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

Linda Brown Westrick
Victoria University of Wellington

January 4, 2016
Computability, Complexity and Randomness 2016
University of Hawaii

Outline

- **Subshifts**
- Embedding Turing computations in SFTs
- Self-similar Turing machine tilings (DRS 2012)
- Seas of squares
- Entropy

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 $d = 1$.

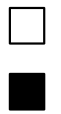
- A **subshift** is a subset $X \subseteq A^{\mathbb{Z}^d}$ which is obtained by forbidding some set of local patterns.
- A local pattern is an element of A^D where D is any finite subset of $\mathbb{Z}^d$
- If F is a set of local patterns,
  $\{x \in A^{\mathbb{Z}^d} : \text{ for all } p \in F, p \text{ does not appear in } x\}$ 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.

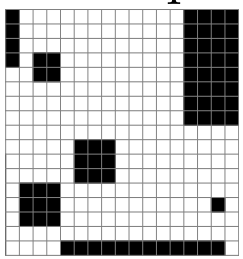
Examples: SFT, effectively closed

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.

Examples

- **SFT example:** Forbid  and  , get the subshift of elements with constant columns.
- **Effectively closed, not SFT:** Forbid any $n \times n$ pattern not consistent with a sea of black squares on a white background.



← consistent, not forbidden

This subshift is *not* an SFT

Reason: large rectangles.

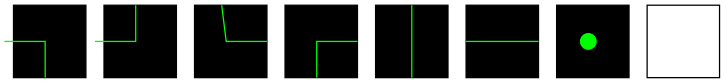
Examples: sofic, effectively closed

A a finite alphabet
$d = 1, 2$

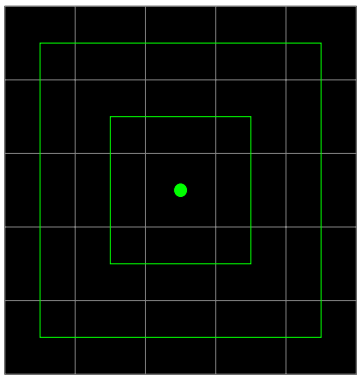
- **Subshift** $X \subseteq A^{\mathbb{Z}^d}$ all elements that omit forbidden patterns
- **SFT** finitely many forbidden patterns
- **Sofic** $X = f(Y)$ SFT $Y \subseteq B^{\mathbb{Z}^d}$ $f : B \to A$.
- **Effectively closed** c.e. set of forbidden patterns.

Examples

- **Sofic, not SFT:** Same sea of squares. Extended alphabet:



Forbid every 3×3 pattern not consistent with a sea of squares with concentric annotations.



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 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
- Stretched 1D effectively closed shift (Durand-Romashchenko-Shen 2012, Aubrun-Sablik 2013)

Effectively closed, non-sofic shifts

- 2D Shift-complex shift (Durand-Levin-Shen 2008, ?)
- Stacked 1D effectively closed shifts without a synchronizing word (Pavlov 2013)

Main result

Definition:
For any set $S \subseteq \mathbb{N}$, let the **S-square shift** be the $\mathbb{Z}^2$-shift on the alphabet {black, white} whose elements consist of seas of non-overlapping black squares on a white background, where the size of each square is in S.

Theorem (W):
For any Π^0_1 set S, the S-square shift is sofic.

Outline

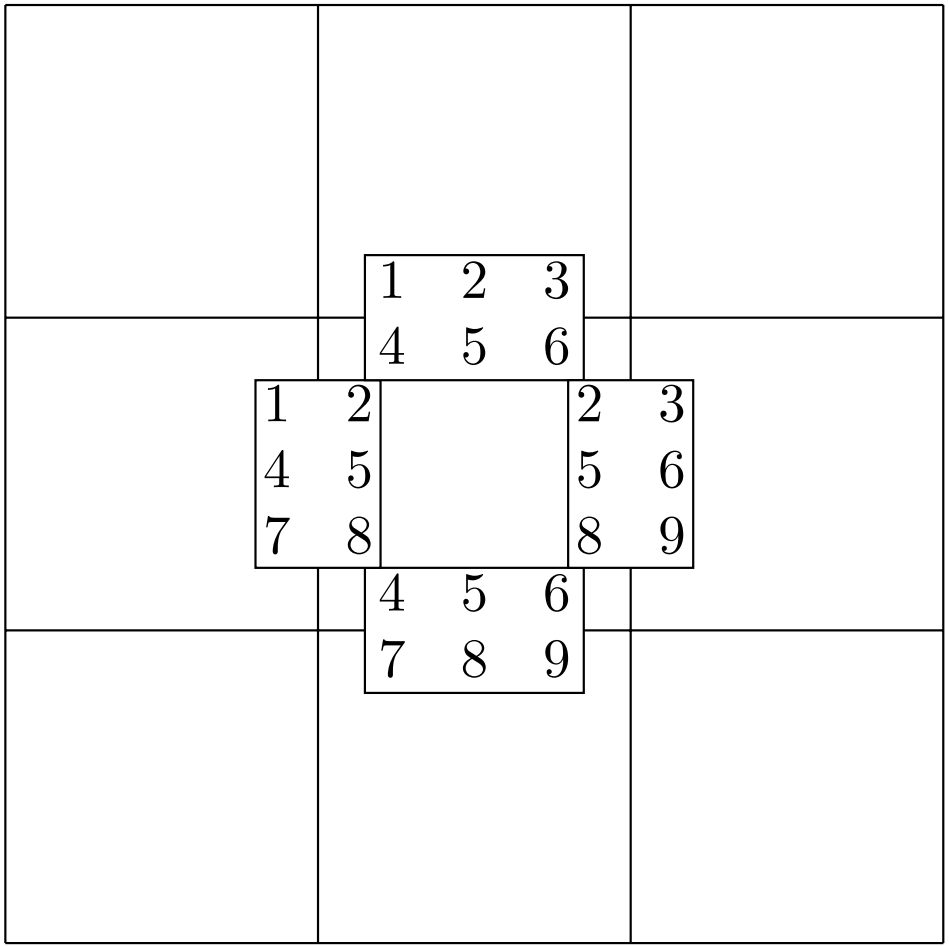
- Subshifts
- **Embedding Turing computations in SFTs**
- Self-similar Turing machine tilings (DRS 2012)
- Seas of squares
- Entropy

Tiling problems and $\mathbb{Z}^2$ SFTs

Historically, work on $\mathbb{Z}^2$ SFTs took place in the context of **tiling problems.**

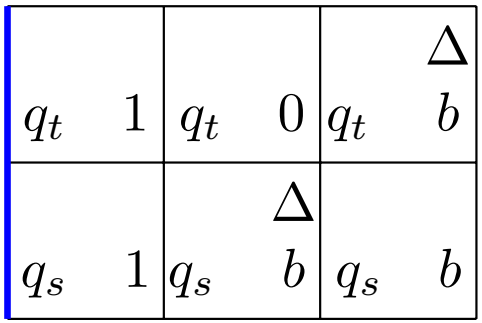
Tiling problems and $\mathbb{Z}^2$ SFTs are essentially the same thing.

1	2	3
4	5	6
7	8	9

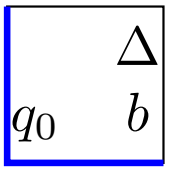


Embedding TMs with Anchor symbols

Kahr, Moore, Wang (1962): for any Turing machine, there is a finite set of tiles, with one designated "anchor tile" so that any tiling of the plane that includes the anchor tile encodes the space-time diagram of the computation of that machine.



Looks consistent.



Anchor symbol.

The anchor tile is made from the close area of the anchor symbol, like in the previous slide.

If the computation halts, the tiling cannot be continued.

Outline

- Subshifts
- Embedding Turing computations in SFTs
- **Self-similar Turing machine tilings (DRS 2012)**
- Seas of squares
- Entropy

Durand, Romashchenko & Shen (2012)

Problem: If we enforce multiple heads, the computation regions collide, making no tilings.

Another solution: DRS (2012). Using a tileset format,

- Fill the entire plane with small computation regions.
- Each region has a computation on the inside, but viewed from the outside, the region is a tile, or "macrotile".
- Double purpose computation.
- Accept the "data" of what tiles are appearing 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.

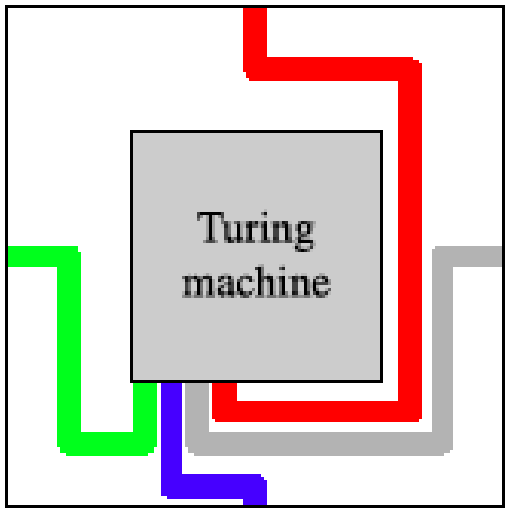


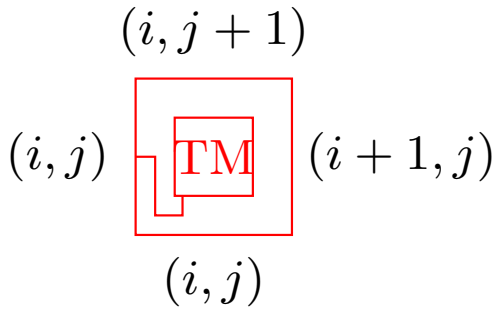
Image source: DRS 2012

Parent Tile, Child Tile

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

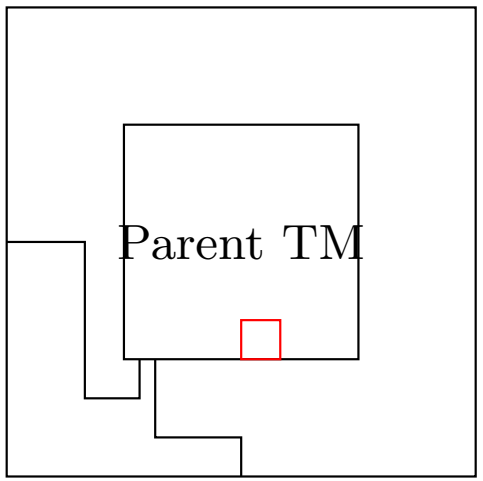
Child side colors contain:

- $2 \log N$ 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

Input size: $O(\log N)$.

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

Universal TM simulation: polytime overhead

Recursion theorem: polytime overhead

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

Available time: $N/2$

Since $\text{poly}(\log N) << N/2$, can choose N large enough that no computation runs out of room.

Expanding tilesets (DRS 2012)

The previous construction layers computations of fixed finite size $\sim N$. But computations of increasing size will be needed.

Let $N_0 < N_1 < N_2 \ldots$ be a "nice" increasing sequence.

Many possibilities: $N_k = k, k^2, 2^k, k!, 2^{2^k}, 2^{2^{2^k}}, \ldots$ $(N_k = 2^{N_{k-1}}$ too fast)

The previous construction can be modified so that every macrotile at level k is made out of $N_{k-1} \times N_{k-1}$ child tiles, giving time for ever longer computations.

From now on, all constructions involve ever increasing numbers of children to form the next parent.

Effectively closed $\mathbb{Z}$-shifts, stretched (DRS 2012)

Given any one-dimensional effectively closed shift, consider this $\mathbb{Z}^2$ shift:

- elements have constant columns
- an element's common row is contained in the given $\mathbb{Z}$-shift.

Theorem (Durand-Romashchenko-Shen 2012): All such shifts are sofic. (this result independenly 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?

Reading and remembering from constant columns

Child *parameter tape* contains:

- One short string, which this child is checking for forbidden words.
- The location of the child in its parent.

Child side colors contain:

- A copy of what is allegedly on the parent parameter tape.

Algorithm:

- If I'm seeing a bit of the parent parameter tape, check it agrees with my alleged copy.
- Use all location data (mine and parent) to figure out if the parent was supposed to learn any bit of its string from me. If so, make sure they match.

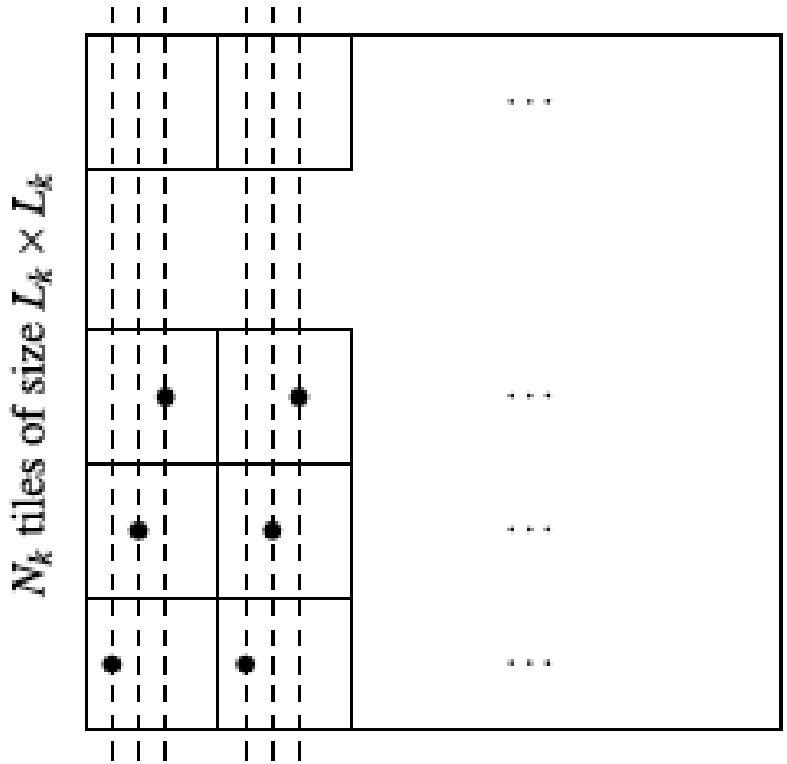


Image source: DRS 2012

Outline

- Subshifts
- Embedding Turing computations in SFTs
- Self-similar Turing machine tilings (DRS 2012)
- **Seas of squares**
- Entropy

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.

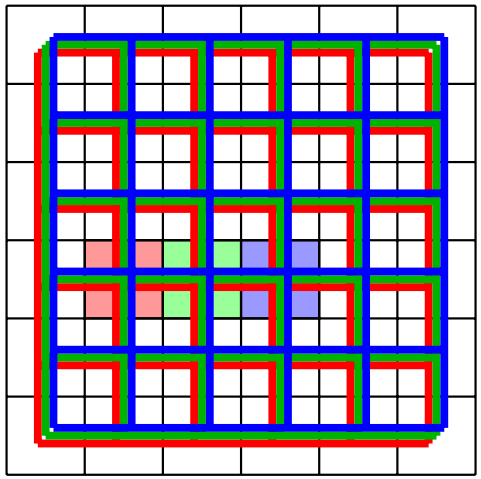
Communicating with the parent

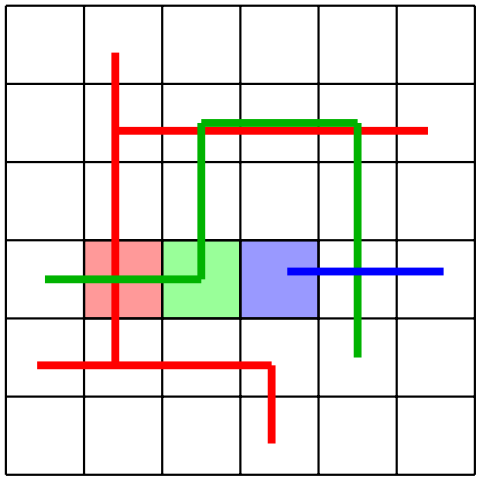
$L_k = \prod_{i=k_0}^{k-1} N_i$.

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.

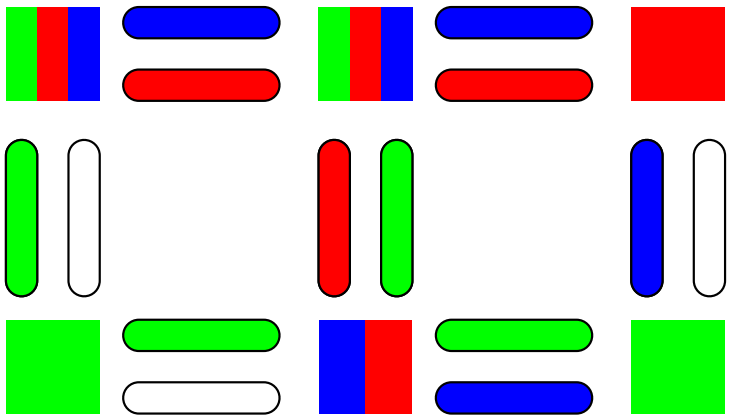
There are $\sim L_{k+1}^{2/3}$ players (train companies, parental records).

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: $\sim L_k^{2/3} N_k^2$.

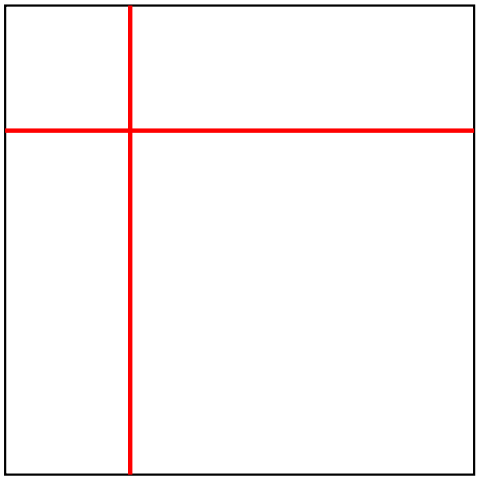
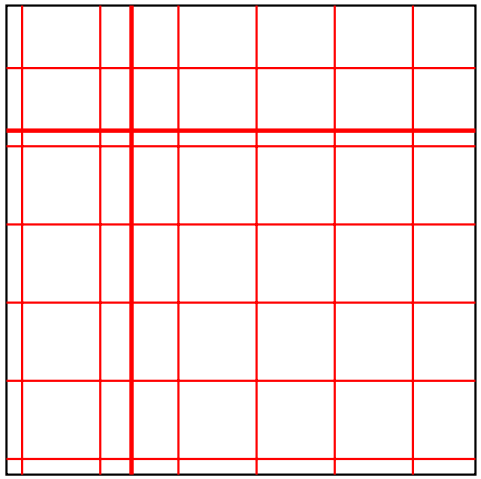
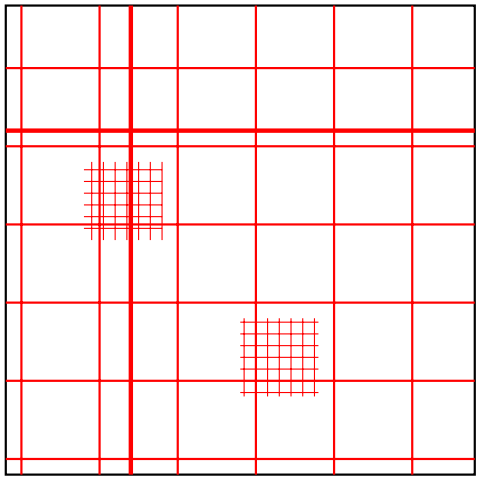
Multiscale plaid concept

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.

Algorithm skeleton

(Expanding tileset stuff is also happening, but omitted for brevity.)

Child parameter tape contains:

- List of square sizes contained within the child tile
- List of deep coordinates for squares partially within the child tile

Child side colors contain:

- List of square sizes written on the parent tape.
- List of deep coordinates of partial squares passing through this tile.

Algorithm:

- Do something that ensures the sizes on my colors are actually on the parent tape.
- Do something with deep partial square coordinates from me and my side colors to figure out sizes of some large squares contained in my parent.
- For each square size contained in me, plus the ones just discovered, check that size appears somewhere in the list of sizes on my colors.
- Enumerate forbidden sizes and kill the tiling if I have one.

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.

Outline

- Subshifts
- Embedding Turing computations in SFTs
- Self-similar Turing machine tilings (DRS 2012)
- Seas of squares
- **Entropy**

Entropy

In any subshift, given an $n \times n$ region, there is some number of ways to fill in that region without using any forbidden words.

Definition: If X is a subshift, and if the number of possibilities for an $n \times n$ region grows as 2^{sn}, then then entropy of X is s.

The entropy of a subshift cannot increase when an alphabet distinction is lost.

In one dimension, for every sofic X, there there an SFT Y such that $f(Y) = X$ and X and Y have the same entropy (Coven-Paul 1965). In two dimensions, it is open whether the same is true.

In general, superimposing a Turing machine adds no entropy, because once a computation is started, there is only one option for how to continue it.

The construction just discussed makes an SFT of greater entropy because of all the different ways the children can divvy up the parent's records. But it can be modified to cement the plaid protocol, at zero extra entropy.

References

Nathalie Aubrun and Mathieu Sablik.

Simulation of effective subshifts by two-dimensional subshifts of finite type.
Acta Appl. Math., 126:35–63, 2013.

Robert Berger.

The undecidability of the domino problem.
Mem. Amer. Math. Soc. No., 66:72, 1966.

Bruno Durand, Leonid A. Levin, and Alexander Shen.

Complex tilings.
J. Symbolic Logic, 73(2):593–613, 2008.

Bruno Durand, Andrei Romashchenko, and Alexander Shen.

Fixed-point tile sets and their applications.
J. Comput. System Sci., 78(3):731–764, 2012.

A. S. Kahr, Edward F. Moore, and Hao Wang.

Entscheidungsproblem reduced to the $\forall\exists\forall$ case.
Proc. Nat. Acad. Sci. U.S.A., 48:365–377, 1962.

Ronnie Pavlov.

A class of nonsofic multidimensional shift spaces.
Proc. Amer. Math. Soc., 141(3):987–996, 2013.

Raphael M. Robinson.

Undecidability and nonperiodicity for tilings of the plane.
Invent. Math., 12:177–209, 1971.