangle$ | D is a DFA that accepts no x}

EQUIV$_{\text{DFA}}$ =

= { $\langle D, D' \rangle$ | D and D' are DFA and L(D) = L(D')}

Encoding different objects with strings

Fix some alphabet Σ .

We use the $\langle \cdot \rangle$ notation to denote the encoding of an object as a string in Σ^* .

Examples:

$\langle M \rangle \in \Sigma^*$ is the encoding a TM M

$\langle D \rangle \in \Sigma^*$ is the encoding a DFA D

$\langle M_1, M_2 \rangle \in \Sigma^*$ is the encoding of a pair of TMs

$\langle M, x \rangle \in \Sigma^*$ is the encoding a pair (M, x) , where M is a TM, and $x \in \Sigma^*$.

Decidability: Poll

ACCEPT$_{\text{DFA}}$ = { $\langle D, x \rangle$ | D is a DFA that accepts x}

SELF-ACCEPT$_{\text{DFA}}$ = {$\langle D \rangle$ | D is a DFA that accepts $\langle D \rangle$}

EMPTY$_{\text{DFA}}$ = { $\langle D \rangle$ | D is a DFA that accepts no x}

EQUIV$_{\text{DFA}}$ =

= { $\langle D, D' \rangle$ | D and D' are DFA and L(D) = L(D')}

Decidability: Examples

ACCEPT$_{\text{DFA}}$ = { $\langle D, x \rangle$ | D is a DFA that accepts x}

SELF-ACCEPT$_{\text{DFA}}$ = {$\langle D \rangle$ | D is a DFA that accepts $\langle D \rangle$}

Theorem:

ACCEPT$_{\text{DFA}}$ is decideable.

SELF-ACCEPT$_{\text{DFA}}$ is decideable.

Proof: Simulate DFA step by step.

Decidability: Examples

$\text{EMPTY}_{\text{DFA}} = \{ \langle D \rangle \mid \text{D is a DFA that accepts no x} \}$

Theorem:

$\text{EMPTY}_{\text{DFA}}$ is decidable.

Proof:

A DFA D accepts the empty language iff
no accepting state is reachable from the start state
via a simple sequence of states.

Try all |Q|! possible such sequences.

Decidability: Examples

EQUIV$_{DFA}$ =

= { $\langle D, D' \rangle$ | D and D' are DFA and L(D) = L(D')}

Theorem:

EQUIV$_{DFA}$ is decidable.

Proof:

Create a DFA D'' for the symmetric difference

$$L(D'') = (L(D) \cap \overline{L(D')}) \cup (\overline{L(D)} \cap L(D'))$$

using the Union and Intersection theorem for DFA.

Run the decider TM for EMPTY$_{DFA}$ on $\langle D'' \rangle$.

Reductions

*Using one problem as a **subroutine** to solve another is a powerful algorithmic technique.*

Definition:

Language A *reduces to* language B means:

"It is possible to decide A using an algorithm for deciding B as a subroutine."

Notation: $A \leq_T B$ (T stands for Turing).

Think, "A is no harder than B".

Reductions

Fact:

Suppose $A \leq_T B$; i.e., A reduces to B.

If B is decidable, then A is also decidable.

Here:

$EQUIV_{DFA} \leq_T EMPTY_{DFA}$ and $EMPTY_{DFA}$ is decidable.

This makes $EQUIV_{DFA}$ decidable.

Indeed, $EQUIV_{DFA}$ is at most as hard as $EMPTY_{DFA}$ because solving $EQUIV_{DFA}$ is easy given a solution to $EMPTY_{DFA}$.

Undecidability

Undecidability

Definition:

A language $L \subseteq \Sigma^*$ is **undecidable** if there is no Turing Machine M which:

1. **Halts on every input** $x \in \Sigma^*$.
2. Accepts inputs $x \in L$ and rejects inputs $x \notin L$.

Poll

Let A be the set of all languages over $\Sigma = \{0,1\}$.

Select all correct ones:

- ***A* is finite**
- ***A* is infinite**
- ***A* is countable**
- ***A* is uncountable**

Poll

Let A be the set of all languages over $\Sigma = \{0,1\}$.

Select all correct ones:

- ***A* is finite**
- ✓ ***A* is infinite**
- ***A* is countable**
- ✓ ***A* is uncountable** $|A| = |\mathcal{P}(\Sigma^*)| = |\mathcal{P}(\mathbb{N})|$

Question:

Is every language in $\{0,1\}^*$ decidable?

$\Leftrightarrow$ Is every function $f : \{0,1\}^* \to \{0,1\}$ computable?

Answer: No!

Every TM is encodable by a finite string.

Therefore the set of all TM's is countable.

So the subset of all *decider* TM's is countable.

Thus the set of all decidable languages is countable.

But the set of **all** languages is the power set of $\{0,1\}^*$ which is uncountable.

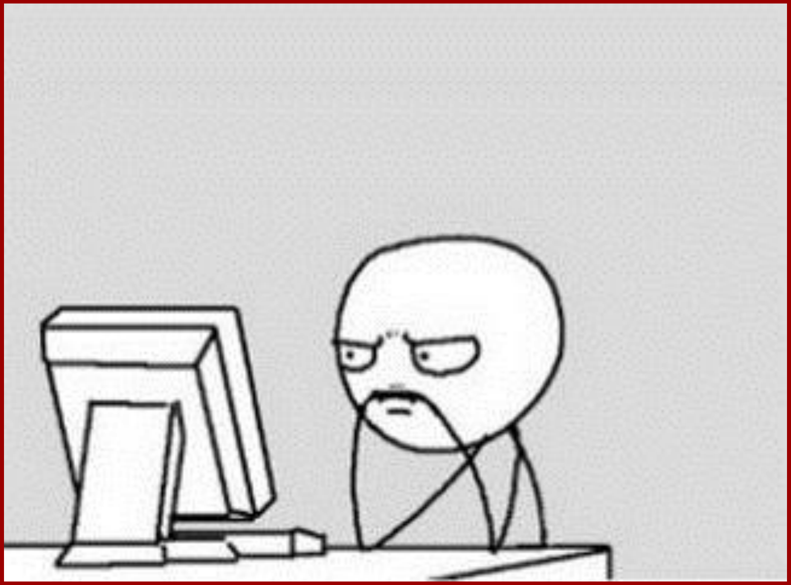
Question:

Is every language in $\{0,1\}^*$ decidable?

$\Leftrightarrow$ Is every function $f : \{0,1\}^* \rightarrow \{0,1\}$ computable?

Answer:

Essentially all (decision) functions are uncomputable!



Study Guide

Definitions:

Cellular Automata (CA)

Reductions

Undecidability

Theorems/proofs:

Turing equivalency of CA

Decidability of several languages

Existence of undecidable problems

Practice:

Decidability Proofs (via Reductions)