Undecidability of the Domino Problem

E. Jeandel and P. Vanier

Loria (Nancy), LACL (Créteil), France

November 20-24

The theorem (again)

Theorem (Berger 1964 (PhD), 1966 (Memoirs of the AMS))

There is no algorithm that decides, given a tileset τ, if τ tiles the plane.

To prove such a statement, we need a formal definition of an algorithm/program/computable function.

Computable functions

Many formal definitions of computable functions:

- with programs (λ-calculus) (Church 1936)
- with axioms (Herbrand-Gödel-Kleene 1936)
- with a mechanical device (Turing 1937)

All definitions are equivalent: there is a single notion of a computable function.

Church-Turing thesis: all reasonable notions of computable functions agree.

Note: these are notions of computable functions, not of algorithms.

Remark on encodings

Computable functions and computable sets are either defined on nonnegative integers or finite words.

To speak about an algorithm that takes tilesets as input, we need to agree on an encoding of tilesets as integers, or finite words.
The exact encoding doesn't matter, as long as it is natural.

Plan

The Turing Machine

The Turing machine is a mathematical abstraction of computations by a human being, introduced by Turing (1937).

Out of all models of computations, the TM is arguably the one most suitable for undecidability proofs, due to its mechanical nature.

The Turing Machine

- The TM works in discrete time steps.
- At each step, the entire work space of the Turing is represented by a one-dimensional *tape*
  - The tape can be simply infinite or biinfinite. Both models are equivalent.
  - Formally, a tape is an element of $A^{\mathbb{N}}$ or $A^{\mathbb{Z}}$, for A some given finite alphabet.

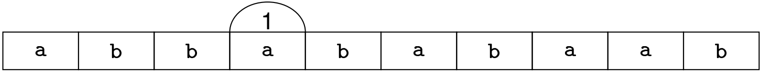
The tape

a	b	b	a	b	a	b	a	a	b

The tape

- A scanning device is installed on the tape. It is called a *head*.
- The scanning device has an internal state, formally an element of a finite set Q

The tape



One step at a time

At each step, the Turing machine operates only depending on what the scanning device can see:

- Its internal state
- The symbol it sees

and may:

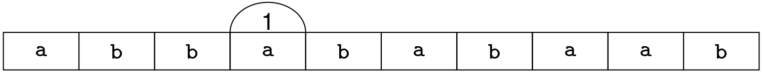
- Change its internal state
- Change the symbol on the head
- Move the head

One step at a time

Formally, this is given by a (finite) map:

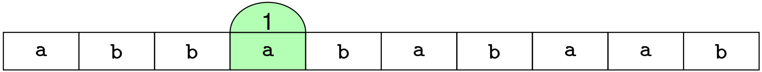
$$\delta : \underset{\substack{\big\uparrow \\ \text{old state}}}{Q} \times \overset{\substack{\text{symbol read} \\ \big\downarrow}}{A} \to \underset{\substack{\big\uparrow \\ \text{new state}}}{Q} \times \overset{\substack{\text{symbol written} \\ \big\downarrow}}{A} \times \underset{\substack{\big\uparrow \\ \text{movement}}}{\{\leftarrow, \rightarrow\}}$$

The tape



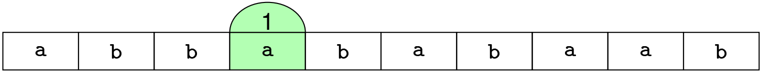
	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

The tape



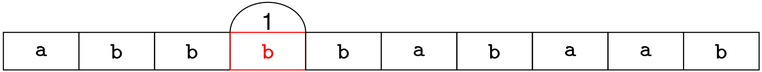
	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

The tape



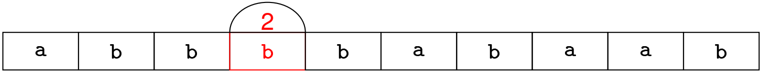
	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

The tape



	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

The tape



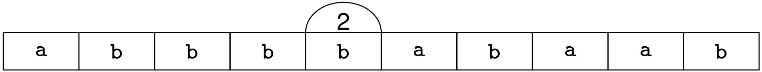
	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

The tape

				2					
a	b	b	b	b	a	b	a	a	b

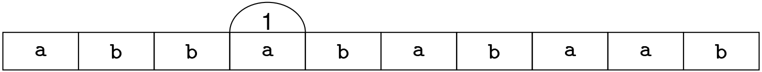
	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

The tape



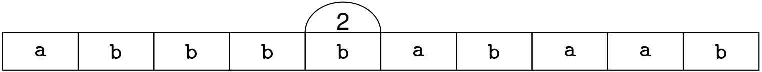
	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

Example



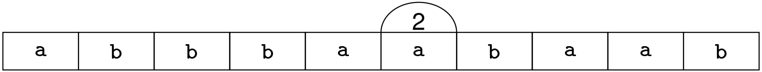
	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

Example



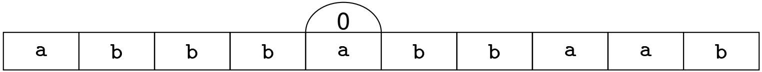
	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

Example



	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

Example



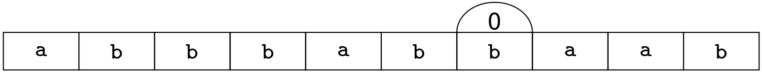
	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

Example

					1				
a	b	b	b	a	b	b	a	a	b

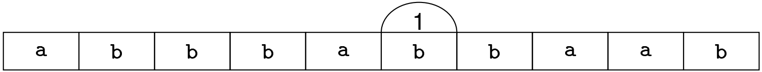
	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

Example



	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

Example

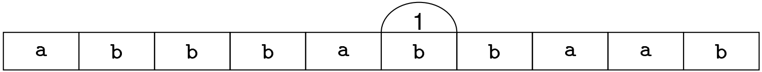


	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

Example

	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

Example



	a	b
0	a,1,$\rightarrow$	b,1,$\leftarrow$
1	b,2,$\rightarrow$	b,0,$\rightarrow$
2	b,0,$\leftarrow$	a,2,$\rightarrow$

What is moving

In the previous example, the head was moving

Dually, we can keep the head in the same place and move the tape

(if the tape is biinfinite)
The approach we will take depends on the context.

Moving head

internal state

$$M : \quad A^{\mathbb{Z}} \times Q \times \mathbb{Z} \to A^{\mathbb{Z}} \times Q \times \mathbb{Z}$$

content of the tape

position of the head

We can switch $\mathbb{Z}$ with $\mathbb{N}$ for simply infinite tapes.

Moving tape

internal state

$$M : \quad A^{\mathbb{Z}} \quad \times \quad Q \quad \rightarrow \quad A^{\mathbb{Z}} \quad \times \quad Q$$

content of the tape

Head always at position 0.

Computation

How to compute with a Turing machine ?

- We write the input on the tape
- We reset the head (at the beginning of the tape, with an initial internal state)
- We run the Turing machine until we go to a special state, called the Halting state
- We read the output from the tape

Computation

A Turing machine defines a function from $\Sigma^\star$ to $\Sigma^\star$ where Σ is the input/output alphabet.

($\Sigma^\star$ is the set of finite words over alphabet Σ)

The function may not be defined everywhere: If the Turing machine does not reach the Halting state, it will not halt, and the function is not defined.

Church-Turing thesis

Any algorithm may be simulated by a Turing

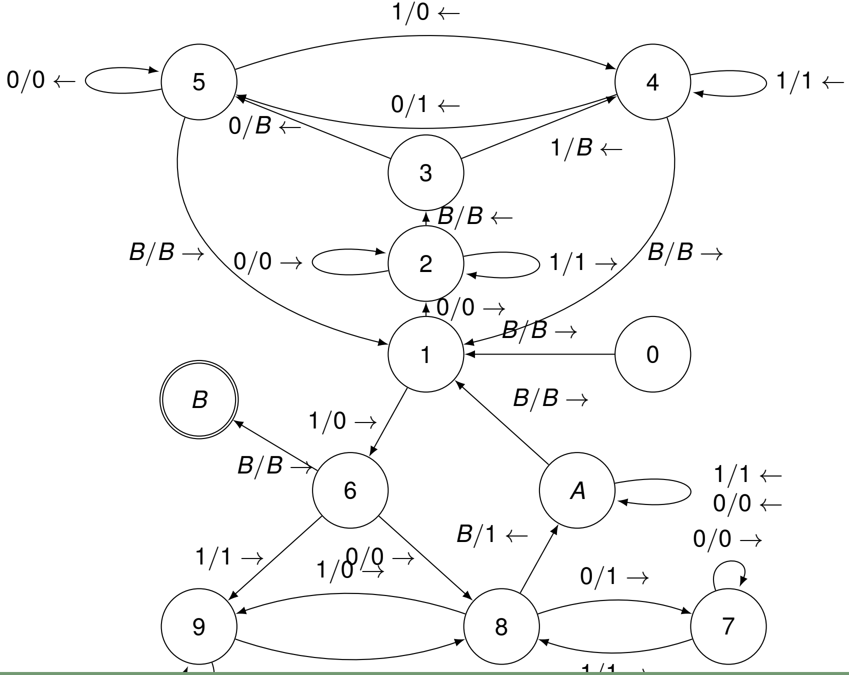
Definition

A function is *computable* if it can be computed by a Turing machine.

Example

	0	1	B
0			B,1,$\rightarrow$
1	0,2,$\rightarrow$	0,6,$\rightarrow$	
2	0,2,$\rightarrow$	1,2,$\rightarrow$	B,3,$\leftarrow$
3		B,4,$\leftarrow$	
4	1,5,$\leftarrow$	1,4,$\leftarrow$	
5	0,5,$\leftarrow$	0,4,$\leftarrow$	B,1,$\rightarrow$
6	0,8,$\rightarrow$	1,9,$\rightarrow$	B,B,$\rightarrow$
7	0,7,$\rightarrow$	1,8,$\rightarrow$	
8	1,7,$\rightarrow$	0,9,$\rightarrow$	1,A,$\leftarrow$
9	0,8,$\rightarrow$	1,9,$\rightarrow$	0,8,$\rightarrow$
A	0,A,$\leftarrow$	1,A,$\leftarrow$	B,1,$\rightarrow$

Initial state: 0
Halting state: B
What is the output starting from `011`?

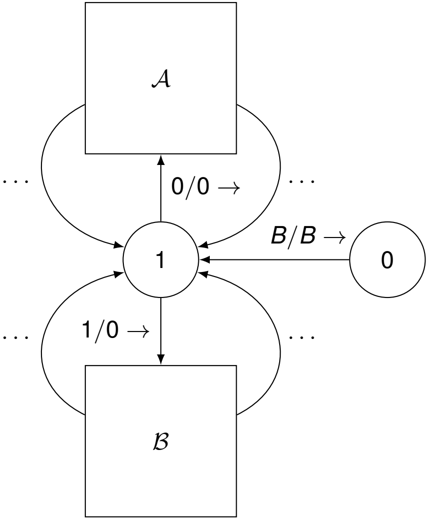

1/0 ←
0/0 ←
5
4
1/1 ←
0/1 ←
0/B ←
1/B ←
3
B/B ←
B/B →
0/0 →
2
1/1 →
B/B →
0/0 →
B/B →
1
0
B
B/B →
1/0 →
B/B →
6
A
1/1 ←
0/0 ←
B/1 ←
0/0 →
1/1 →
0/0 →
1/0 →
0/1 →
9
8
7

How to understand the Turing machine

To understand this particular Turing machine, the input has to be seen as an number represented in binary from least to most significant bit.

So `011` represents the number 6.

Form of the Turing Machine



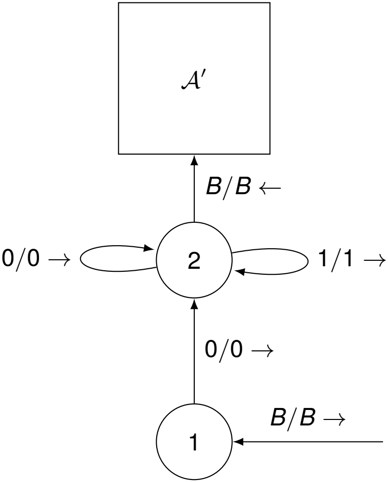
How does this TM work ?

1. If the input begins with `0`
   - do $\mathcal{A}$

else
   - do $\mathcal{B}$

Our input is `011`, so it begins with `0`, let's see $\mathcal{A}$

Inside block A

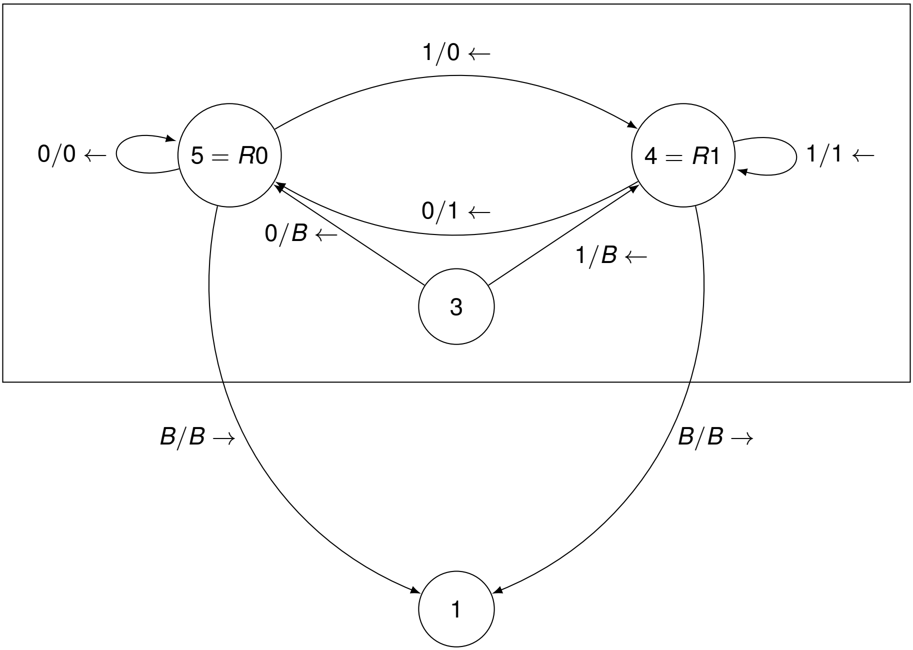


How does this TM work ?

1. If the input begins with 0
   - go to the end of the input
   - do $\mathcal{A}'$

else

- do $\mathcal{B}$


$1/0 \leftarrow$
$0/0 \leftarrow$
$5 = R0$
$4 = R1$
$1/1 \leftarrow$
$0/1 \leftarrow$
$0/B \leftarrow$
$1/B \leftarrow$
3
$B/B \rightarrow$
$B/B \rightarrow$
1

How does this TM work ?

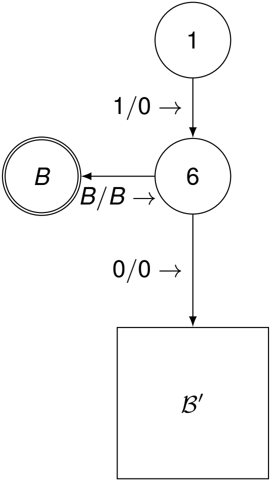
1. If the input begins with `0`
   - go to the end of the input
   - shift the input to the left
   - go back to 1.

else
   - do $\mathcal{B}$

How does this TM work ?

1. If n is even
   - divide n by 2
   - go back to 1.

else
   - do $\mathcal{B}$


1
$1/0 \rightarrow$
B
6
$B/B \rightarrow$
$0/0 \rightarrow$
$\mathcal{B}'$

How does this TM work ?

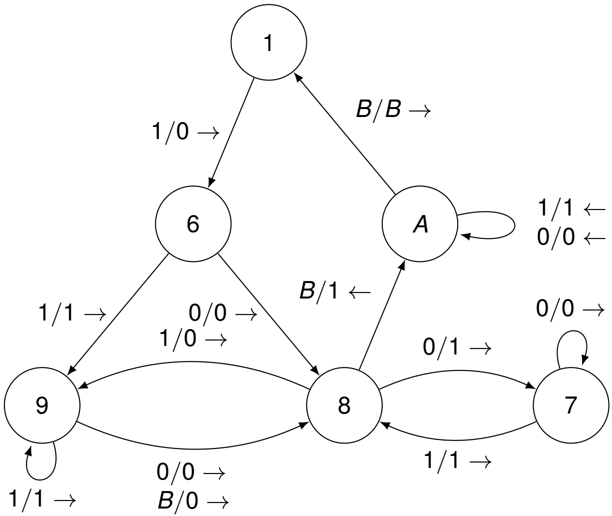
1. If n is even
   - divide n by 2
   - go back to 1.

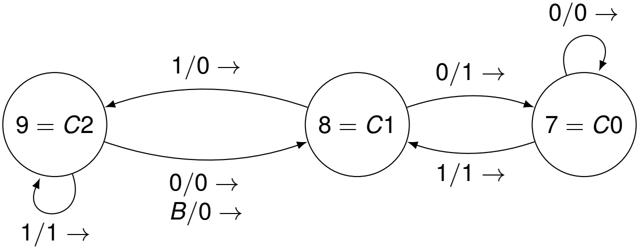
else
   - If $n = 1$
     - Output 0 and halt
   - else
     - Do $\mathcal{B}'$

How does this TM work ?

1. If n is even
   - $n \leftarrow n/2$
   - go back to 1.

else
   - If $n = 1$
     - Output 0 and halt
   - else
     - $n \leftarrow 3n + 1$
     - go back to 1.


1
6
A
9
8
7
1/0 →
B/B →
1/1 ←
0/0 ←
B/1 ←
1/1 →
0/0 →
1/0 →
0/0 →
0/1 →
1/1 →
0/0 →
B/0 →
1/1 →
1/1 →

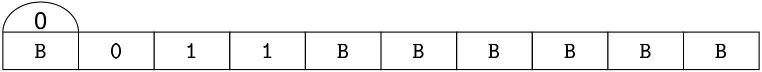

$0/0 \rightarrow$
$1/0 \rightarrow$
$0/1 \rightarrow$
$9 = C2$
$8 = C1$
$7 = C0$
$1/1 \rightarrow$
$0/0 \rightarrow$
$B/0 \rightarrow$
$1/1 \rightarrow$

Conclusion

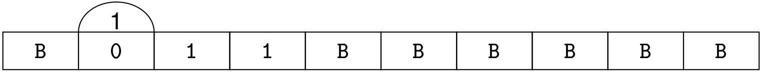
The Turing machine, starting from n (seen as a number coded in binary)

- halts and output 0 if the Collatz sequence starting from n eventually reaches the integer 1
- does not halt otherwise.

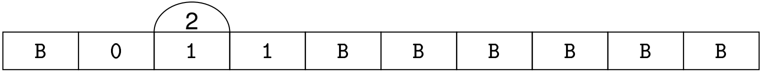
The tape



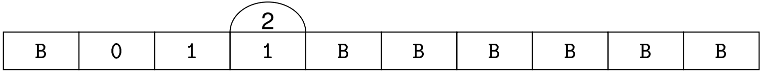
The tape



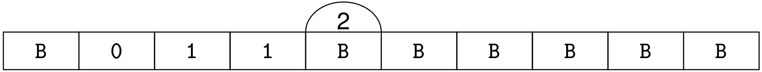
The tape



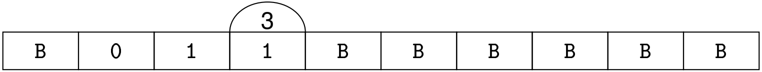
The tape



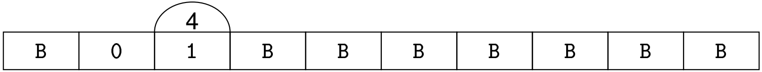
The tape



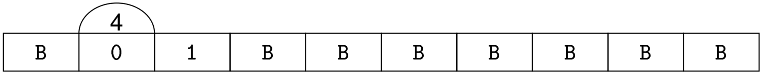
The tape



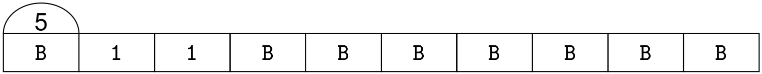
The tape



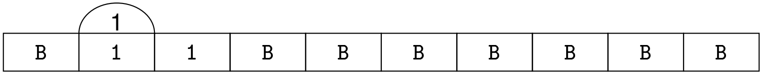
The tape



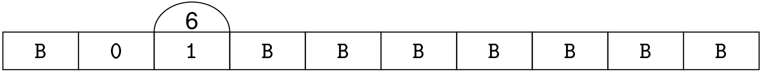
The tape



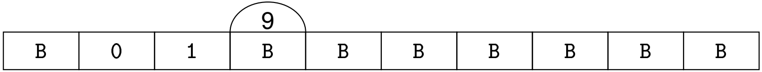
The tape



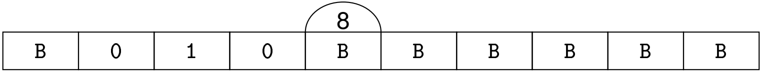
The tape



The tape

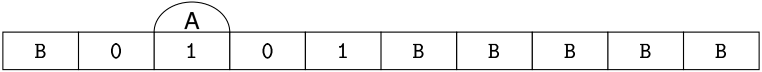


The tape



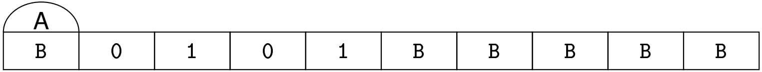
The tape

The tape

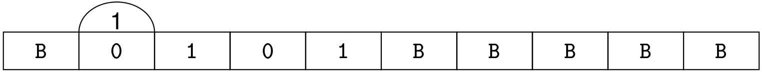


The tape

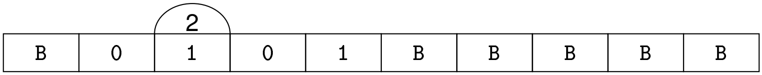
The tape



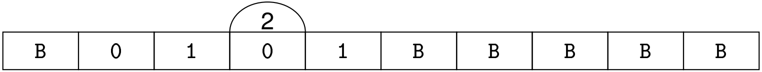
The tape



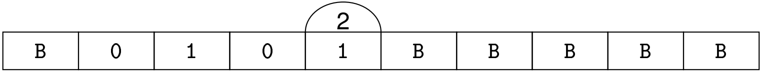
The tape



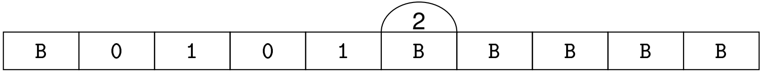
The tape



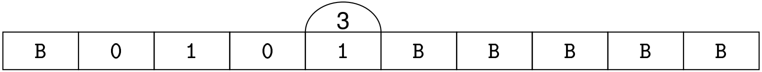
The tape



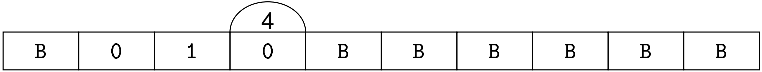
The tape



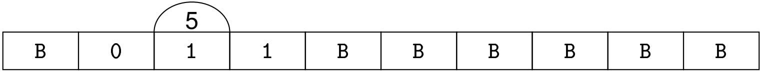
The tape



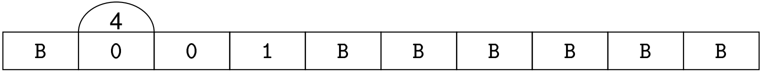
The tape



The tape



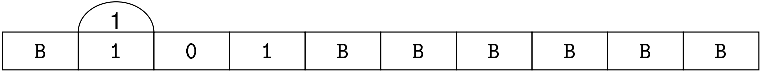
The tape



The tape

5									
B	1	0	1	B	B	B	B	B	B

The tape



Conclusion

With Turing machines, we can code:

- Arithmetic
- Subroutines
- Flow control (`if`, `while`)

""""Therefore""""

We can code any algorithm with a Turing machine

Plan

The theorem (again)

Theorem (Berger 1964 (PhD), 1966 (Memoirs of the AMS))

There is no algorithm that decides, given a tileset τ, if τ tiles the plane.

How to use the concept of a Turing machine to prove this ?

We first need a hard problem on Turing machines

The Halting problem

Theorem

There is no algorithm that can decide, given a Turing machine M and an input n, if M halts on input n.

Proof idea

The set of Turing machines is countable, write M_n for the function computed by the n-th Turing machine

By Cantor's diagonal argument, there exists a function which is not computed by a Turing machine:

$$f(n) = \begin{cases} 0 & \text{if } M_n(n) \text{ does not halt} \\ \mathit{undef} & \text{if } M_n(n) \text{ halts} \end{cases}$$

Proof idea

The set of Turing machines is countable, write M_n for the function computed by the n-th Turing machine

By Cantor's diagonal argument, there exists a function which is not computed by a Turing machine:

$$f(n) = \begin{cases} 0 & \text{if } M_n(n) \text{ does not halt} \\ \textit{undef} & \text{if } M_n(n) \text{ halts} \end{cases}$$

The only thing we need to turn f into an algorithm is to decide in which case we are

Proof idea

The set of Turing machines is countable, write M_n for the function computed by the n-th Turing machine

By Cantor's diagonal argument, there exists a function which is not computed by a Turing machine:

$$f(n) = \begin{cases} 0 & \text{if } M_n(n) \text{ does not halt} \\ \textit{undef} & \text{if } M_n(n) \text{ halts} \end{cases}$$

Therefore no algorithm can decide in which case we are

The Halting problem revisited

Theorem

There is no algorithm that can decide, given a Turing machine M, if M halts on the empty input.

(Given a machine M and an input n, we can build a machine M^n s.t. M^n on the empty input simulates M on input n.)

How to use it ?

We will build an algorithm that, starting from a Turing machine *M*, will build a tileset τ s.t.

Deciding if τ tiles the plane is the same as deciding if *M* halts on the empty input.

Therefore no algorithm can decide if a tileset tiles the plane, as it would be able to decide if a Turing machine halts.

How to use it ?

- It is easy to see that a tileset τ does not tile the plane: just find an N s.t. it does not tile a $N \times N$ square
- It is easy to see that a Turing machine halts : just find an N s.t. it halts in N steps.

Therefore our transformation should satisfy:

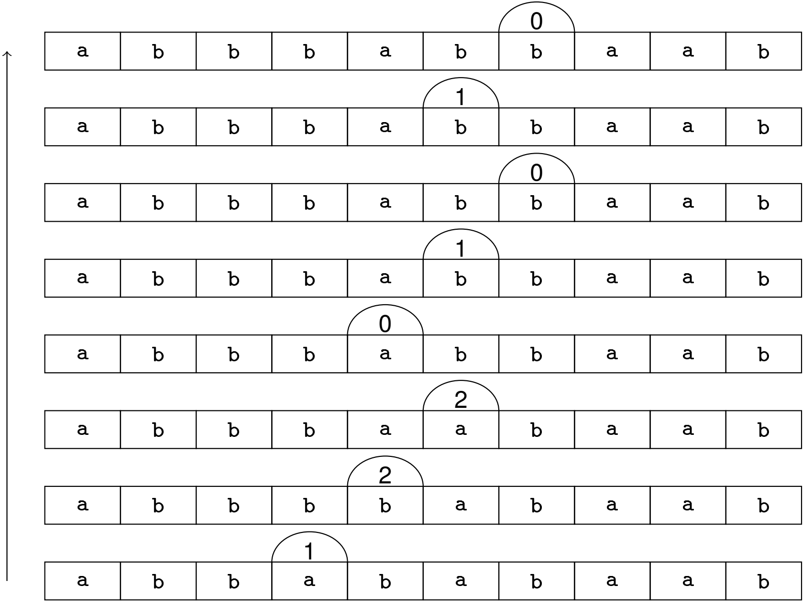
τ tiles the plane iff M does not halt on empty input.

How to do it?

How to transform a Turing machine into a tileset ?

It is actually very easy! (sadly with a *caveat*).

Space-time diagram

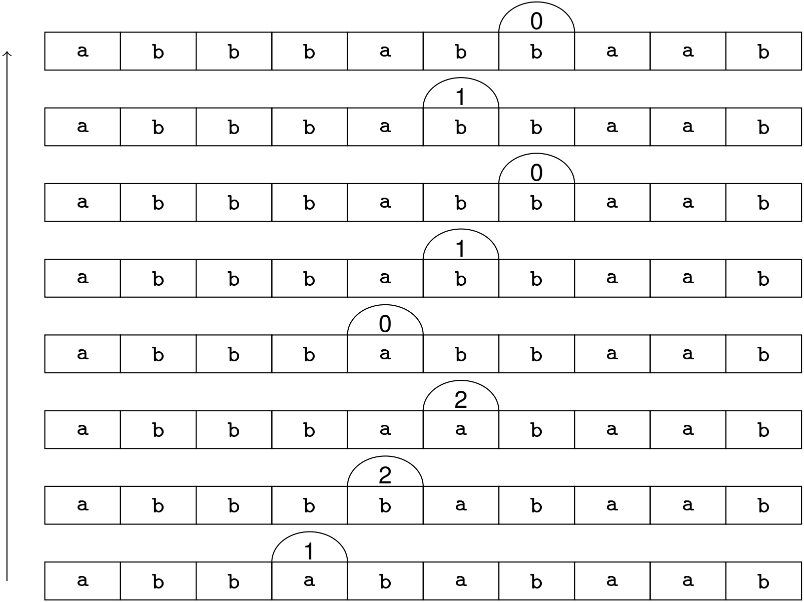


Space-time diagram

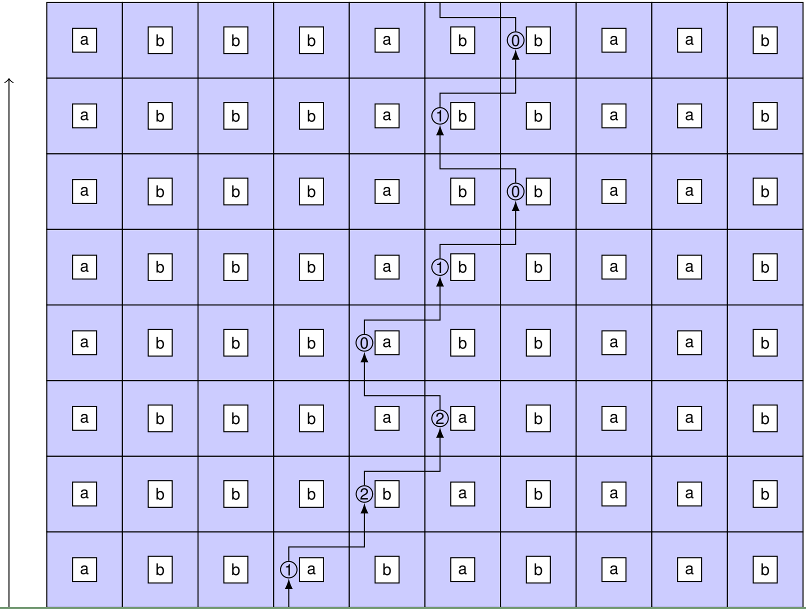
The space-time diagram of a Turing machine is almost a tiling.

- Every constraint can be expressed locally

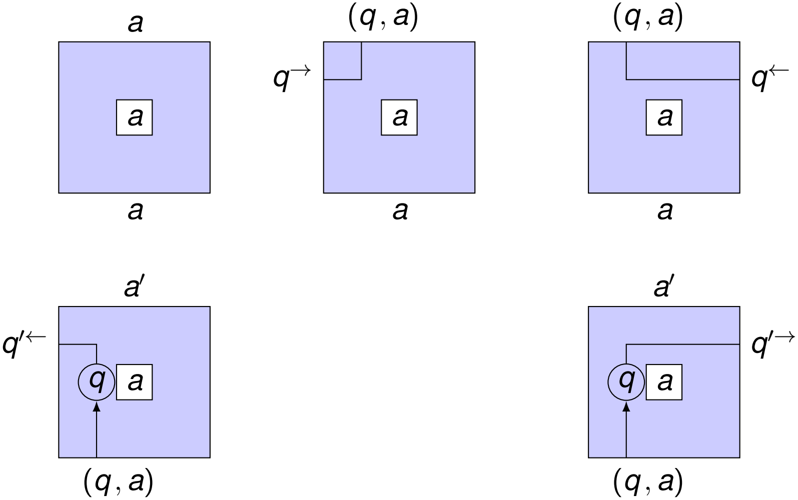
Space-time diagram



Space-time diagram



Tiles



if $\delta(q, a) = (q', a', \leftarrow)$ if $\delta(q, a) = (q', a', \rightarrow)$

The encoding

Every infinite computation of a Turing machine gives rise to a tiling of the upper half plane

It can be completed into a tiling of the entire plane by adding “blank tiles” in the bottom half plane.

Bad news

Not every tiling of the plane is an infinite computation of a Turing machine.

Problems

Erratic configurations:

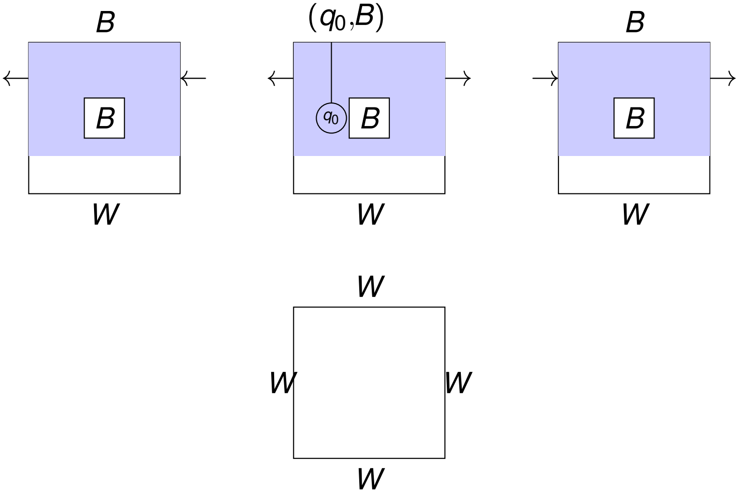
- tilings with no head per row
- tilings with more than one head per row
- tilings may no start from the initial state
- tilings using only the blank tile

Good news

If one row of a tiling represents a configuration of the TM, then the upper lane above it is correct.

If we can force one row, we can force everything above.

Semi-solution



The Fixed Domino Problem

Let t be the tile at the middle of the previous slide.
Then there exists a tiling that contains t iff the TM does not halt on the empty input.

Theorem (Wang, Kahr-Moore-Wang)

There is no algorithm that decides, given a tileset τ and a tile t whethere there exist a tiling by τ that contains t.

Variants

Theorem

There is no algorithm that decides, given a tileset τ and a tile t whethere there exist a tiling of a quarter of the plane by τ with t at the bottom left.

Use a Turing machine with a simply infinite tape, and “border tiles”.

Theorem

There is no algorithm that decides, given a tileset τ and two tiles t_1, t_2 if there is a tiling of a square by τ with t_1 at the bottom left and t_2 at the top right.

Use border tiles to form a square, and take t_2 to contain the halting state of M.

The Domino Problem

What is missing to prove the undecidability of the Domino Problem ?

The Domino Problem

Two methods:

- Find a way to force the tile t to appear
- Change the coding to be certain that a head will appear in every row
  - Does not solve everything.

We will explain this next time.

Plan

The Periodic Domino Problem

We are missing some ingredients for the Domino Problem, but we are almost ready to prove the undecidability of the Periodic Domino Problem.

Theorem (Gurevich-Koryakov)

There is no algorithm that decides, given a tileset τ whethere there exist a periodic tiling by τ.

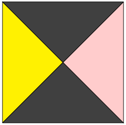
First Ingredient

Assume there is an aperiodic tileset.

Periodic tilings ?

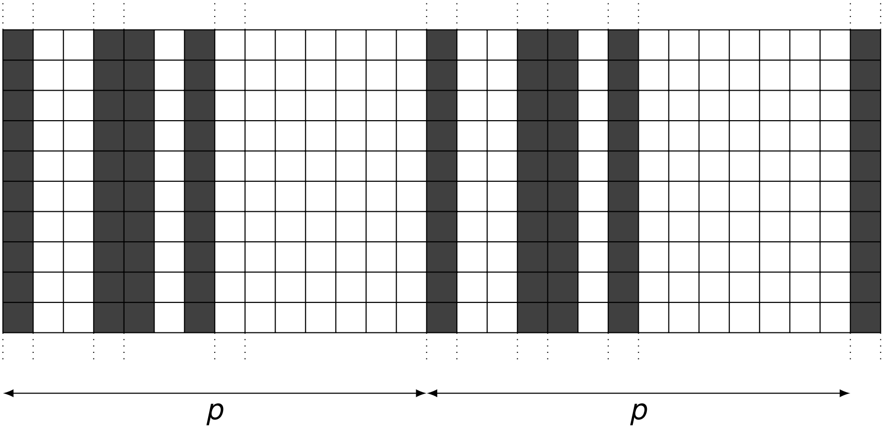
None

Second Ingredient

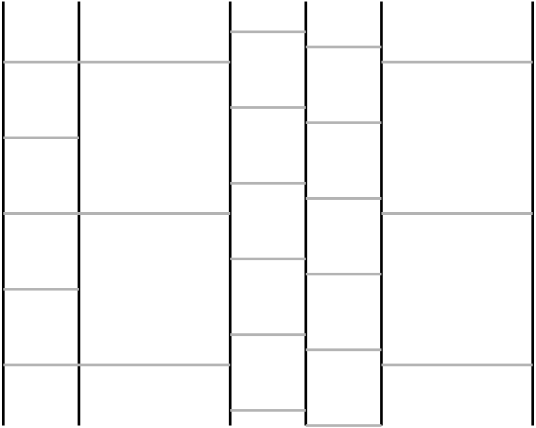


for each color that appear in the aperiodic tileset.

Periodic tilings ?

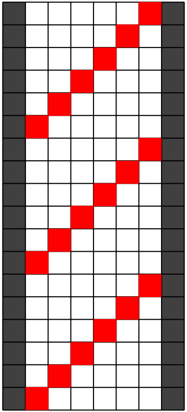


Third Ingredient. Goal



Third Ingredient. Concept

A red particle that teleports once it reads a vertical wall



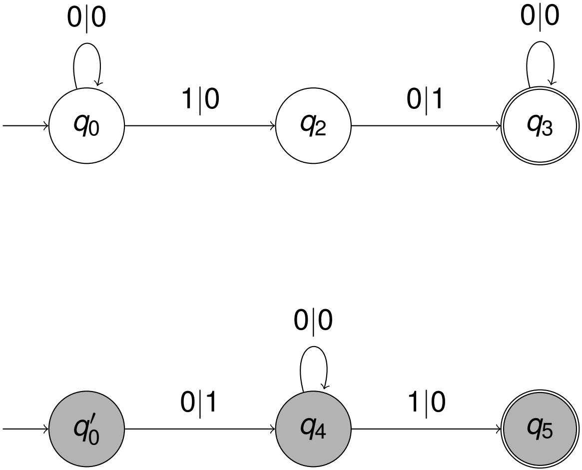
Third Ingredient. Formalization

The particle is a 1, the void is 0

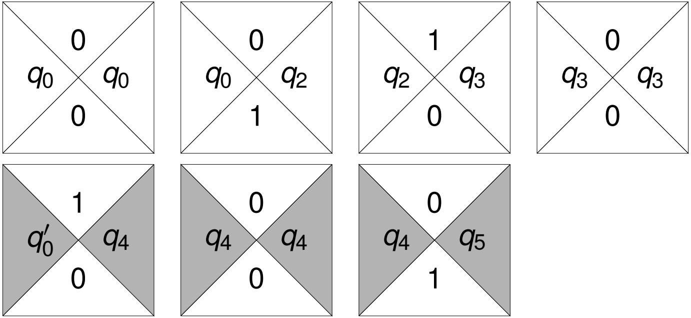
Each line is therefore a word in $0^\star 1 0^\star$.

Third Ingredient. A transducer

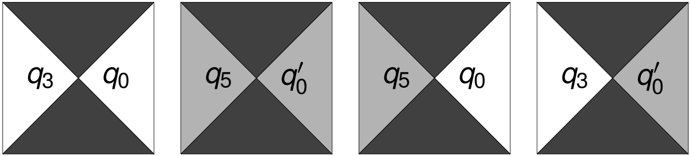
The transducer takes one line to the next one.



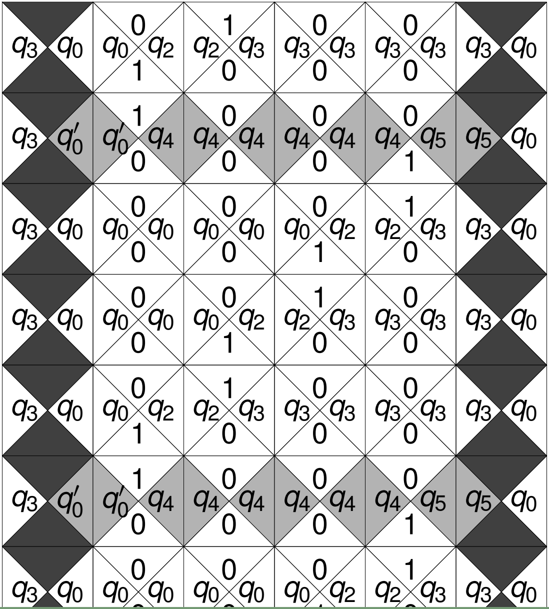
Third Ingredient. White tiles



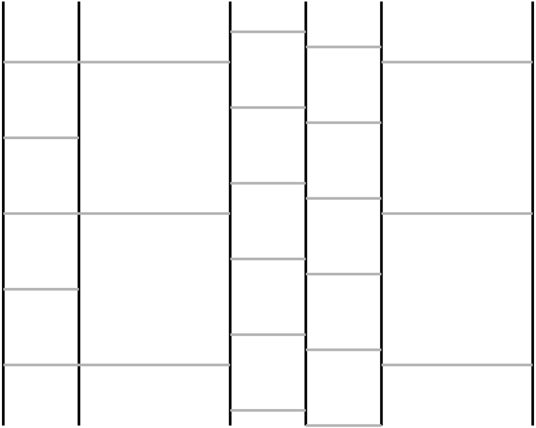
Third Ingredient. Black tiles



Third Ingredient. Proof of concept



Third Ingredient. Goal: ACHIEVED



Fourth Ingredient

The previous result:

Theorem

There is no algorithm that decides, given a tileset τ and two tiles t_1, t_2 if there is a tiling of a square by τ with t_1 at the bottom left and t_2 at the top right.

Fourth Ingredient

Given a tileset τ with colors in C, and tiles t_1, t_2, superimpose the tileset τ with the tileset we built:

- Every tile inside a square can hold an element of τ
- Tiles on the border of squares can hold anything with colors in C (even if not in τ)
- The tile on the bottom-left on the square should be t_1.
- The tile on the top-right on the square should be t_2.

Last condition are easy to ensure, as the corners can be spotted easily.

The proof

There exists a tiling of period n

$\downarrow$

This tiling contains a square of size p

$\downarrow$

This square contains a tiling of τ
with t_1 at the bottom left
and t_2 at the top right

The proof

There exist a tiling of a square by τ
with t_1 at the bottom left
and t_2 at the top right.

$$\downarrow$$

There exist a tiling of period $p + 1$

Next time

The proof of the undecidability of the Domino Problem (finally!)