Computation and information in sofic shifts

Linda Brown Westrick
University of Connecticut

January 7, 2017
ASL Winter Meeting and JMM
Atlanta

Outline

- **Symbolic dynamics**
- Tiling problems
- Computation and sofic shifts
- Self-similar Turing machine tilings (DRS 2012)
- Seas of squares and other sofic shifts

Symbolic Dynamics

Dynamical system (X, T), T invertible. Any $x \in X$ has a *trajectory*

$$\ldots, T^{-1}(x), x, T(x), \ldots$$

If A is a finite set and $p : X \to A$ partitions X, then x has a *symbolic trajectory*

$$\ldots, p(T^{-1}(x)),\ p(x),\ p(T(x)), \ldots$$

Left-shifting the symbolic trajectory of x gives the symbolic trajectory of $T(x)$.

Definition. Let A be a finite alphabet. A *subshift* is a subset of $A^{\mathbb{Z}}$ that is topologically closed and closed under the shift operation.

A subshift with the shift operation is a dynamical system. Conversely, one can often recover x from its symbolic trajectory, so subshifts can encode general dynamical behaviors.

Classes of subshifts

Equivalent definition. A *subshift* is a subset $X \subseteq A^{\mathbb{Z}}$ obtained by avoiding some set of "forbidden" strings.

Example 1: $X = \{x \in 2^{\mathbb{Z}} : \text{ the string 11 never appears in } x\}$
Example 2: $X = \{x \in 2^{\mathbb{Z}} : \text{ no string of the form } 10^{2n}1 \text{ appears in } x\}$

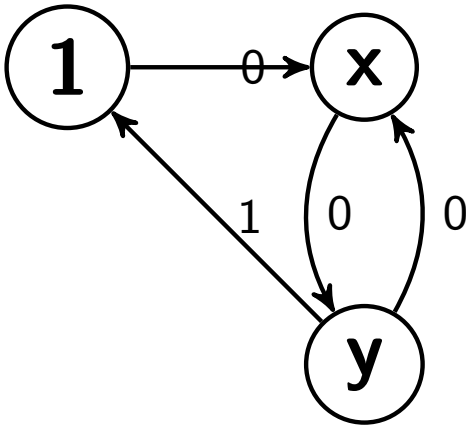
- A subshift is called a **shift of finite type** if it can be obtained by forbidding a finite set of strings.
- A subshift X on an alphabet A is called **sofic** if there is a shift of finite type Y on an alphabet B, and a map $f : B \to A$, such that $X = f(Y)$ (abusing some notation here)
- A subshift is **effectively closed** if it can be obtained by forbidding a computably enumerable set of strings; or equivalently, a computable set.

Sofic shifts and regular languages

A subshift X on an alphabet A is called *sofic* if there is a shift of finite type Y on an alphabet B, and a map $f : B \to A$, such that $X = f(Y)$

A set of strings F is *regular* if there is a finite automaton which accepts exactly the strings from F.

A subshift is sofic if and only if it is obtained by forbidding a regular set F.



1	x	y	x	y	1	x	y
1	0	0	0	0	1	0	0

Two-dimensional subshifts

A *two-dimensional subshift* is a subset $X \subseteq A^{\mathbb{Z}^2}$ which is topologically closed and closed under both shift operations.

Equivalently, the two-dimensional subshifts are exactly those obtained by forbidding a set of rectangular patterns.

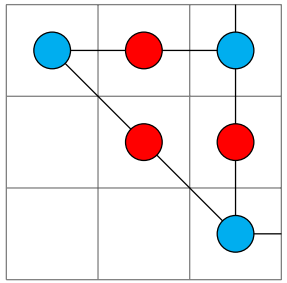
Example: Forbid  and  , get the subshift of configurations with constant columns.

Define SFTs, sofic shifts, and effectively closed shifts as before.

SFT and sofic examples

SFT example
Consider an alphabet B of red and blue circles and line segments.

The shift of *two-colored* configurations is an SFT.



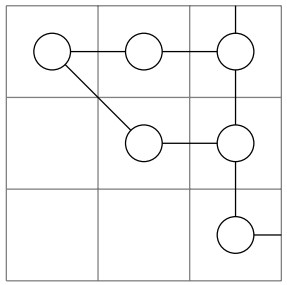
Consistent pattern.

Let A be the same alphabet as B, minus the colors.

Letting $f : B \to A$ be the map that forgets the colors gives:

Sofic example
The subshift of *two-colorable* configurations is sofic.

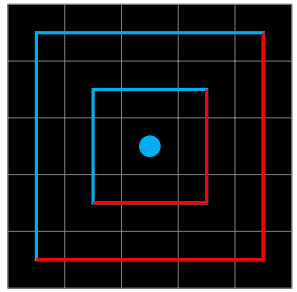
This subshift is not an SFT, due to the possibility of large cycles.



Forbid due to 5-cycle

SFT and sofic examples

SFT example
The subshift whose elements consist of non-overlapping annotated black squares on a white background.



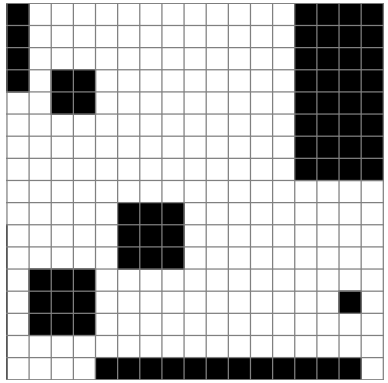
Annotated square.

Letting f be the map that forgets the annotations:

Sofic example

The subshift whose elements consist of non-overlapping black squares on a white background.

This subshift is *not* an SFT, due to large rectangles.



Consistent with sea of squares, do not forbid.

Outline

- Symbolic dynamics
- **Tiling problems**
- Computation and sofic shifts
- Self-similar Turing machine tilings (DRS 2012)
- Seas of squares and other sofic shifts

Tiling problems

A *Wang tileset* is a finite set of square tiles with colored edges.

Tiles may be placed adjacent to each other if they have the same color on the edge that they share.

The tiling problem asks: given a tileset, is there a legal way to fill the plane with tiles from that tileset? (Tiles can be used more than once.)

Every tileset specifies a (possibly empty) two-dimensional SFT. The tiles are the alphabet, and the forbidden patterns are pairs of adjacent tiles with mismatched edges.

Conversely, every two-dimensional SFT can be realized by a tileset.

Turing computations in tilings/SFTs

Wang (1962) showed how to design tilesets with infinite tilings which depict space-time diagrams of arbitrary Turing computations.

Given a fixed Turing machine with states q_i and tape alphabet $\{0, 1, b\}$, forbid all 2×3 patterns that could never appear in that machine's space-time diagram.

		Δ
q_t 1	q_t 0	q_t b
	Δ	
q_s 1	q_s b	q_s b

Forbid this jumping head.

Δ
q_0 b

Anchor symbol.

Result: any configuration that contains the anchor symbol contains the space-time diagram of the TM on empty input.

Removing the anchor tile

There is an infinite tiling with anchor tile if and only if the given Turing machine runs forever.

How to force the anchor tile to appear?

Berger (1966). Finite computations of all sizes, fractally arranged.

Corollary. The tiling problem is undecidable.

Hanf, Meyer (1974). There are nonempty SFTs with no computable elements.

Outline

- Symbolic dynamics
- Tiling problems
- **Computation and sofic shifts**
- Self-similar Turing machine tilings (DRS 2012)
- Seas of squares and other sofic shifts

A naive analogy

One-dimensional sofic shifts support hidden *finite automaton* computation and are characterized by forbidding *regular* sets of strings.

Could two-dimensional sofic shifts, which support hidden *Turing* computation, be characterized by forbidding *computably enumerable* sets of local patterns?

No. This is equivalent to asking if the two-dimensional sofic shifts coincide with the effectively closed shifts.

Every sofic shift is effectively closed, but the inclusion is strict.

Computation in sofic shifts

Examples of effectively closed, non-sofic shifts:

- The mirror shift
- 2-dimensional shift-complex shift (Rumyantsev-Ushakov 2006)
- Stacked 1-dimensional effectively closed shifts without a synchronizing word (Pavlov 2013)

Motivating theme: Understand the power and limitations of computation in two-dimensional sofic shifts.

Open questions in the area

If X is a one-dimensional shift obtained by forbidding strings from F, its *stacked shift* is the two-dimensional shift with the same forbidden strings, now considered as rectangular patterns.

If X is sofic, then its stacked shift is sofic.

Question (Jeandel). If X is a one-dimensional shift whose stacked shift is sofic, must X be sofic?

There are even many examples of 1-dimensional non-sofic X, where the soficity of stacked-X is not known.

Question (old). Is every two-dimensional sofic shift obtainable as image of a two-dimensional SFT with the same entropy?

Recent progress

Hochman and Meyerovitch (2010). The entropies of the two-dimensional SFTs and sofic shifts are exactly the right-computably enumerable numbers.

Given an underlying pattern, they use a superimposed Turing computation to approximate the desired entropy, halting the computation to kill the pattern if it seems to have too much entropy.

Durand, Romashchenko and Shen (2012). Every effectively closed two-dimensional shift whose configurations have constant columns is sofic. (also independently obtained by Aubrun and Sablik 2013).

Unavoidable reliance on hidden Turing computations.

Results

Definitions:

- For any set $S \subseteq \mathbb{N}$, let the **S-square shift** be the $\mathbb{Z}^2$-shift on the alphabet {black, white} whose configurations consist of seas of non-overlapping black squares on a white background, where the size of each square is in S.
- Let the **distinct-square shift** consist of the configurations in which no finite size of square is repeated.
- A $\mathbb{Z}^2$-shift is **α-sparse** if there is a constant C such that the shift forbids every $N \times N$ pattern with more than CN^α black symbols.

Theorem (W):
The following shifts are sofic:

- The S-square shift for any Π^0_1 set S.
- Any effectively closed subshift of the distinct-square shift.
- Any effectively closed α-sparse shift for $\alpha < 1$.

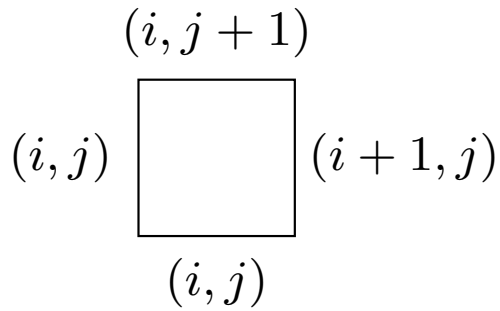
Outline

- Symbolic dynamics
- Tiling problems
- Computation and sofic shifts
- **Self-similar Turing machine tilings (DRS 2012)**
- Seas of squares and other sofic shifts

Durand, Romashchenko & Shen (2012)

DRS define a tileset:

- Tiles organize themselves into $N \times N$ regions.



- Each region has a space-time diagram on the inside, but viewed from the outside, the region is a tile, or "macrotile".
- The macrotiles behave just like the original small tiles, but with larger N.
- This behavior is enforced by the computation happening in the tile.

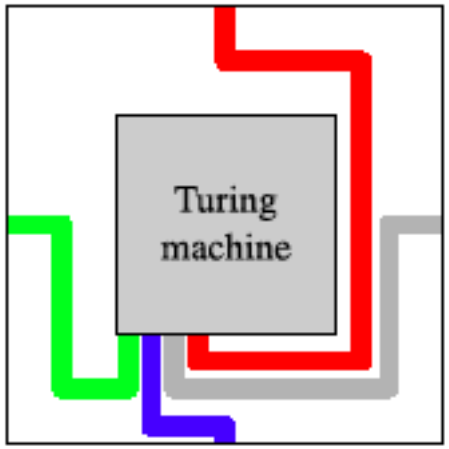


Image source: DRS 2012

Parent Tile, Child Tile

Consider a parent "macrotile" made from an $N \times N$ array of child tiles.

The parent computation

- Accepts the "data" of what colors are being displayed at the edge of the region as input.
- Analyzes the input to see if the edges make a good macrotile, and kills the computation if not.
- Makes sure its own parent is running the same program.
- Also does other desired computations.

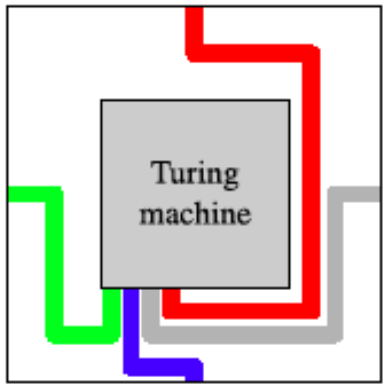


Image source: DRS 2012

Effectively closed shifts with constant columns

Theorem (Durand-Romashchenko-Shen 2012): Any effectively closed shift whose elements have constant columns is sofic.
(this result independently obtained by Aubrun-Sablik 2013)

Idea: Given an configuration with constant columns, superimpose TM tiles to

- "read" the common row
- make what has been read available at all levels
- simultaneously, enumerate forbidden $\mathbb{Z}$-patterns
- kill the element if a pattern it contains is enumerated.

Issue: How can a higher-level macrotile learn about what is written on the pixel level, since it can't interact with that level directly?

Parent-child communication

Issue: How can a higher-level macrotile learn about what is written on the pixel level, since it can't interact with that level directly?

Solution: pass info up from child to parent

- Children who are sitting on the parent tape "read" it
- Whisper to other siblings about what is there
- If the parent tape does not contain a thing which a child wants the parent to know, the child kills the tiling.

Outline

- Symbolic dynamics
- Tiling problems
- Computation and sofic shifts
- Self-similar Turing machine tilings (DRS 2012)
- **Seas of squares and other sofic shifts**

S-square shift: plan and obstacles

Plan: Given a sea of squares (unrestricted sizes), superimpose TM tiles to

- "read" and record the sizes of squares that appear inside them
- propagate this information to their parents
- simultaneously, enumerate forbidden sizes
- kill the element if one of the collected sizes is enumerated

Obstacles:

- A forbidden-size square can appear once and disqualify the whole sea, so each tile must record every single size inside itself.
- The parent's parameter tape becomes too large for children to copy it, yet each child must make sure the parent received its records.
- The input to each computation region is large relative to the region; the algorithm must run in less than quadratic time to fit inside.

Recording all the sizes

A macrotile at level k has $\sim N_{k-1}$ tape size and a pixel width of $L_k = N_{k-1} \dots N_1 N_0$.

To record all sizes from a macrotile at level k, $\sim L_k^{2/3}$ bits are needed.

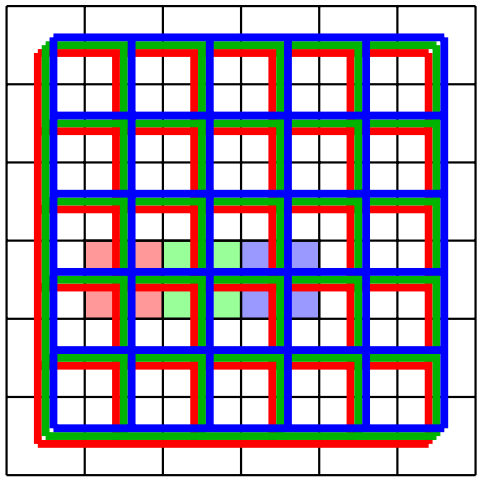
For that to fit on the tape, we let N_k grow so fast $L_k^{2/3} << N_{k-1}$.

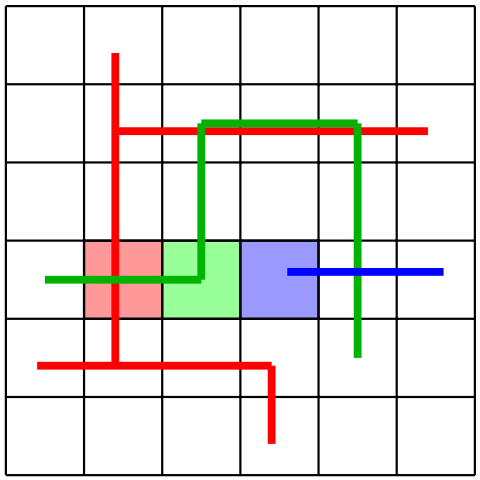
In DRS, all bits of the parent's parameter tape are passed among all children. Impossible here:

Bits of parent data $\approx L_{k+1}^{2/3} > N_k^{2/3} >> N_{k-1} \approx$ length of child tape.

Communicating with the parent

Idea: Each child nondeterministically chooses what parental information to share with each of its neighbors, and hopes to receive parental reassurance about each of its own recorded sizes.





Left: sharing everything

Right: selective sharing

Use a counter to certify the information is genuinely from the parent.

A cooperative game of Ticket to Ride

There are $\sim N_k^2$ vertices (cities, child tiles), arranged in a square grid.

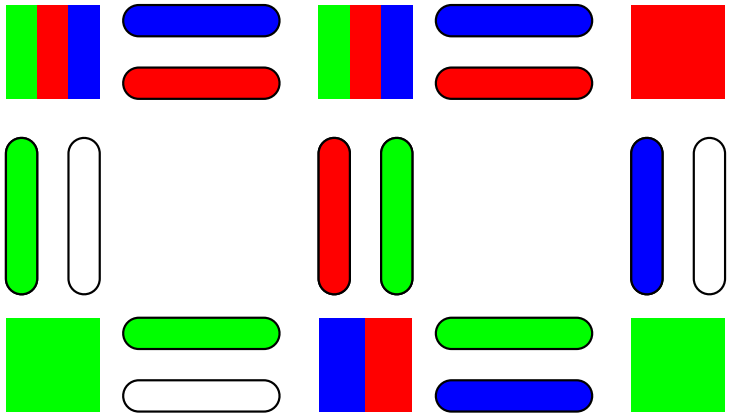
There are $\sim L_{k+1}^{2/3}$ players (train companies, parental records).

Each vertical or horizontal edge (connector, child side color) has $\sim L_k^{2/3}$ tracks.

In any $N \times N$ subgrid of vertices, at most $\sim (NL_k)^{2/3}$ players have a city in that grid.

The players cooperatively win if there is a way to divvy up the tracks so that every player can connect all their cities together.

The S-square algorithm works if and only if the players can always win.



The players won.

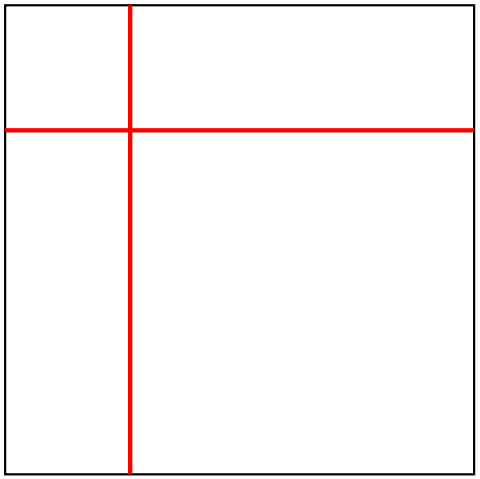
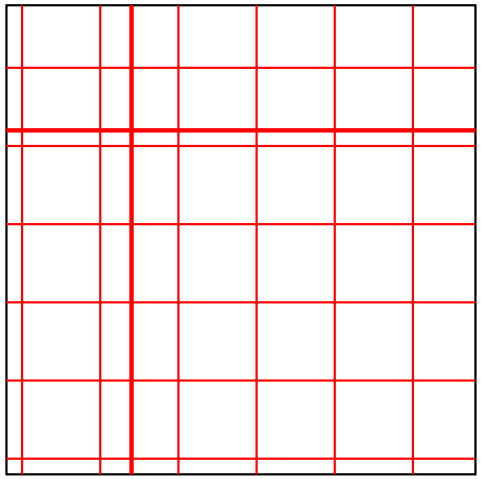
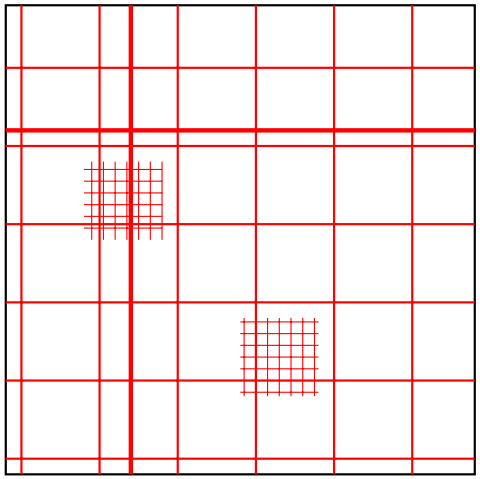
Multiscale plaid concept

The players can win the game with a multiscale plaid track pattern:

- All players take turns laying vertical tracks, top-to-bottom, as tightly as reasonable ($L_k^{2/3}$ players per vertical track.)
- All players lay horizontal tracks in the same fashion. (1st layer of plaid).
- This makes natural square subregions, in which each player has a vertical and horizontal track.
- Within each $N \times N$ subregion, $N(L_k)^{2/3}$ players have tracks, but only $(NL_k)^{2/3}$ players have cities there.
- Make another layer of tight plaid, within that subregion only, using only the players that have cities in that subregion.
- This tighter plaid makes smaller subregions, more players drop out.
- Recurse in all subregions until some fixed small size of subregion is reached, then let the small number of remaining players connect directly to their cities.

Multiscale plaid analysis

All players make a single connected component that includes all their cities.

How many tracks per edge were used?

- At each level of recursion, $L_k^{2/3}$ tracks per edge.
- Some fixed constant number of tracks per edge for the bottom step.
- Using $N_k = 2^{2^{2^k}}$, there are $\sim 2^k$ levels of recursion.
- Relative to $L_k^{2/3}$, this 2^k is an ignorable log factor.

Thank you.