Seas of squares with sizes from a Π^0_1 set

Linda Brown Westrick
Victoria University of Wellington

June 20, 2016
CIRM, Marseille

Outline

- **Subshifts**
- Self-similar Turing machine tilings (DRS 2012)
- Seas of squares

Some classes of subshifts

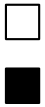
Definitions. Let A be a finite alphabet. Let d be a positive integer. In this talk, usually $d = 2$ and sometimes $d = 1$.

- A **subshift** is a subset $X \subseteq A^{\mathbb{Z}^d}$ which is obtained by forbidding some set of local patterns.
- A local pattern is an element of A^D where D is any finite subset of $\mathbb{Z}^d$
- If F is a set of local patterns,
  $\{x \in A^{\mathbb{Z}^d} :$ for all $p \in F$, p does not appear in $x\}$ is a subshift.
- A subshift is called a **shift of finite type** if it can be obtained by forbidding a finite set of local patterns.
- 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 c.e. set of local patterns; or equivalently, a computable set.

SFT examples

A a finite alphabet
$d = 1, 2$

- **Subshift** $X \subseteq A^{\mathbb{Z}^d}$ all elements that omit forbidden patterns
- **SFT** finitely many forbidden patterns
- **Sofic** $X = f(Y)$ SFT $Y \subseteq B^{\mathbb{Z}^d}$ $f : B \to A$.
- **Effectively closed** c.e. set of forbidden patterns.

- Forbid  and , get the subshift of configurations with constant columns.
- Given a fixed Turing machine with states q_i, forbid all 2×3 patterns that could never appear in that machine's space-time diagram.

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

Forbid this jumping head.

Δ q_0 b

Anchor symbol.

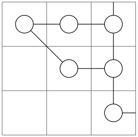
Result: any configuration that contains the anchor symbol contains the space-time diagram of the TM on empty input.

Sofic Examples

A a finite alphabet
$d = 1, 2$

- **Subshift** $X \subseteq A^{\mathbb{Z}^d}$ all elements that omit forbidden patterns
- **SFT** finitely many forbidden patterns
- **Sofic** $X = f(Y)$ SFT $Y \subseteq B^{\mathbb{Z}^d}$ $f : B \to A$.
- **Effectively closed** c.e. set of forbidden patterns.

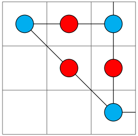
- Any SFT. $(A = B, X = Y)$.
- Subshift of two-colorable configurations



Forbid due to 5-cycle

This subshift is not an SFT.
Reason: large cycles.

However, the shift of *two-colored* configurations is an SFT. Letting f be the map that forgets the colors gives the two-colorable configurations.



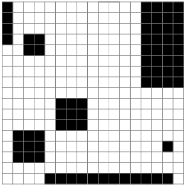
Consistent pattern.

Sofic Examples

A a finite alphabet
$d = 1, 2$

- **Subshift** $X \subseteq A^{\mathbb{Z}^d}$ all elements that omit forbidden patterns
- **SFT** finitely many forbidden patterns
- **Sofic** $X = f(Y)$ SFT $Y \subseteq B^{\mathbb{Z}^d}$ $f : B \to A$.
- **Effectively closed** c.e. set of forbidden patterns.

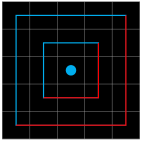
- The subshift whose elements consist of non-overlapping black squares on a white background.



Consistent with sea of squares, do not forbid.

This subshift is *not* an SFT.
Reason: large rectangles.

Why sofic?
Expand the alphabet to provide witness of square-ness:



Relation SFT $\subseteq$ sofic $\subseteq$ effectively closed

A a finite alphabet
$d = 1, 2$

- **Subshift** $X \subseteq A^{\mathbb{Z}^d}$ all elements that omit forbidden patterns
- **SFT** finitely many forbidden patterns
- **Sofic** $X = f(Y)$
  SFT $Y \subseteq B^{\mathbb{Z}^d}$
  $f : B \to A$.
- **Effectively closed** c.e. set of forbidden patterns.

Every SFT is sofic. $(A = B,\ X = Y)$.

Every sofic shift is effectively closed. Algorithm: given Y and f, and given a pattern p in alphabet A, forbid p if and only if for all $q \in f^{-1}(p)$, q is forbidden in Y.

These implications are strict.

Motivating question

What properties of a c.e. set of forbidden words can guarantee that the resulting effectively closed shift is sofic?

Further examples

Sofic shifts

- Various substitution-rule shifts
- Even connected components shift
- Odd connected components shift (Cassaigne, unpublished)
- Stacked 1D sofic shifts
- Any effectively closed shift whose configurations have constant columns (Durand-Romashchenko-Shen 2012, Aubrun-Sablik 2013)
- Effective S-adic systems (Aubrun-Sablik 2014)

Effectively closed, non-sofic shifts

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

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

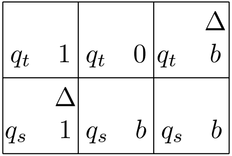
- Subshifts
- **Self-similar Turing machine tilings (DRS 2012)**
- Seas of squares

Forcing Turing computations in SFT configurations

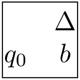
A tileset is a finite set of squares (tiles) with colored edges
Two tiles can go next to eachother if they agree on the edge they share.

The tiling problem: given a tileset, can you tile the plane?
Observe: the set of such infinite tilings is a subshift.

Recall our SFT example:



Forbid this jumping head.



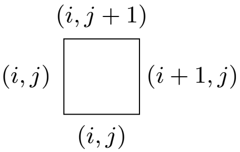
Anchor symbol.

Wang (1962) produced an essentially similar tileset.
There is a tiling with anchor tile iff the TM run forever.
Problem: how to force the anchor tile to appear?
Berger (1966) Finite computations of increasing size.

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.
- Accept the "data" of what colors are being displayed at the edge of the region as input. Analyze the input to see if the edges make a good macrotile. Kill the computation if not.
- Also do whatever computation was originally interesting.



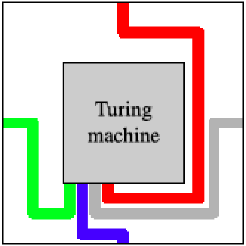


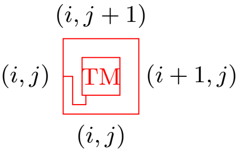
Image source: DRS 2012

Parent Tile, Child Tile

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

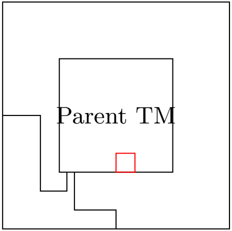
Child side colors contain:

- $2 \log N$ bits to communicate a location (i, j)
- Finite number of bits associated to a universal Turing Machine computation.
- Finite number of bits corresponding to a wire.



The child tile's computation verifies:

- Coordinates increment appropriately?
- If (i, j) is in the computation region, are TM bits coherent?
- If (i, j) is in a wire location, are wire bits coherent?
- If (i, j) is at the nth bit of the program tape for the universal TM, is the nth bit of *this program* written on the tape?



Universal TM simulation and the Recursion theorem

Assuming $N_0 < N_1 < \dots$ is the sequence of sizes of macrotiles at level i, Input size: $O(\log N_i)$.

Algorithm: Polynomial time, as written before application of the recursion theorem.

Universal TM simulation: polytime overhead

Recursion theorem: polytime overhead

Runtime of resulting program: $\text{poly}(\log N_i)$.

Available time: $N_{i-1}/2$

For appropriate choice of the sequence $\langle N_j \rangle$, we have $\text{poly}(\log N_i) << N_{i-1}/2$, so no computation runs out of room.

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?

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

- Subshifts
- Self-similar Turing machine tilings (DRS 2012)
- **Seas of squares**

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 $L_k = N_{k-1} \dots N_1 N_0$.

Maximum number of distinct sizes of square that can fit in an $L \times L$ region?
Bound by $x_1^2 + \cdots + x_m^2 < L^2$. To maximise m, let $x_i = i$.
Result: m is bounded by $\sim L^{2/3}$.

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 need: $(N_0 N_1 \dots N_{k-1})^{2/3} << N_{k-1}$.
Triple exponential $N_k = 2^{2^{2^k}}$ is fast enough. Double exponential is too slow.

Note: Unavoidably, $N_{k-1}^{2/3} << L_k^{2/3}$. Therefore, the algorithm that is run using this input must be polynomial with exponent strictly less than 3/2, or it will overrun the computation region.

Conclusion: asymptotics of holding and processing info are ok.

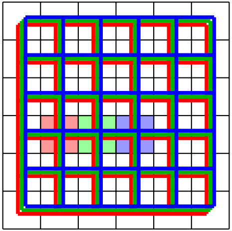
Communicating with the parent

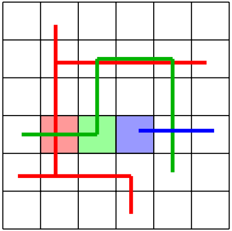
$L_k = \prod_{i=k_0}^{k-1} N_i$.

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.

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.

The question

So far our algorithm achieves:

- If the parent tape does not contain a record which some child needs, there will be no legal message chain to that child, so the tiling cannot be made.
- If the parent has all the needed records, and **IF** there is some way to simultaneously connect each record on the parent tape with the individual children who need it **without overloading any child by passing too many records through it**, the children will nondeterministically find this way.

So, is there always a way?

A cooperative game of Ticket to Ride

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

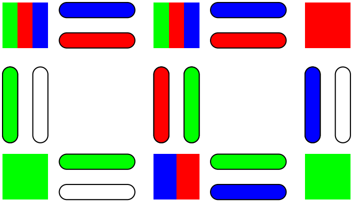
There are $\sim L_{k+1}^{2/3}$ players (train companies, parental records).

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.

Necessity of the $N \times N$ subgrid condition

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.

At most $\sim L_k^{2/3}$ players care about any given city.

Counterexample:

- Consider a square subgrid of cities where each city has the full $\sim L_k^{2/3}$ number of players, but each player has at most once city.
- Side length of this subgrid is $N_k^{1/3}$
- Fill the whole board with $N_k^{4/3}$ such subgrids.
- Each player must connect $N_k^{4/3}$ cities, each at distance $N_k^{1/3}$ from each other: $N_k^{5/3}$ connections needed
- Multplying by all players, total connections needed: $L_{k+1}^{2/3} N_k^{5/3}$.
- Total connections available: $\sim L_k^{2/3} N_k^2$.

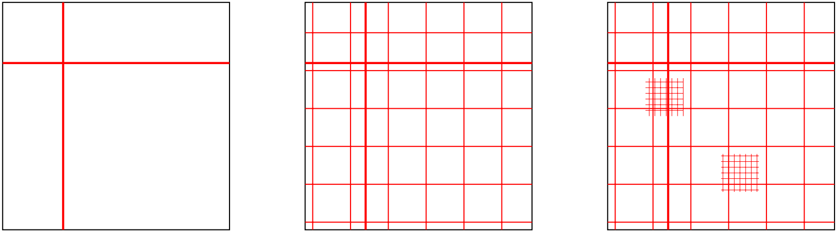
Multiscale plaid concept

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.
- Total $(2^k + C)L_k^{2/3} \sim L_k^{2/3}$ tracks per edge. Done.

Runtime considerations

We need to make sure this algorithm runs in polynomial exponent-3/2 time.

Things to check:

- Familiar operations which are fast on modern architectures are slow on Turing machines. Turns out a multi-tape TM is necessary for our algorithm to be subquadratic. (On an MTM, it is linear.)
- Good news: MTM just as easy to implement in a tiling.
- A given MTM can be simulated, with only constant overhead, by a universal MTM.
- The constant-overhead recursion theorem works.

Questions

- What bound on the growth rate of the number of $N \times N$ patterns could guarantee that a shift is sofic?
- What properties of a shift guarantee that every effectively closed subshift of it is sofic?
- (Jeandel) It is immediate that if X is a 1D sofic shift, then the 2D shift of configurations with rows belonging to X is sofic. Does the converse hold?

Thank you.