Emptiness Problems for Distributed Automata

Fabian Reiter

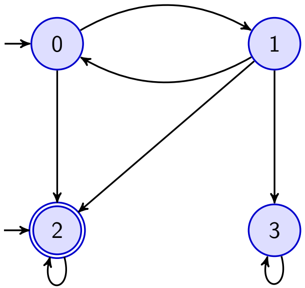
IRIF, Université Paris Diderot

September 21, 2017

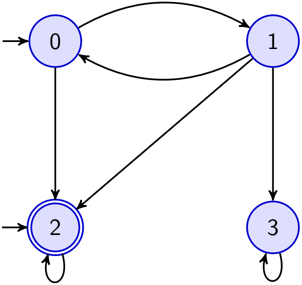
Joint work with Antti Kuusisto

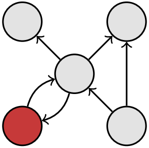
Distributed automata

Distributed automata

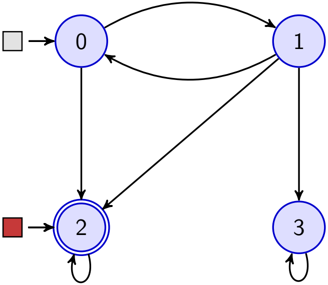


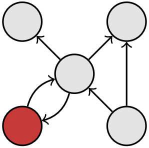
Distributed automata



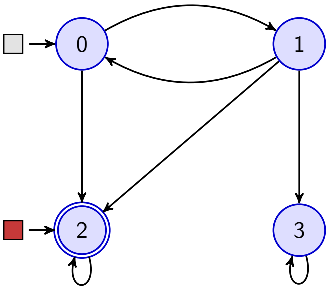


Distributed automata

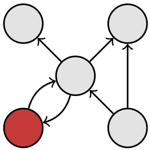




Distributed automata

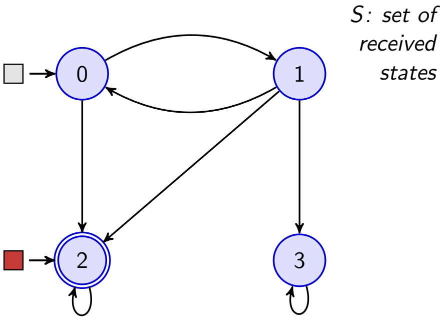


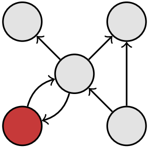
Transition function:
$\delta\colon Q \times 2^Q \to Q$
(Q: set of states)



Distributed automata

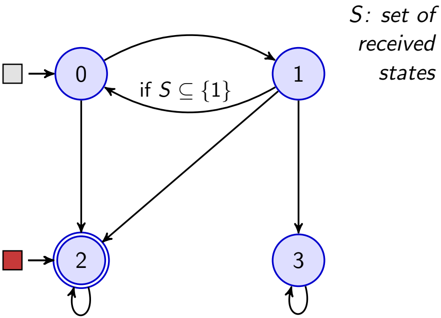
Transition function:
$\delta\colon Q \times 2^Q \to Q$
(Q: set of states)

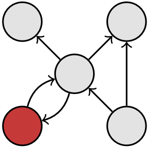




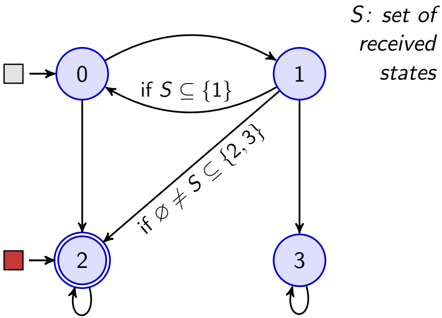
Distributed automata

Transition function:
$\delta\colon Q \times 2^Q \to Q$
(Q: set of states)

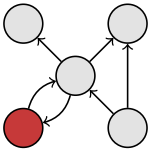




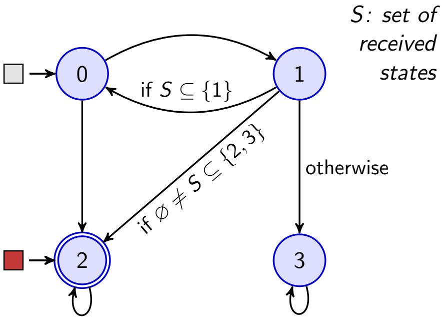
Distributed automata



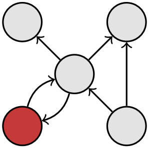
Transition function:
$\delta\colon Q \times 2^Q \to Q$
(Q: set of states)



Distributed automata

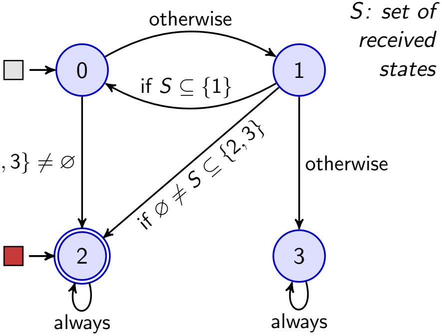


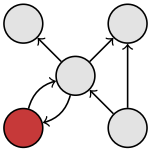
Transition function:
$\delta\colon Q \times 2^Q \to Q$
(Q: set of states)



Distributed automata

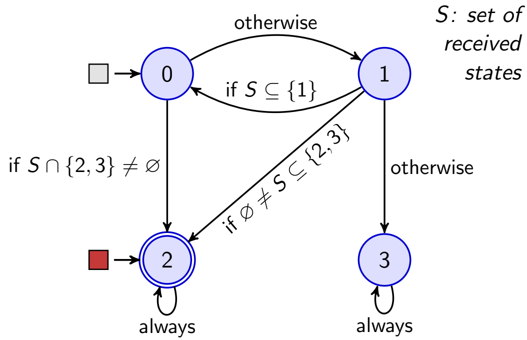
Transition function:
$\delta \colon Q \times 2^Q \to Q$
(Q: set of states)



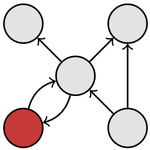


Distributed automata

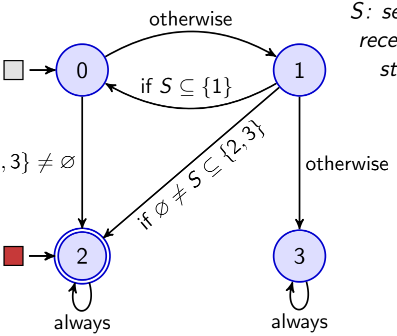
Transition function:
$\delta\colon Q \times 2^Q \to Q$
(Q: set of states)



Synchronous run:



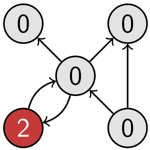
Distributed automata



S: set of received states

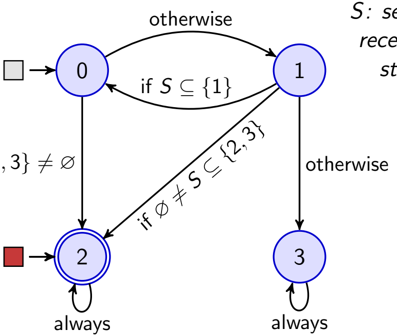
Transition function:
$\delta \colon Q \times 2^Q \to Q$
(Q: set of states)

Synchronous run:



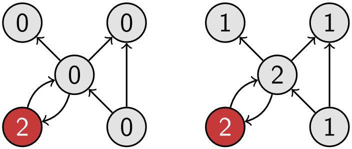
Distributed automata

S: set of received states



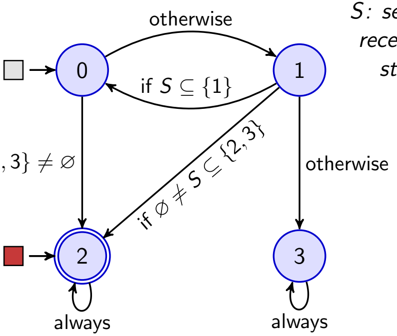
Transition function:
$\delta \colon Q \times 2^Q \to Q$
(Q: set of states)

Synchronous run:



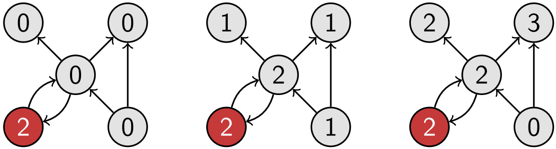
Distributed automata

Transition function:
$\delta\colon Q \times 2^Q \to Q$
(Q: set of states)



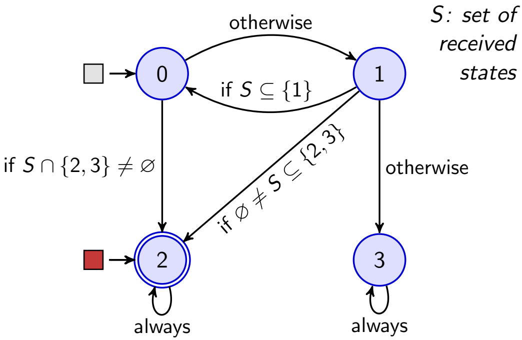
S: set of received states

Synchronous run:

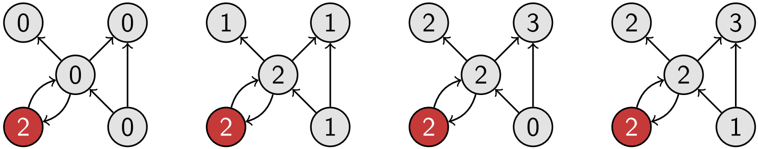


Distributed automata

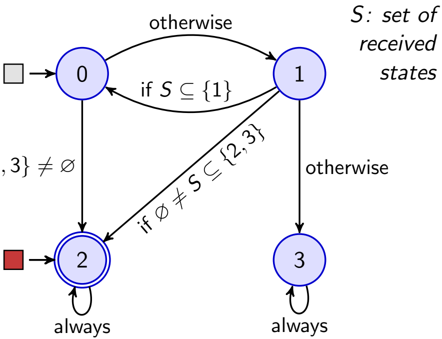
Transition function:
$\delta \colon Q \times 2^Q \to Q$
(Q: set of states)



Synchronous run:

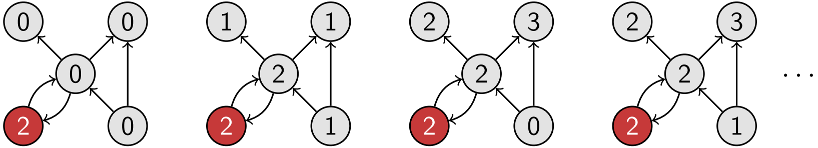


Distributed automata



Transition function:
$\delta\colon Q \times 2^Q \to Q$
(Q: set of states)

Synchronous run:



Emptiness problem

Emptiness problem

Automaton A *accepts* digraph G on node $v \in G$
iff v visits an accepting state at some time $t \in \mathbb{N}$.

Emptiness problem

Automaton A *accepts* digraph G on node $v \in G$
iff v visits an accepting state at some time $t \in \mathbb{N}$.

Emptiness problem:

Does automaton A accept on some node in some digraph?

Emptiness problem

Automaton A *accepts* digraph G on node $v \in G$
iff v visits an accepting state at some time $t \in \mathbb{N}$.

Emptiness problem:

Does automaton A accept on some node in some digraph?

Simple reduction:

Emptiness problem

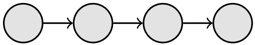
Automaton A *accepts* digraph G on node $v \in G$
iff v visits an accepting state at some time $t \in \mathbb{N}$.

Emptiness problem:

Does automaton A accept on some node in some digraph?

Simple reduction:

undecidable on dipaths



Emptiness problem

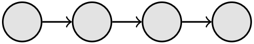
Automaton A *accepts* digraph G on node $v \in G$
iff v visits an accepting state at some time $t \in \mathbb{N}$.

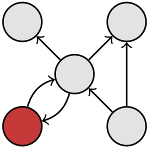
Emptiness problem:

Does automaton A accept on some node in some digraph?

Simple reduction:

undecidable on dipaths $\implies$ undecidable on digraphs





Simulating a Turing machine

Simulating a Turing machine

Would be easy on doubly linked dipaths:

Simulating a Turing machine

Would be easy on doubly linked dipaths:

Turing machine

with alphabet $\{\square, \blacksquare\}$
and state set $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Simulating a Turing machine

Would be easy on doubly linked dipaths:

space $\longrightarrow$

Turing machine

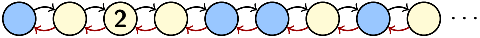
with alphabet $\{\square, \blacksquare\}$
and state set $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Simulating a Turing machine

Would be easy on doubly linked dipaths:

space ⟶

space ⟶



Turing machine

with alphabet {□, □}

and state set $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Simulating a Turing machine

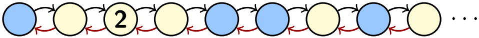
Would be easy on doubly linked dipaths:

space $\longrightarrow$

Turing machine

with alphabet $\{\square, \blacksquare\}$
and state set $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

space $\longrightarrow$

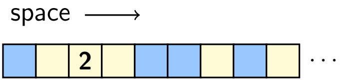


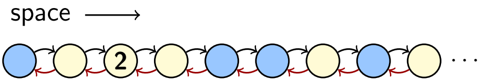
Distributed automaton

with state set
$\{\square, \blacksquare\} \times \{\epsilon, \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Simulating a Turing machine

Would be easy on doubly linked dipaths:





Turing machine

with alphabet $\{\square, \blacksquare\}$
and state set $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Distributed automaton

with state set
$\{\square, \blacksquare\} \times \{\epsilon, \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

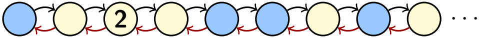
But not on simple dipaths:

Simulating a Turing machine

Would be easy on doubly linked dipaths:

space $\longrightarrow$

space $\longrightarrow$



Turing machine

with alphabet $\{\square, \blacksquare\}$
and state set $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Distributed automaton

with state set
$\{\square, \blacksquare\} \times \{\epsilon, \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

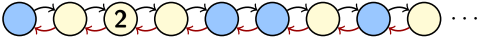
But not on simple dipaths:

Simulating a Turing machine

Would be easy on doubly linked dipaths:

space ⟶

space ⟶



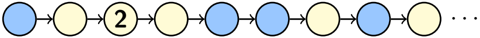
Turing machine

with alphabet $\{\square, \blacksquare\}$
and state set $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Distributed automaton

with state set
$\{\square, \blacksquare\} \times \{\epsilon, \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

But not on simple dipaths:

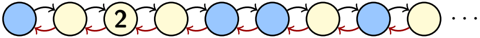


Simulating a Turing machine

Would be easy on doubly linked dipaths:

space ⟶

space ⟶



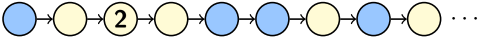
Turing machine

with alphabet $\{\square, \square\}$
and state set $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Distributed automaton

with state set
$\{\square, \square\} \times \{\epsilon, \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

But not on simple dipaths:



One-way communication ☹

Exchanging space and time

Exchanging space and time

Turing machine

alphabet : $\{\square, \blacksquare\}$

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Exchanging space and time

Turing machine

space $\longrightarrow$

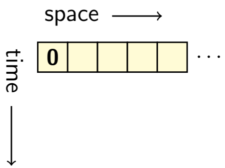
time $\longrightarrow$

alphabet : $\{\square, \square\}$

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Exchanging space and time

Turing machine

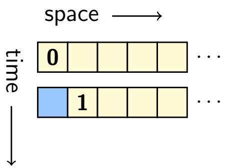


alphabet : {□, □}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Exchanging space and time

Turing machine

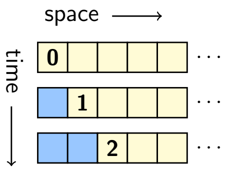


alphabet : {□, □}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Exchanging space and time

Turing machine

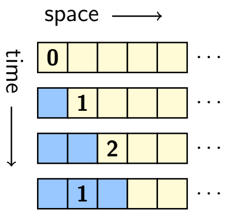


alphabet : $\{\square, \blacksquare\}$

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Exchanging space and time

Turing machine

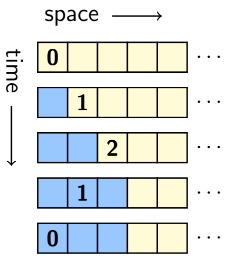


alphabet : {□, ■}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Exchanging space and time

Turing machine

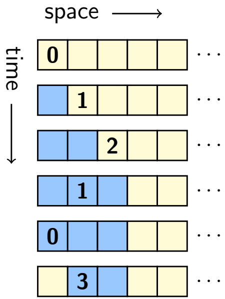


alphabet : {□, □}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Exchanging space and time

Turing machine

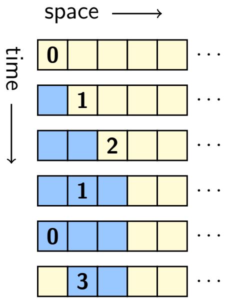


alphabet : $\{\square, \blacksquare\}$

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Exchanging space and time

Turing machine



alphabet : {□, □}

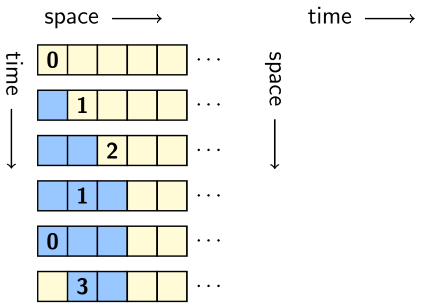
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Distributed automaton

Exchanging space and time

Turing machine

Distributed automaton



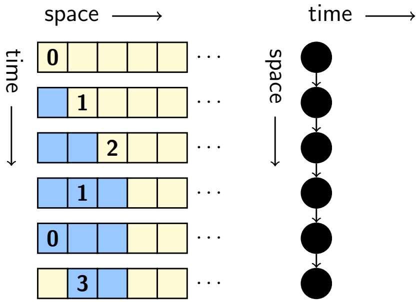
alphabet : $\{\square, \blacksquare\}$

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Exchanging space and time

Turing machine

Distributed automaton



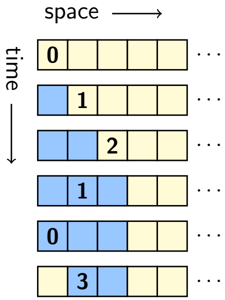
alphabet : $\{\square, \square\}$

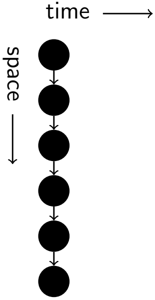
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

Exchanging space and time

Turing machine

Distributed automaton





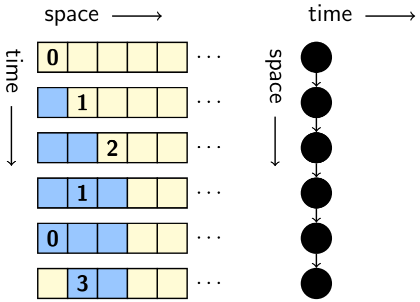
alphabet : {□, □}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

Exchanging space and time

Turing machine | **Distributed automaton**



alphabet : {□, □}

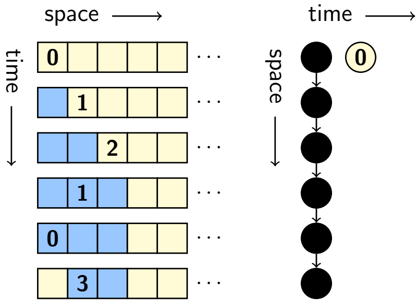
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

: nodes “visiting” a Turing cell

Exchanging space and time

Turing machine **Distributed automaton**



alphabet : {□, ■}

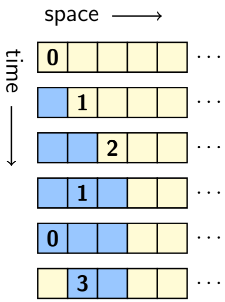
state set : {**0**, **1**, **2**, **3**}

● : waiting node

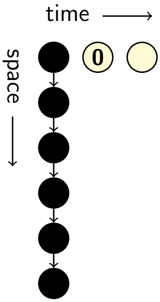
● (1) ○ : nodes "visiting" a Turing cell

Exchanging space and time

Turing machine



Distributed automaton



alphabet : {□, ■}

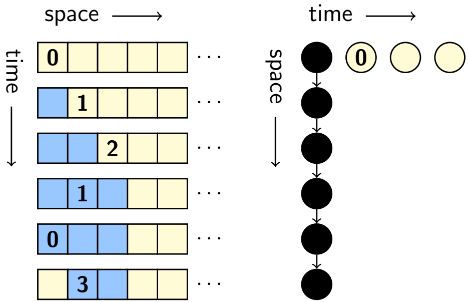
state set : {**0**, **1**, **2**, **3**}

● : waiting node

○ (1) ○ : nodes "visiting" a Turing cell

Exchanging space and time

Turing machine **Distributed automaton**



alphabet : {□, ■}

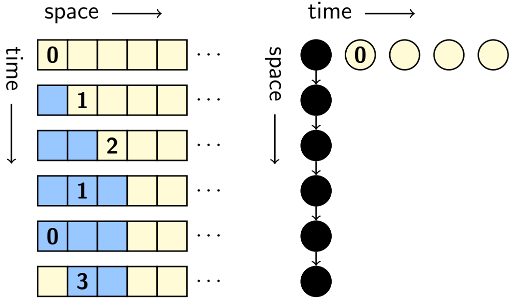
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

● (1) ○ : nodes "visiting" a Turing cell

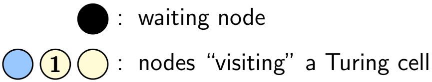
Exchanging space and time

Turing machine | **Distributed automaton**



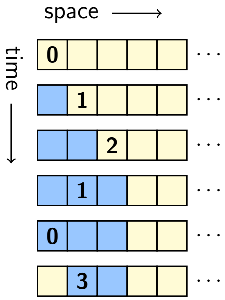
alphabet : {□, □}

state set : {**0**, **1**, **2**, **3**}

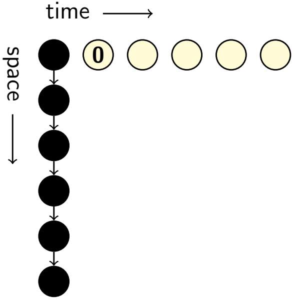


Exchanging space and time

Turing machine

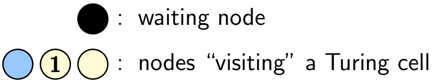


Distributed automaton



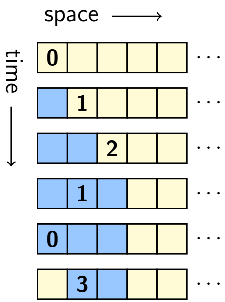
alphabet : {□, □}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

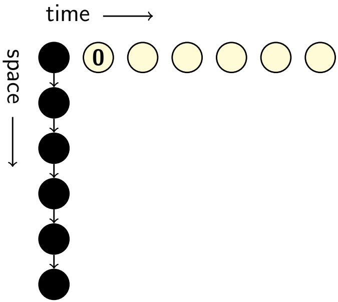


Exchanging space and time

Turing machine



Distributed automaton



alphabet : {□, □}

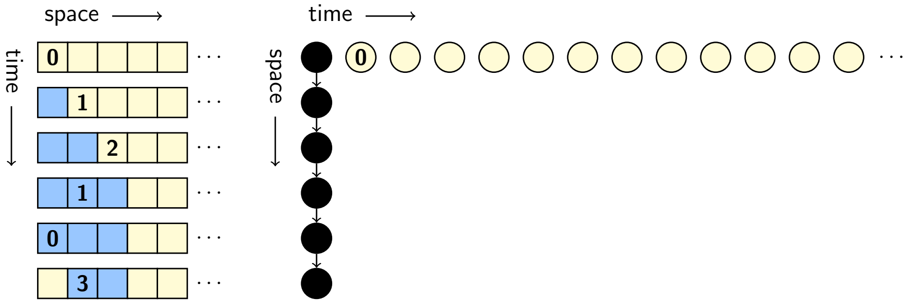
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

○ (1) ○ : nodes "visiting" a Turing cell

Exchanging space and time

Turing machine **Distributed automaton**



alphabet : {□, □}

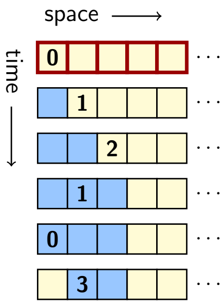
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

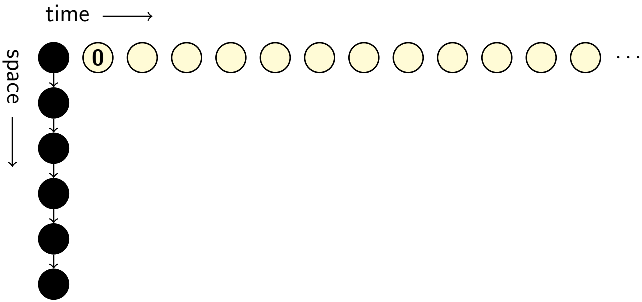
○ ① ○ : nodes “visiting” a Turing cell

Exchanging space and time

Turing machine



Distributed automaton



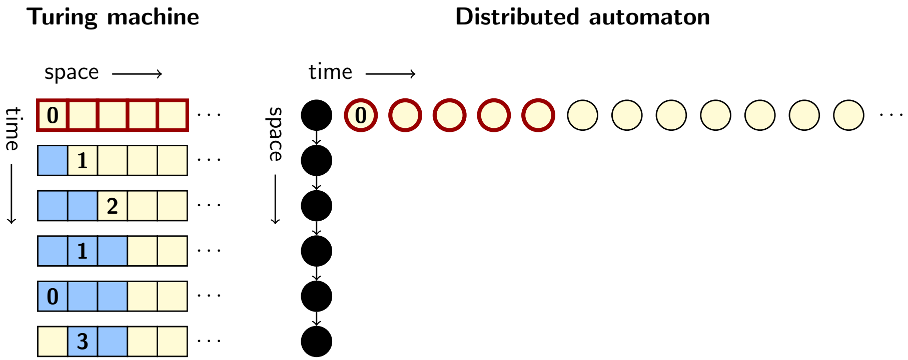
alphabet : {□, ■}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

○ (1) ○ : nodes “visiting” a Turing cell

Exchanging space and time



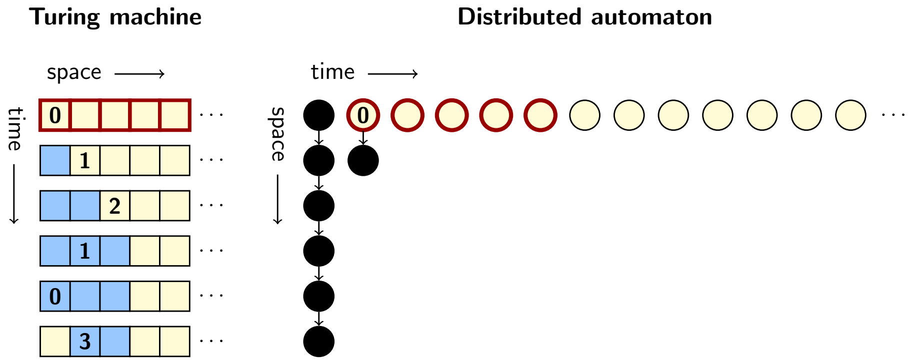
alphabet : {□, □}

state set : {**0**, **1**, **2**, **3**}

● : waiting node

○ (1) ○ : nodes "visiting" a Turing cell

Exchanging space and time



alphabet : {□, □}

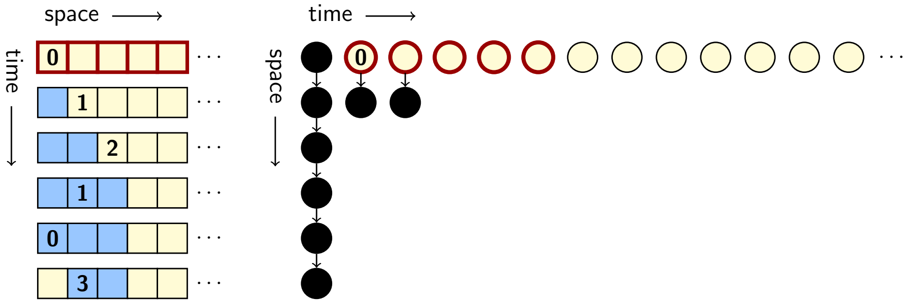
state set : {**0**, **1**, **2**, **3**}

● : waiting node

○ (1) ○ : nodes “visiting” a Turing cell

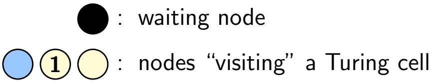
Exchanging space and time

Turing machine | **Distributed automaton**



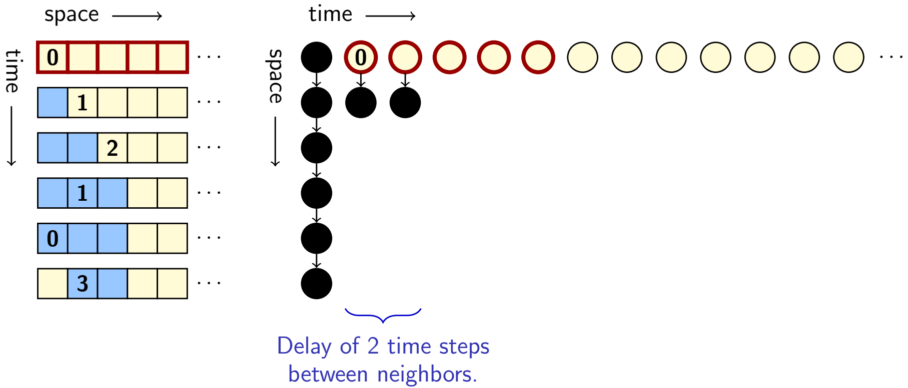
alphabet : {□, ■}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$



Exchanging space and time

Turing machine **Distributed automaton**



alphabet : {□, □}

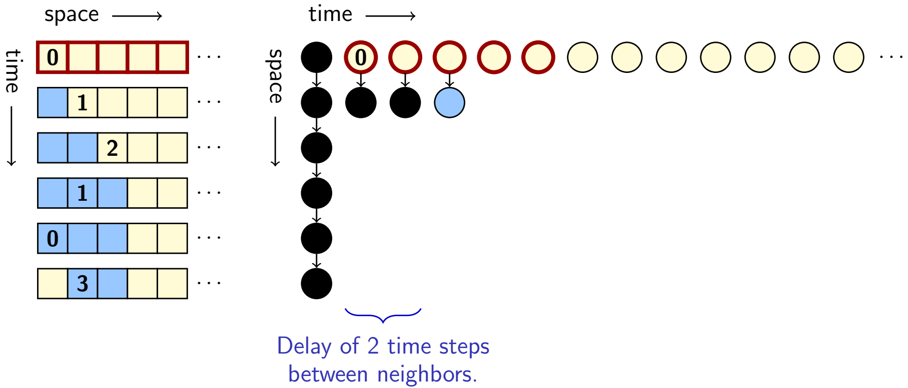
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

1 : nodes “visiting” a Turing cell

Exchanging space and time

Turing machine **Distributed automaton**



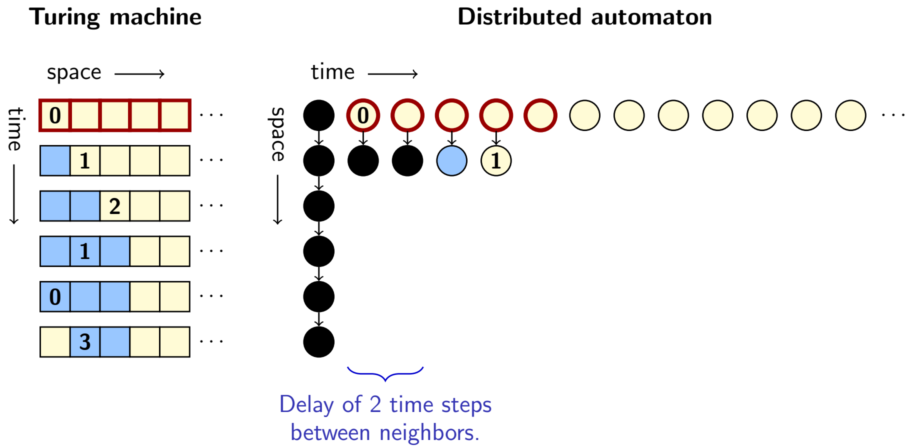
alphabet : {□, ■}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

○ ① ○ : nodes "visiting" a Turing cell

Exchanging space and time



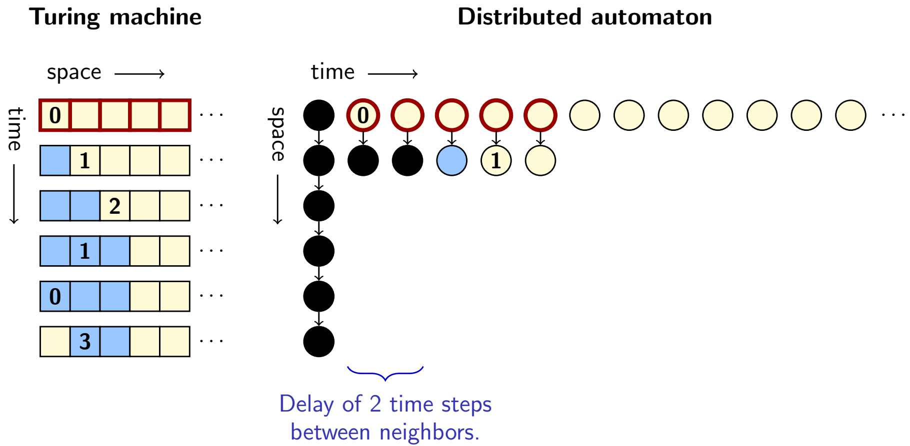
alphabet : {□, □}

state set : {**0**, **1**, **2**, **3**}

: waiting node

: nodes "visiting" a Turing cell

Exchanging space and time



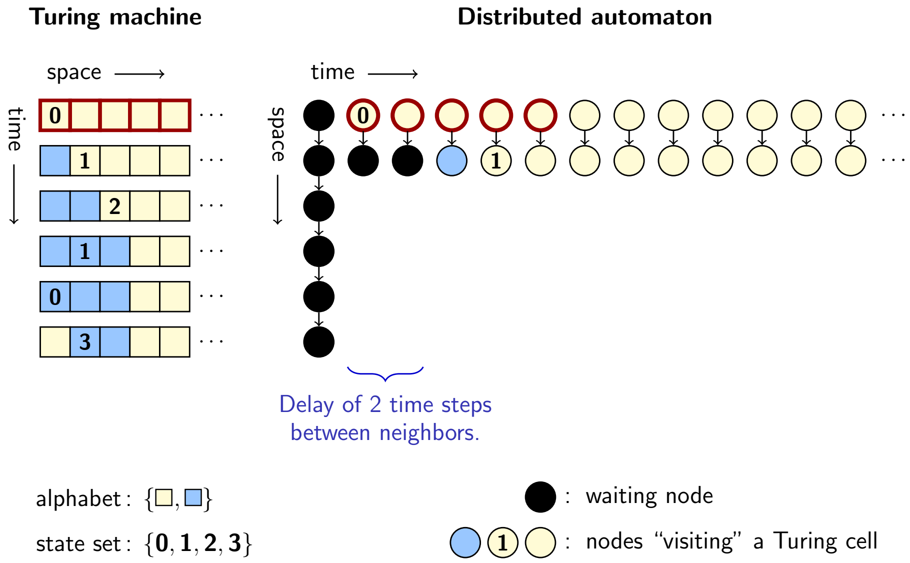
alphabet : {□, □}

state set : {**0**, **1**, **2**, **3**}

● : waiting node

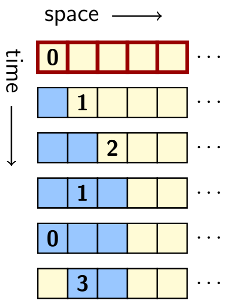
○ ① ○ : nodes “visiting” a Turing cell

Exchanging space and time

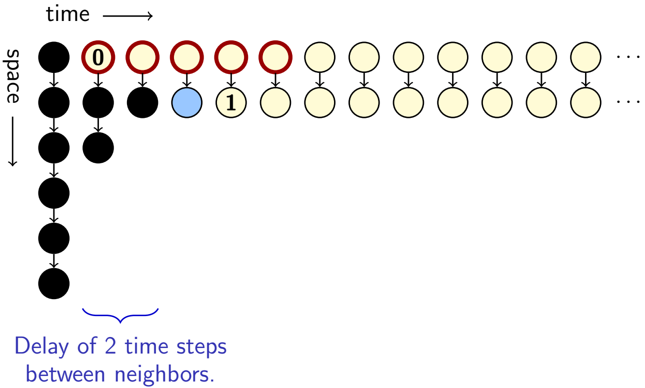


Exchanging space and time

Turing machine



Distributed automaton



alphabet : {□, □}

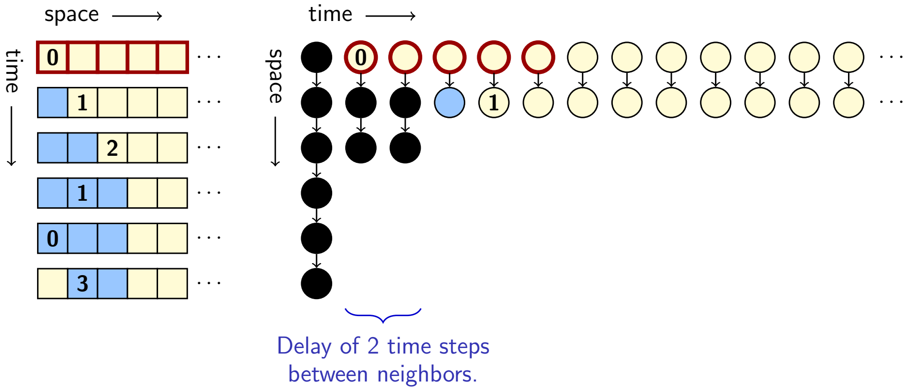
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

○ ① ○ : nodes "visiting" a Turing cell

Exchanging space and time

Turing machine | **Distributed automaton**



alphabet : {□, □}

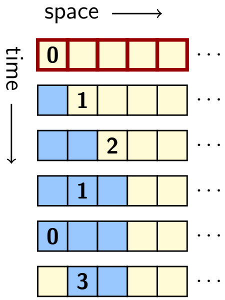
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

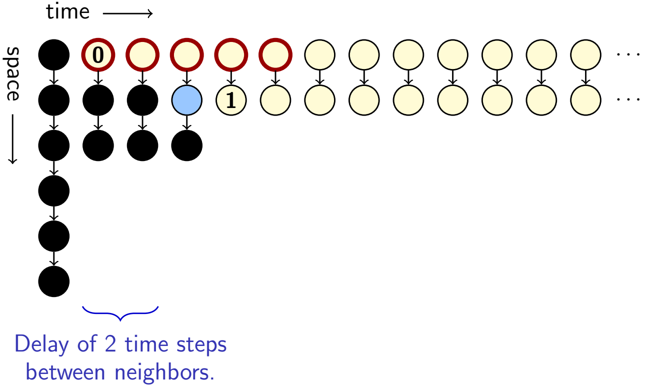
○ ① ○ : nodes "visiting" a Turing cell

Exchanging space and time

Turing machine



Distributed automaton



alphabet : {□, □}

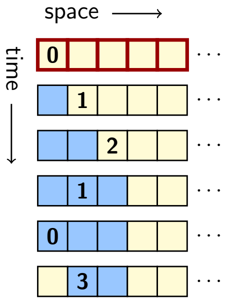
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

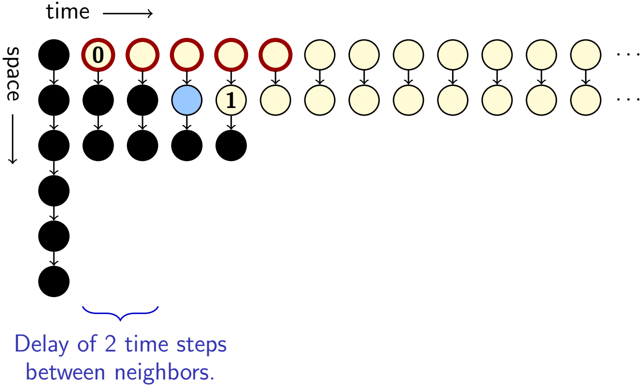
○ (1) ○ : nodes "visiting" a Turing cell

Exchanging space and time

Turing machine



Distributed automaton



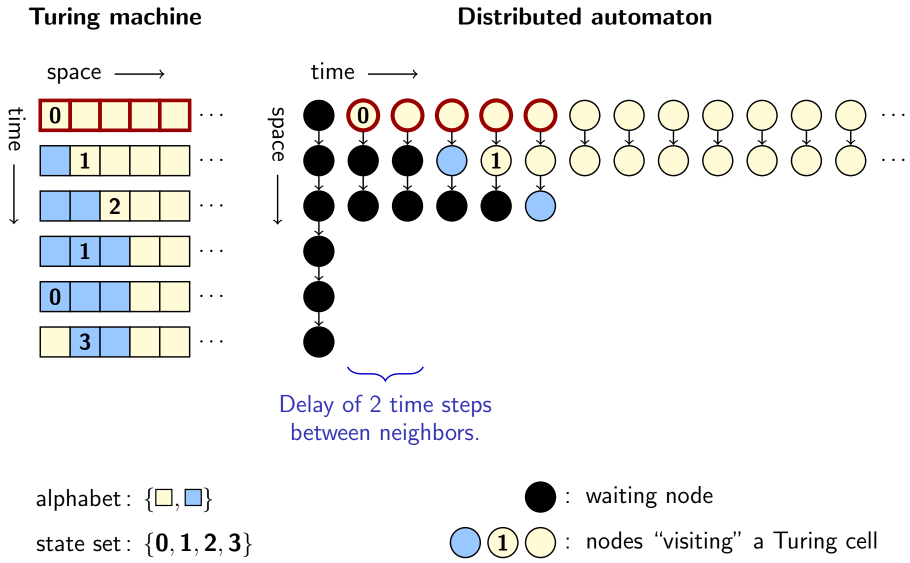
alphabet : {□, □}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

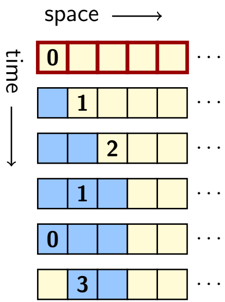
○ (1) ○ : nodes "visiting" a Turing cell

Exchanging space and time

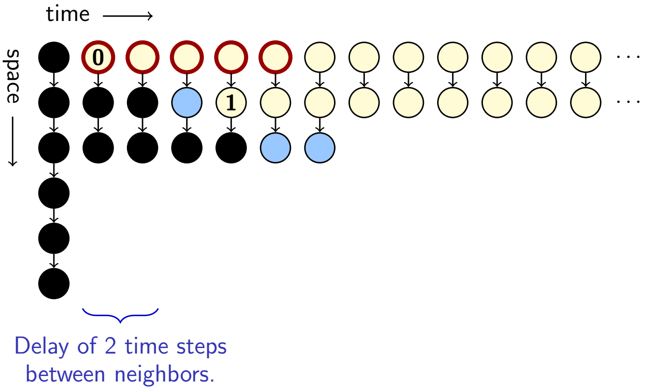


Exchanging space and time

Turing machine



Distributed automaton



alphabet : {□, ■}

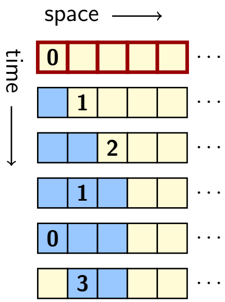
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

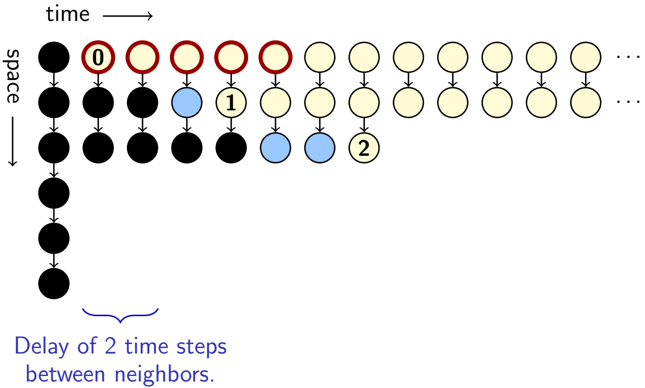
◯ (1) ◯ : nodes "visiting" a Turing cell

Exchanging space and time

Turing machine

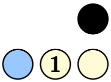


Distributed automaton



alphabet : {□, ■}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

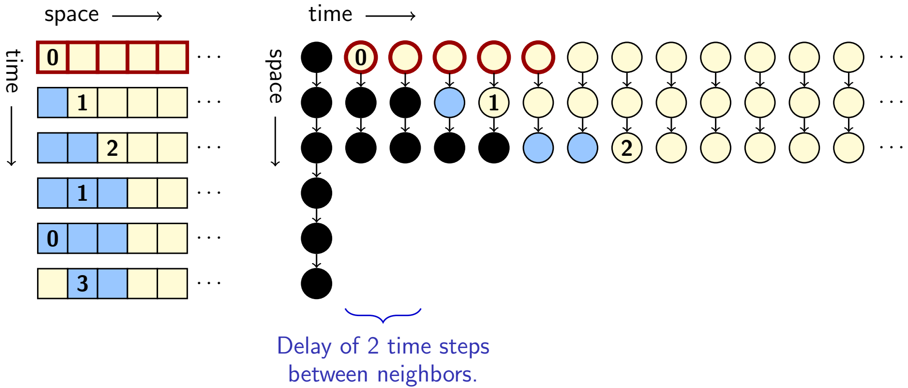


: waiting node

: nodes "visiting" a Turing cell

Exchanging space and time

Turing machine | **Distributed automaton**



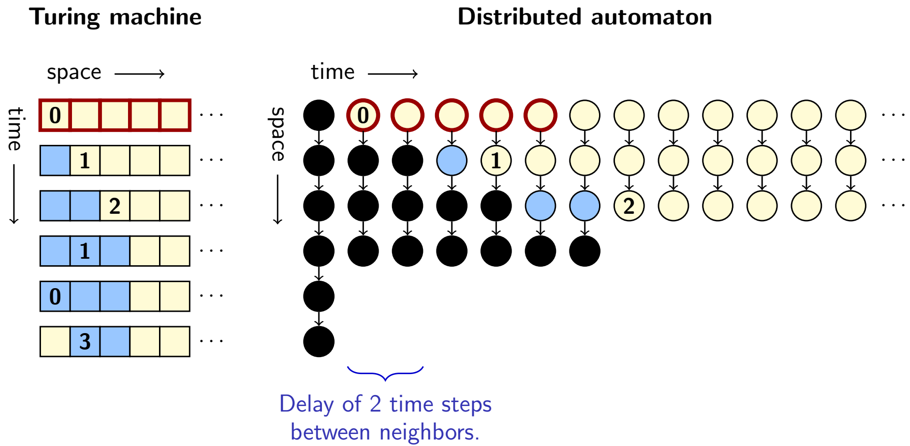
alphabet : {□, □}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

○ (1) ○ : nodes "visiting" a Turing cell

Exchanging space and time



alphabet : {□, □}

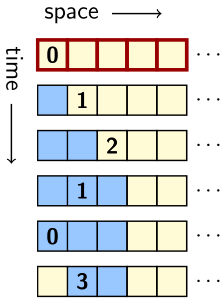
state set : {**0**, **1**, **2**, **3**}

● : waiting node

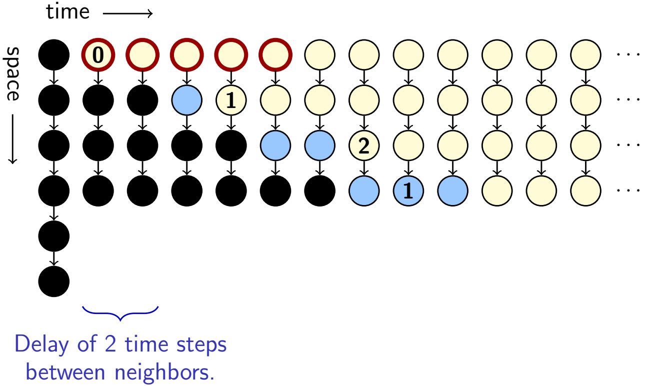
● (1) ○ : nodes "visiting" a Turing cell

Exchanging space and time

Turing machine



Distributed automaton



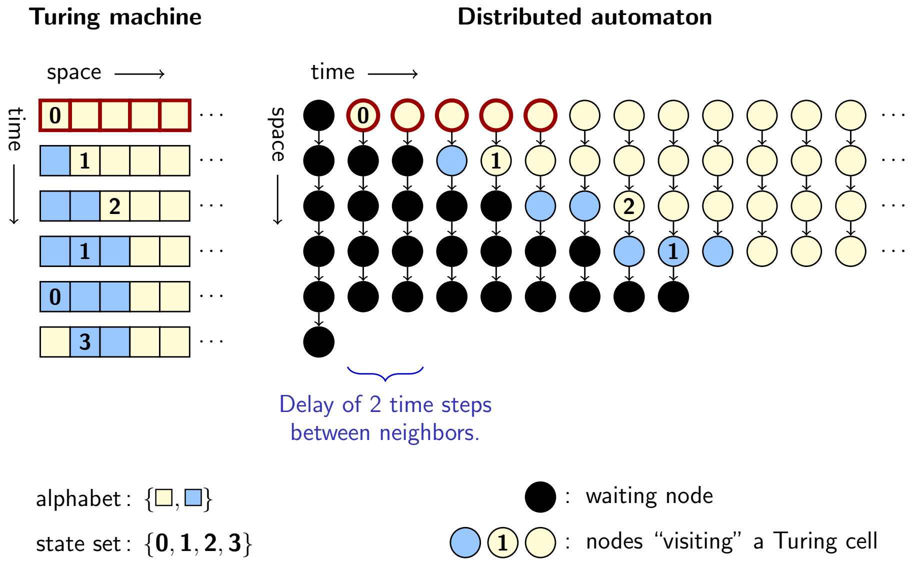
alphabet : {□, □}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

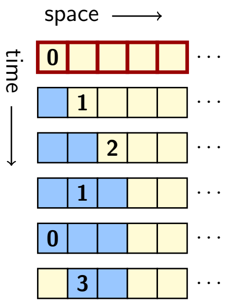
○ (1) ○ : nodes "visiting" a Turing cell

Exchanging space and time

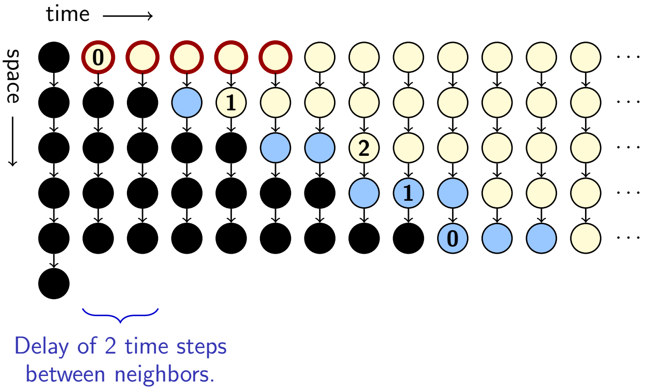


Exchanging space and time

Turing machine



Distributed automaton



alphabet : {□, □}

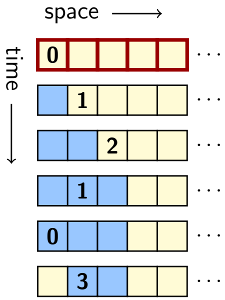
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

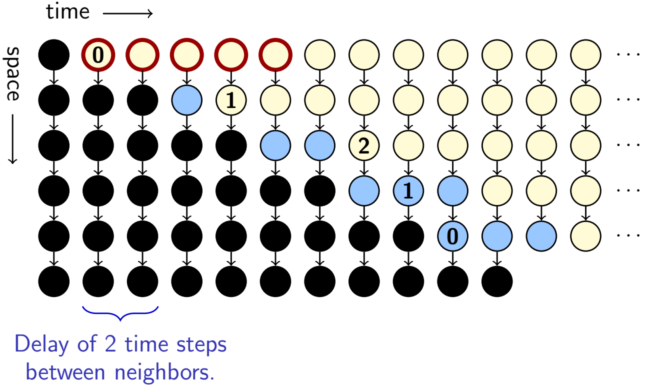
○ (1) ○ : nodes "visiting" a Turing cell

Exchanging space and time

Turing machine



Distributed automaton



alphabet : {□, ■}

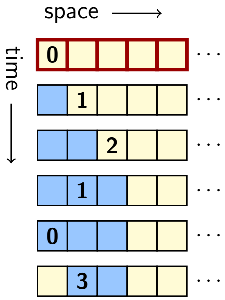
state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

● : waiting node

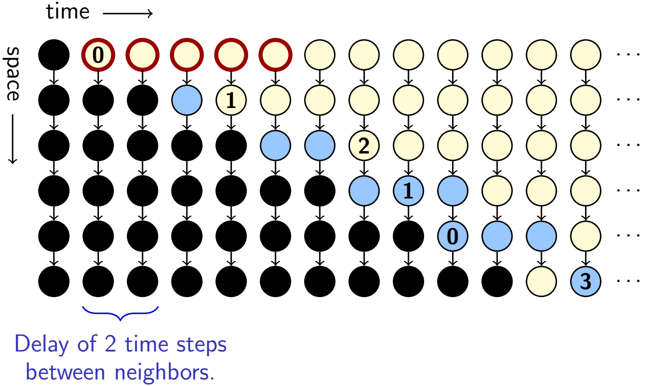
(blue) (1) (yellow) : nodes “visiting” a Turing cell

Exchanging space and time

Turing machine



Distributed automaton



alphabet : {□, □}

state set : $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$

: waiting node

: nodes “visiting” a Turing cell

Results

Results

Emptiness problem undecidable:

Results

Emptiness problem undecidable:

- In general.

Results

Emptiness problem undecidable:

- In general.
- For *quasi-acyclic* automata.

Results

Emptiness problem undecidable:

- In general.
- For *quasi-acyclic* automata.

  ↖ State diagrams acyclic except for self-loops.

Results

Emptiness problem undecidable:

- In general.
- For *quasi-acyclic* automata.

State diagrams acyclic except for self-loops.

Emptiness problem decidable in LOGSPACE:

Results

Emptiness problem undecidable:

- In general.
- For *quasi-acyclic* automata.

  ↖ State diagrams acyclic except for self-loops.

Emptiness problem decidable in LOGSPACE:

- For *forgetful* automata.

Results

Emptiness problem undecidable:

- In general.
- For *quasi-acyclic* automata.

  ↖ State diagrams acyclic except for self-loops.

Emptiness problem decidable in LOGSPACE:

- For *forgetful* automata.

  ↖ Nodes cannot remember their own state.

Results

Emptiness problem undecidable:

- In general.
- For *quasi-acyclic* automata.

State diagrams acyclic except for self-loops.

Emptiness problem decidable in LOGSPACE:

- For *forgetful* automata.

Nodes cannot remember their own state.

On words:	MSO logic	=	forgetful automata

Results

Emptiness problem undecidable:

- In general.
- For *quasi-acyclic* automata.

  ↖ State diagrams acyclic except for self-loops.

Emptiness problem decidable in LOGSPACE:

- For *forgetful* automata.

  ↖ Nodes cannot remember their own state.

On words:	MSO logic	$=$	forgetful automata
On trees:	———	$\subsetneq$	———

Results

Emptiness problem undecidable:

- In general.
- For *quasi-acyclic* automata.

  ↖ State diagrams acyclic except for self-loops.

Emptiness problem decidable in LOGSPACE:

- For *forgetful* automata.

  ↖ Nodes cannot remember their own state.

On words:	MSO logic	$=$	forgetful automata
On trees:	——	$\subsetneq$	——
On digraphs:	——	$\not\subseteq$ $\not\supseteq$	——

Thank you!