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.

Last time

- Algorithms can be represented by Turing machines
- Turing machines can be encoded into tilings

However, the encoding has a default: it needs a “catalyst” tile to launch the computation.

Proof of the Undecidability of the Domino Problem

Proofs fall into 4 categories:

- Berger (64)-Robinson (71) constructions
- Aanderaa-Lewis (74)
- Kari (07)
- Durand-Romashchenko-Shen (08)

Many, many variants of Berger-Robinson.

Proof of the Undecidability of the Domino Problem

Two different ways to solve the caveat we saw on tuesday:

- Force the catalyst tile to appear (Berger,Robinson,Aanderaa-Lewis,DRS)
- Use a different encoding of Turing machines (Kari)

We will present here the two methods

Proof of the Undecidability of the Domino Problem

Remark

- Everything I'm presenting now is very well known
- The lecture notes have a third proof (Aanderaa-Lewis) which is almost completely unknown

If you know everything about Kari and Robinson, read the lecture notes!

Plan

Historical Notes

- Inspired by the proof of Berger and Robinson, using ideas of Ollinger for simplification
- Robinson’s proof is not that different from Berger
    - Berger has an aperiodic set of 103 tiles in his PhD

The Catalyst tile

We want to force a specific tile to appear so that it can kickstart the computation

Reminder

A tileset τ tiles the plane iff it tiles arbitrarily large squares

Corollary

If every tiling of the plane contains a tile t, then there exists n s.t. t appears in every $n \times n$ square

The Catalyst tile

The specific tile *t* should appear *everywhere*.

There should be infinitely many computations in one tiling of the plane.

How to do it

To obtain the Undecidability of the Domino Problem:

- Find an aperiodic tileset τ
- Look at occurrences of a specific tile *t* inside this τ
- Build computations around this tile, hoping that they do not overlap

How to do it

How do you obtain an aperiodic tileset ?

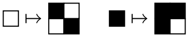
Robinson-Berger: Use substitution-like tilings.

Substitutions

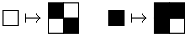
Definition

A substitution is a map ϕ from A to $A^{n \times n}$ for some n.
Given a substitution ϕ and an initial pattern w, we can iterate ϕ on w to obtain arbitrarily large patterns.

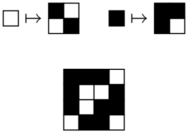
Example



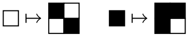
Example

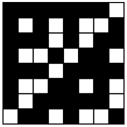


Example

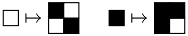


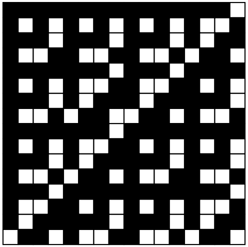
Example



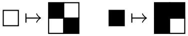


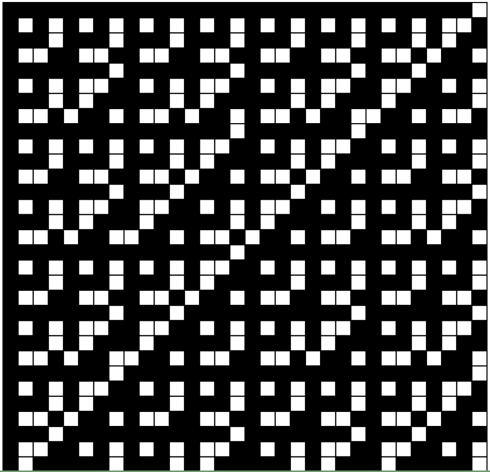
Example



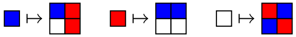


Example

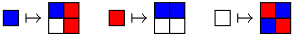




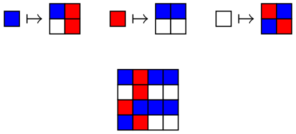
Example



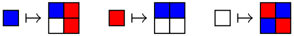
Example

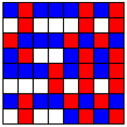


Example

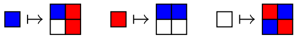


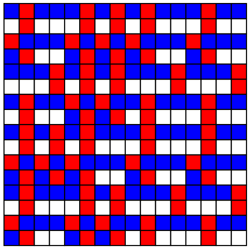
Example



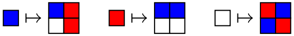


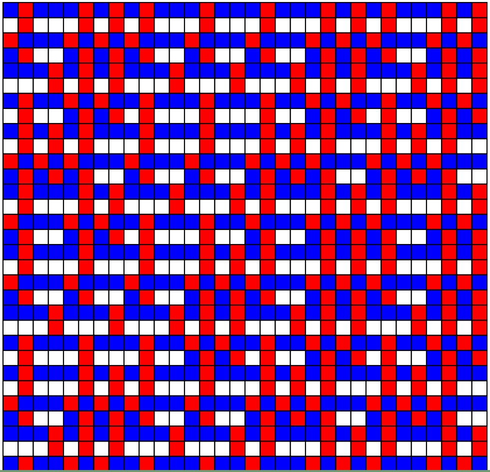
Example





Example





Substitutive tilesets

Definition

The subshift S_ϕ associated to ϕ is the set of colorings of the plane that look like $\phi^k(w)$ for some large value of k.

(Of course not an exact definition)

Definition

A tileset τ is substitutive if tilings by τ look like S_ϕ up to recoloring.

A lot of different substitutive tilesets in the literature. But how can you obtain them ?

Intrinsically substitutive tileset

Definition

A tileset τ is intrinsically substitutive if the substitution ϕ is defined directly on the tiles of τ, and encoded directly into the local constraints of τ.

Intrinsically substitutive tileset

What do we mean exactly ?

- $\phi(w)$ is a valid tiling iff w is a valid tiling

There is a one-to-one correspondence between the colors on the east side of τ and colors on the east side of $\phi(\tau)$.

- We can desubstitute: For every tiling x by τ, there exists y s.t $\phi(y) = x$

All* tilings of a 2×2 square are of the form $\phi(t)$ for $t \in \tau$

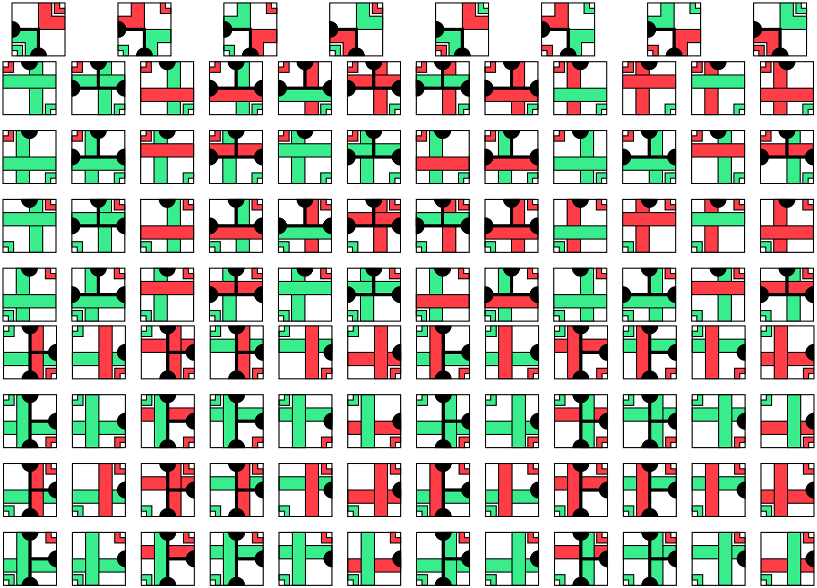
Intrinsically substitutive tileset

Theorem (Durand-Levin-Shen 2004, Ollinger 2008)

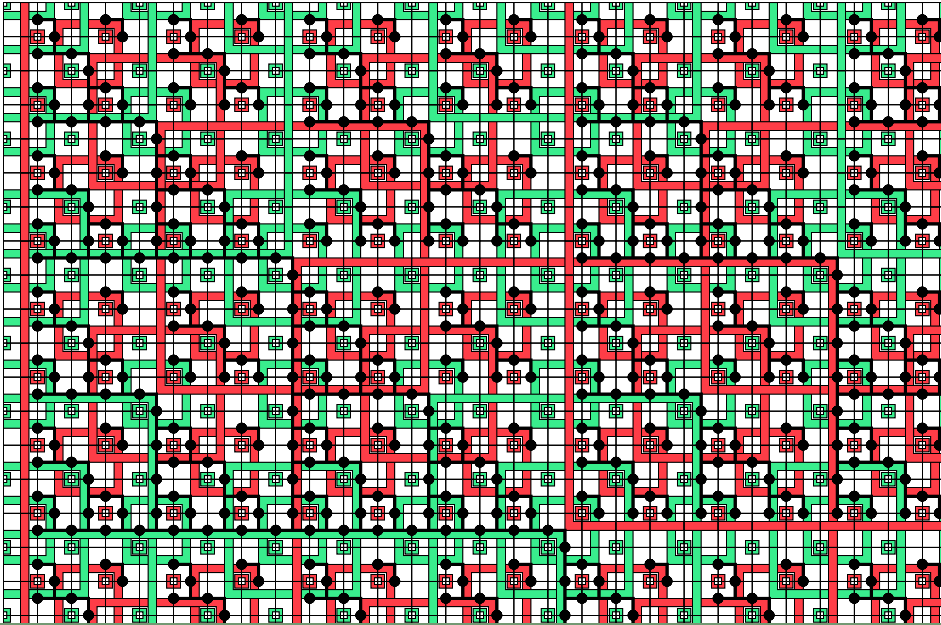
There is an intrinsically substitutive tileset

These are the first proven examples. However Berger's set of 103 tiles IS intrinsically substitutive, and Robinson's set ISN'T.

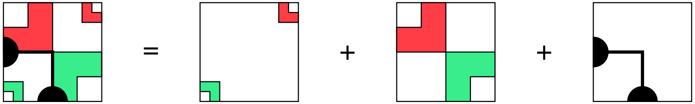
The tileset of Ollinger

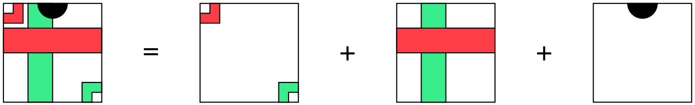


A tiling

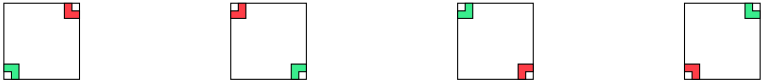


How does it work



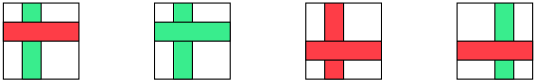


First layer

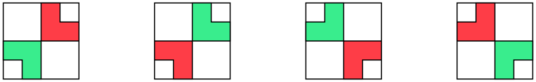


The first layer forces tiles to be grouped into 2×2 squares.

Second layer



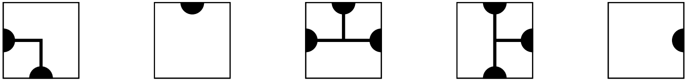
(16 different tiles)



(4 different tiles)

The second layer is the image of the first layer by the substitution (more on this later)

The third layer



The third layer codes the “chair substitution”.

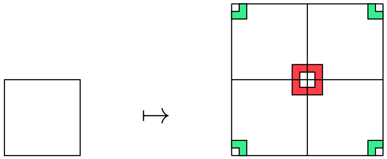
Substitution

ϕ is a 2×2 substitution. When substituting, the original tile is “found” on the bottom-left of the 2×2 square.

- Layer 1 of w is encoded into layer 2 of $\phi(w)$
- Layer 2 and 3 of w are enlarged into $\phi(w)$

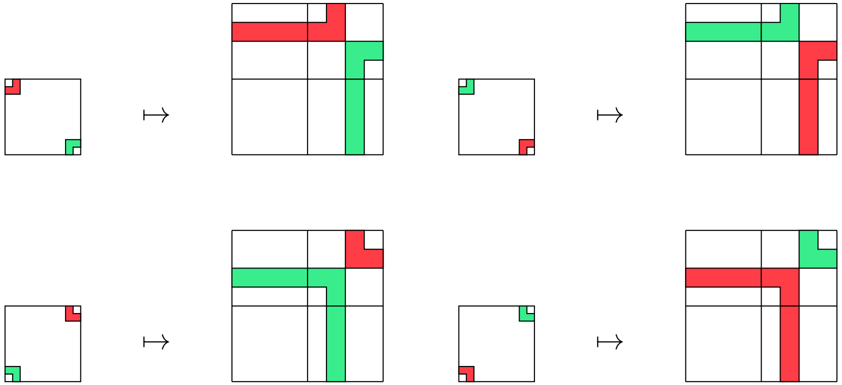
Layer 0 $\rightarrow$ Layer 1

Every $\phi(w)$ is of the form:



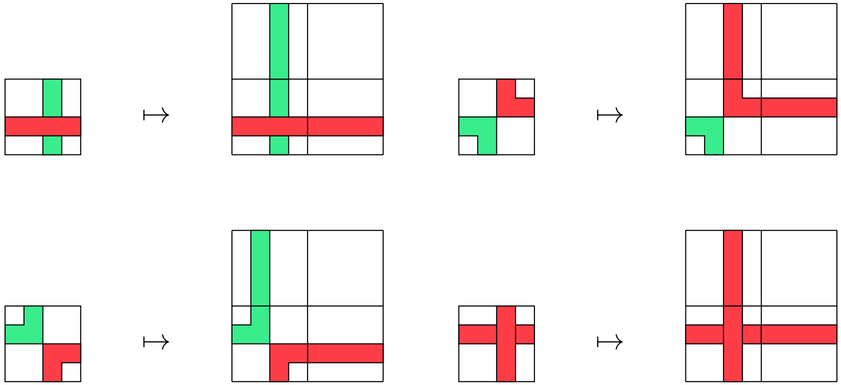
Layer 1 $\to$ Layer 2

Layer 1 is enlarged into Layer 2



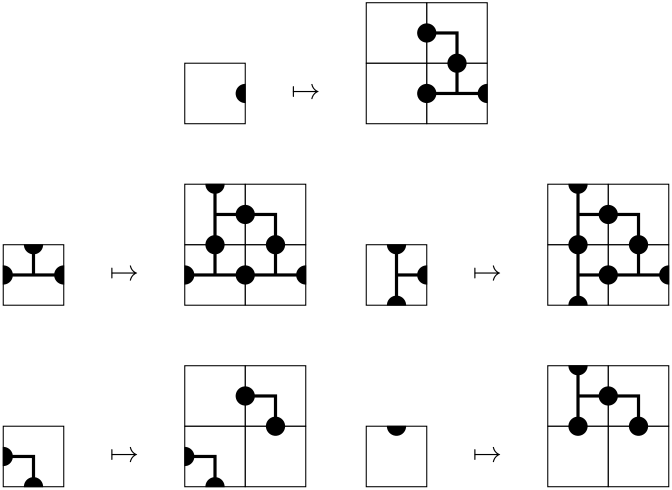
Layer 2 $\rightarrow$ Layer 2

Layer 2 is propagated into the adjacent tiles

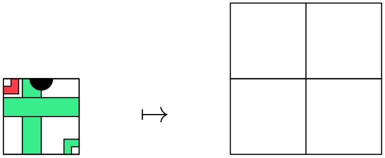


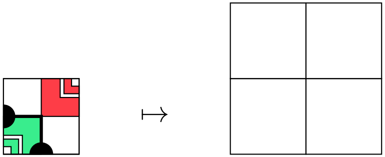
Layer 3 $\to$ Layer 3

Layer 3 is also propagated

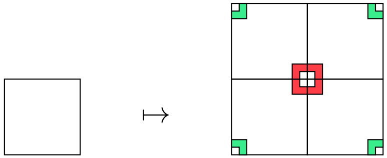


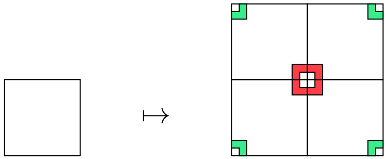
Two examples



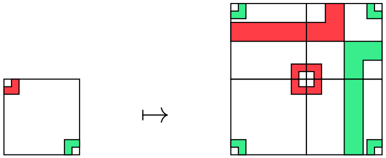


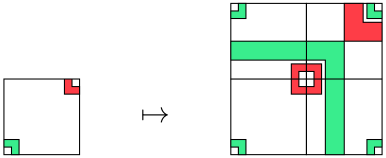
Two examples



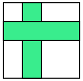


Two examples

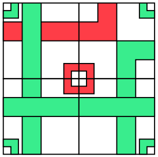


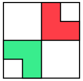


Two examples

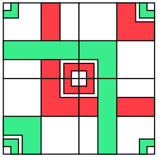


$\mapsto$

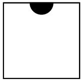




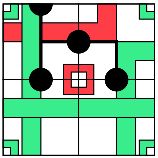
$\mapsto$

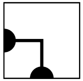


Two examples

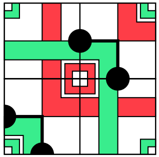


$\mapsto$

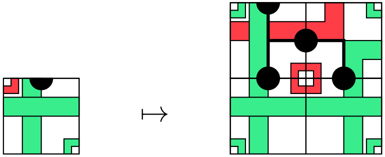




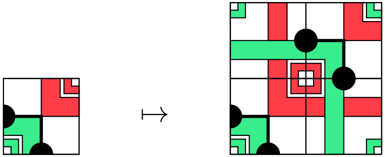
$\mapsto$



Two examples



$\mapsto$



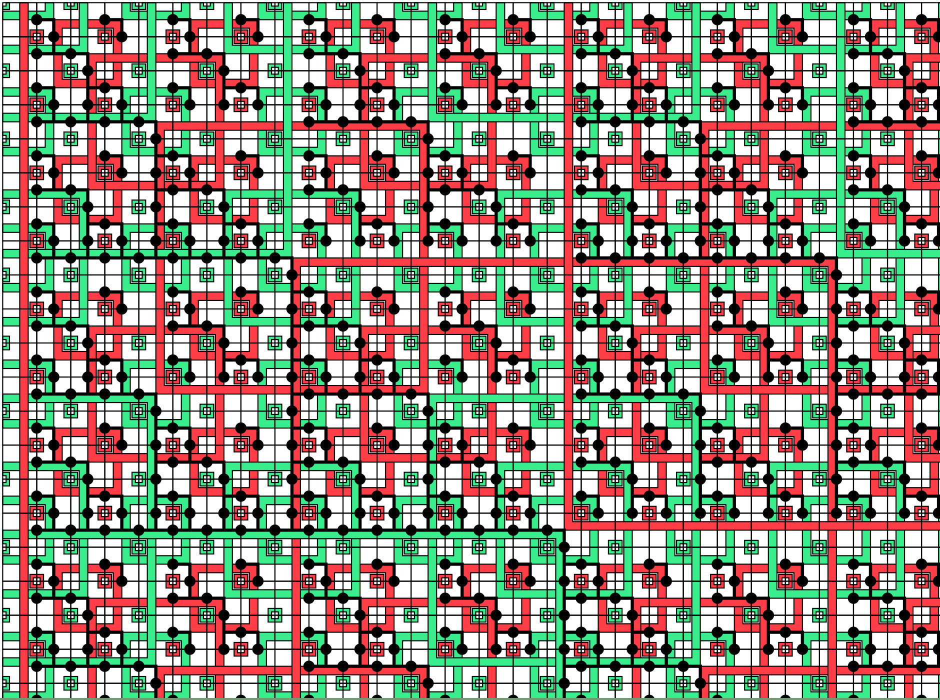
$\mapsto$

Theorem

Theorem (Ollinger 2008)

The tileset τ is intrinsically substitutive

Proof: Just* inspect all 2×2 patterns.



Theorem

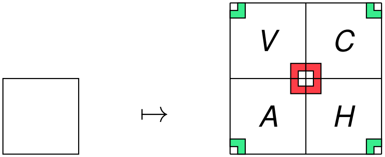
The fact the tileset is substitutive *proves* it is aperiodic but does not *explain* everything

Why do we need the third layer ?

What are the rules between the layers ?

The substitution again

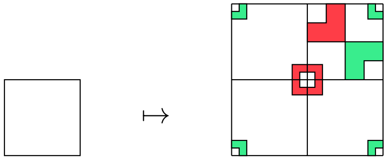
Every $\phi(w)$ is of the form:



A: any tile
C: corner tile
H: horizontal tile: propagate signal from *A* horizontally
V: vertical tile: propagate signal from *A* vertically

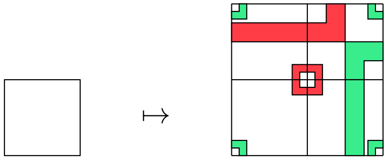
The substitution again

What if we have something of the form



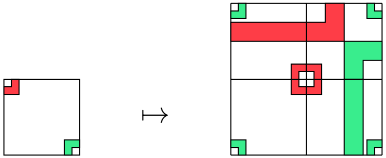
The substitution again

What if we have something of the form



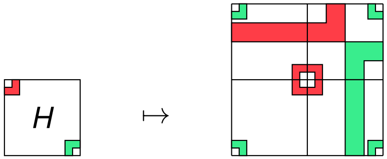
The substitution again

What if we have something of the form



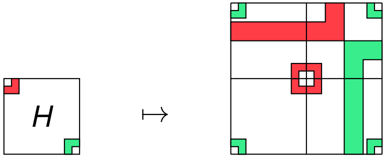
The substitution again

What if we have something of the form



The substitution again

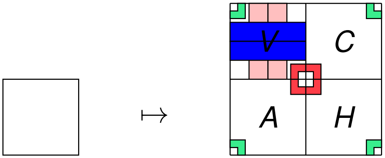
What if we have something of the form



The corner at the top right of the square determines what kind of tiles we can put at the bottom left of the square

That's what the third layer is ensuring.

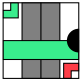
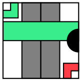
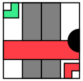
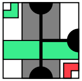
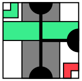
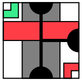
More details



What color appear on the blue part determines which corner is in *C*.

Rules for V

Here are the possible rules for the tile at the top left (tile of type V):

Similar rules for the tile at the bottom right.

Theorem

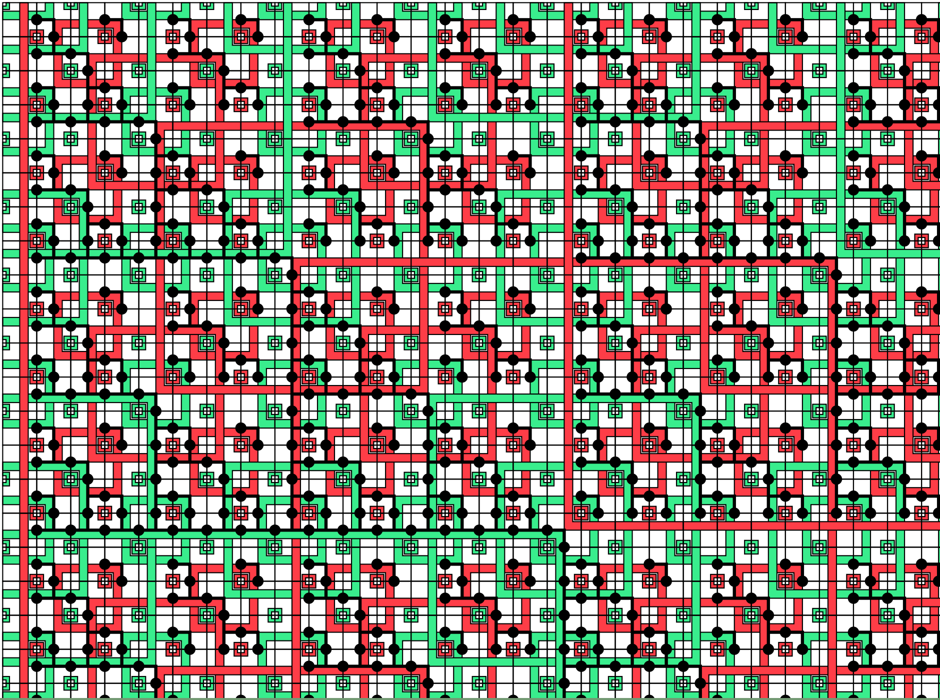
Theorem (Ollinger 2008)

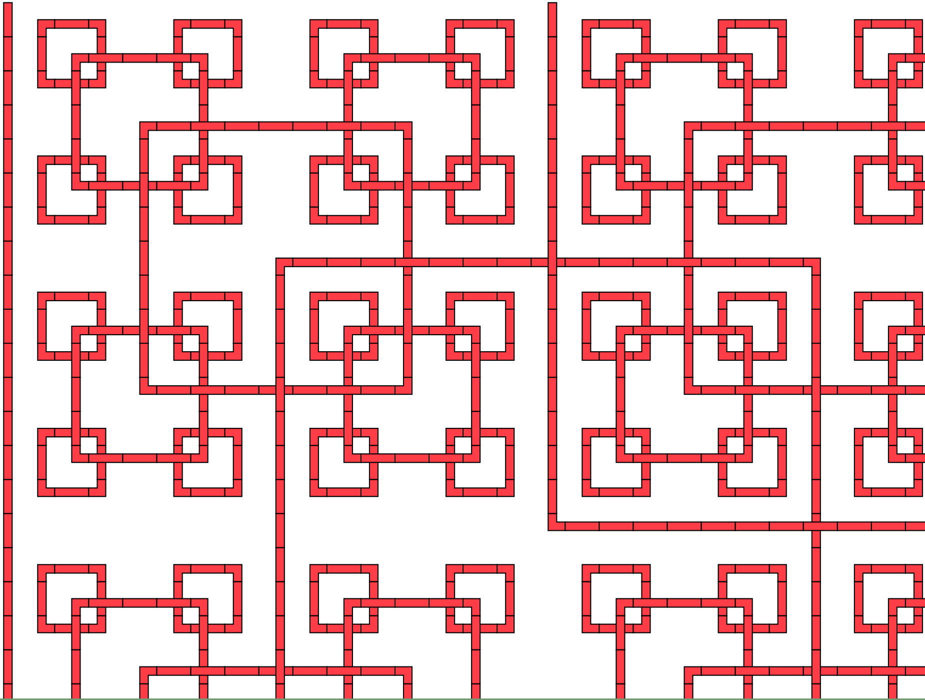
The tileset τ is instrinsically substitutive, and we more or less understand why.

What do we do now ?

We will encode computations inside the tilings s.t. there is a tiling iff some Turing Machine does not halt.

First, we simplify the tilings by looking only at the red color at layer 2





What we have

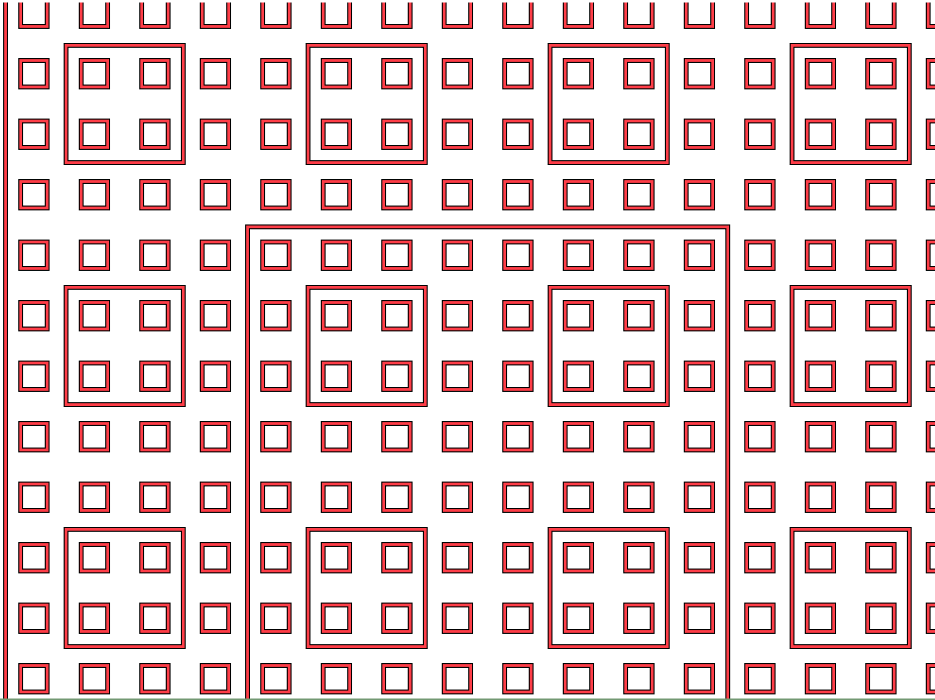
We obtain squares in which we could put a Turing machine

Squares have a designated corner in which we could put the catalytic tile.

Problem

Squares intersect

Solution: Keep only half the squares.
Technically: two different shades of red (dark/light), dark red can only cross light red.

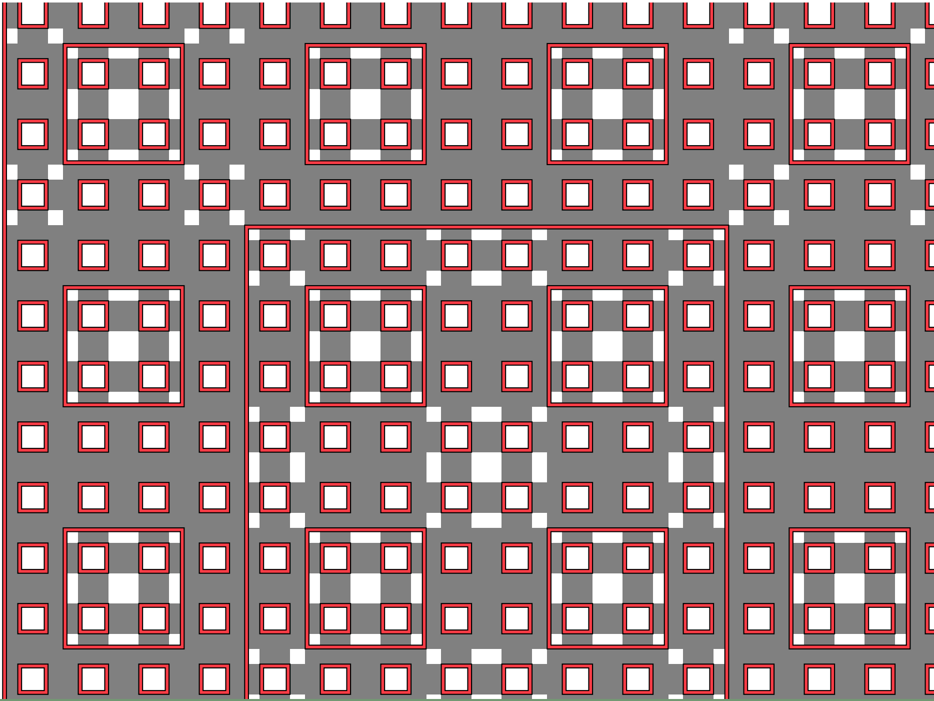


Problem

Squares inside squares.

We cannot change this

Technical trick due to Robinson: Use on bigger square only the space that is not hindered by the smaller squares.



What to do now

- We now put our Turing machine in the space left blank
- It will be initialized by the bottom left corner of a square, which is always free (blank)
- The free (blank) space inside a square is not connected, however we can use the gray cells to transmit information(colors) between two blank cells on the same row/column
- Lot of technical details.

Idea

- Suppose there is a tiling of the plane
- Then there are tilings of arbitrarily large red squares
- These squares contains computation of the Turing machine for an arbitrarily long time
- Therefore the Turing machine does not halt

Idea

- Suppose the TM does not halt
- Build a tiling of the plane by putting in each red square the beginning of the computation of the TM.

Therefore there exists a tiling iff the TM does not halt.

Remarks

- Essentially Robinson’s proof
    - Robinson is not substitutive. Ollinger is a cover of Robinson that is intrinsically substitutive.
- Berger’s proof builds infinite vertical strips rather than squares
    - The vertical strips overlap, so same trick is used
- Berger’s set of ~~104~~ 103 tiles is almost the same as Robinson/Ollinger !

Plan

Kari construction

Main idea :

Use a different encoding of Turing machines into tilings that do not have the problems we encountered

Problems (from last time)

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

New encoding

This new encoding (to be defined) forces that there is exactly one head per row.

Which problem remains ?

The Immortality Problem

Theorem (Hooper 1966)

There is no algorithm that decides, given a Turing machine, if there exists SOME configuration on which it runs forever.

Configuration can be arbitrarily (arbitrarily initial state, arbitrarily initial filling of the tape)

New encoding + Hooper Theorem = WIN

Note: Proof of Hooper's Theorem is *very hard*.

Kari's idea

Two steps:

It is very easy to simulate affine maps with tilesets

A Turing Machine can be encoded into piecewise affine maps.

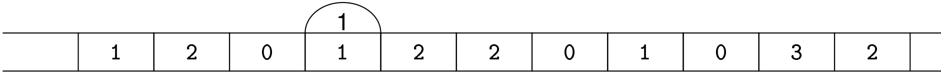
How to encode Turing machines

A Turing Machine can be encoded into piecewise affine maps.

Idea: use the *moving tape* model.

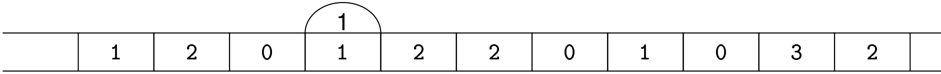
Idea

Suppose the states and symbols of the Turing machine are integers.



Idea

Suppose the states and symbols of the Turing machine are integers.



Code it into:

$$\begin{aligned} L &= 0.021\ldots \\ R &= 0.12201032\ldots \\ S &= 1 \end{aligned}$$

Encoding

Every configuration of the Turing machine will be encoded into a tuple $(L, R, S) \in \mathbb{R}^3$, representing the left part, the right part, and the state of the configuration.

How does it work

The TM is in state q and reads an x is the same as

$$S \in [q, q+1[\text{ and } R \in [x/10, x/10 + 0.1[$$

How does it work

We replace the x by a y is the same as

$$R \leftarrow R + (y - x)/10$$

How does it work

We move the head to the right is the same as

$$L \leftarrow (L + x)/10$$

$$R \leftarrow 10(R - x/10)$$

How does it work

If we are in state q and read x, we output y, change to state q' and we move to the right:

$$f(L, R, S) = ((L + y)/10, 10(R - x/10), q')$$

$$\text{if } S \in [q, q+1[\text{ and } R \in [x/10, x/10 + 0.1[$$

Result

A Turing machine is essentially the same as a piecewise affine map

Caveats

- The base in which we represent the tape should be strictly bigger than the alphabet.
    - To avoid $0.099999\cdots = 0.100000\ldots$
- Some inputs of the piecewise affine map do not correspond to “real” tapes
    - This is usually not a problem

Result

Theorem (Hooper 1966)

There is no algorithm that decides, given a Turing machine, if there exists SOME configuration on which it runs forever.

Theorem

There is no algorithm that decides, given a 3D piecewise affine map f, if there exists x s.t $f^n(x)$ is defined for all n.

Now it remains to..

Kari’s idea

It is very easy to simulate affine maps with tilesets

(if they have rational coefficients)

We will only explain how to simulate a map $f : x \rightarrow ax + b$. Adding linear constraints ($x < c$) or going to higher dimensions is easy.

Idea

One row of colors will represent a real number x.

One row of a tiling represents transitions between two rows of color (the colors at the south, and the colors at the north)

We will design the tileset so that one row of the tiling will represent the operation $x \to f(x)$.

Question

How do we represent real numbers as biinfinite sequences ?

The β-sequence

Definition

The β-sequence for a number x is the biinfinite sequence

$$\beta_n(x) = \lfloor nx \rfloor - \lfloor (n-1)x \rfloor$$

Examples:

- $\beta(0) = \ldots 000000000000000000000000 \ldots.$
- $\beta(1/2) = \ldots 010101010101010101010101 \ldots.$
- $\beta(1/3) = \ldots 100100100100100100100100 \ldots.$
- $\beta(\sqrt{2} - 1) = \ldots 001010010100101010010100 \ldots.$

Properties of β-sequences

- $\beta_n(x) \in \{\lfloor x \rfloor, \lfloor x + 1 \rfloor\}$
- n consecutive bits of $\beta(x)$ gives a good approximation of x:

$$\frac{\sum_{i=k+1}^{k+n} \beta_i(x)}{n} = \frac{\lfloor (k+n)x \rfloor - \lfloor kx \rfloor}{n} \in [x - \frac{1}{n}, x + \frac{1}{n}]$$

Backwards

We have obtained a map from real numbers to biinfinite sequences of 0 and 1.

We need a reverse map, for example:

$$r(w) = \liminf_{n \to +\infty} \frac{\sum_{i=-n}^{n} w_i}{2n+1}$$

Notice that $r(\beta(x)) = x$.

Main theorem

Theorem

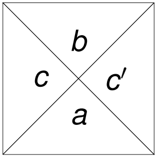
For every $q \in \mathbb{Q}$ there exists a tileset τ s.t.

- *For every x, there exists a tiling of a row with $\beta(x)$ at the bottom and $\beta(qx)$ at the top*
- *In a tiling of a row by τ, we have $r(t) = qr(b)$ where t and b denote the colors on the top and bottom of the row.*

First item: The tileset simulates multiplication by q if fed the right input.
Second item: It also simulates multiplication by q on bad inputs.

Idea of the proof

The tileset τ consists of all tiles of the form



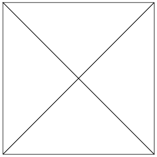
with $c, c' \in [-q, 1]$ and

$$qa + c = b + c'$$

Ideally we would like $qa = b$. c represents a carry from the past, and c' a carry we will give to our future.

First point

This works if we have $\beta(x)$ on bottom and $\beta(qx)$ on top


$\beta_n(qx)$
$q\lfloor (n-1)x \rfloor - \lfloor q(n-1)x \rfloor$
$q\lfloor nx \rfloor - \lfloor qnx \rfloor$
$\beta_n(x)$

Second point

Suppose we have t on top and b at bottom.
Then

$$q \sum_{i=-n}^{n} b_i + c_{-n} = \sum_{i=-n}^{n} t_i + c_n$$

Therefore

$$\frac{q \sum_{i=-n}^{n} b_i}{2n+1} = \frac{\sum_{i=-n}^{n} t_i}{2n+1} + \frac{c_n - c_{-n}}{2n+1}$$

And

$$qr(b) = r(t)$$

as $c_n - c_{-n} = O(1)$

Slight problem

Note: we need a *finite* tileset

We only need carries that can occur on the β-sequences. If $q = n/m$ we only need carries in $[-q, 1]$ that are rational fractions of m, and there are finitely many of them.

Main theorem

Theorem

For every piecewise affine map f with rational coefficients, there exists a tileset τ s.t.

- *For every x, there exists a tiling of a row with $\beta(x)$ at the bottom and $\beta(f(x))$ at the top*
- *In a tiling of a row by τ, we have $r(t) = f(r(b))$ where t and b denote the colors on the top and bottom of the row.*

End of the proof

Let f be a piecewise affine map.

Suppose there exist n s.t. $f^n(x)$ is defined for all n.

Then there exists a tiling of a half plane with $\beta(f^n(x))$ as the bottom color of the n-th row

(By the first point)

Therefore there exists a tiling of the plane by compactness

End of the proof

Let f be a piecewise affine map.

Suppose there exist a tiling of the plane

Let b_n be the color at the bottom of the n-th row.

Then $r(b_n) = f(r(b_{n-1}))$ and therefore $r(b_n) = f^n(x)$ for $x = r(b_0)$

(By the second point)

Therefore $f^n(x)$ is defined for all n.

End of the proof

There is a tiling of the plane by τ iff there exists x s.t. $f^n(x)$ is defined for all n.

No algorithm can solve the latter problem, therefore no algorithm can solve the former.

Final remark

The proof is considerably simpler, but we have to admit the theorem of Hooper.

- Somehow, the difficulty has been shifted into the proof of the theorem of Hooper
- The proof of Hooper's theorem uses tricks similar to Robinson (in particular substitutions)

Plan

1. Berger-Robinson
2. The Kari construction
   - Encoding of TM
3. Conclusion

Conclusion

Two different proofs of the Undecidability of the Domino Problem

- One more proof in the lecture notes
- One last proof will be given in the full version of the lecture notes.